Vector Commitments With Proofs of Smallness: Short Range Proofs and More

Benoît Libert

Zama, France

Abstract. Vector commitment schemes are compressing commitments to vectors that make it possible to succinctly open a commitment for individual vector positions without revealing anything about other positions. We describe vector commitments enabling constant-size proofs that the committed vector is small (i.e., binary, ternary, or of small norm). As a special case, we obtain range proofs featuring the shortest proof length in the literature with only 3 group elements per proof. As another application, we obtain short pairing-based NIZK arguments for lattice-related statements. In particular, we obtain short proofs (comprised of 3 group elements) showing the validity of ring LWE ciphertexts and public keys. Our constructions are proven simulation-extractable in the algebraic group model and the random oracle model.

Keywords. Vector commitments, range proofs, ring LWE ciphertexts.

1 Introduction

Vector commitments (VCs) [81,26] allow a user to commit to a vector $\boldsymbol{m} \in D^n$ over some domain D by generating a short commitment. Later, the committer can succinctly open individual entries of $\boldsymbol{m}$. Here, "succinctly" means that the partial opening information (called "proof") should have constant size, no matter how large the committed vector is. As in standard commitments, a vector commitment scheme should satisfy two security properties: (i) The *binding* property, which ensures that no efficient adversary can open a commitment to two different values at the same position $i \in [n]$; (ii) The *hiding* property, which guarantees that revealing a subset of components does not reveal any information about messages at remaining positions.

Vector commitments found a number of applications in the context of zero-knowledge databases [81], verifiable data streaming [77], authenticated dictionaries [96], de-centralized storage [24], succinct arguments [9,78], cryptocurrencies [95] and blockchain transactions [9,61].

In this paper, we consider the problem of extending vector commitments with optimally short proofs that the committed vector $\boldsymbol{m}$ has small entries. A straightforward solution is to generically use a general-purpose succinct non-interactive argument (SNARK) for all NP languages [88]. While the SNARKs of [65,54,67] would give constant-size proofs, they would require to represent the statement as an arithmetic circuit. Then, the latter would have to compute the

opening algorithm (including exponentiations in a group) of the commitment scheme, which would result in a complex circuit. In turn, this would require a large structured common reference string (CRS) and make the proof generation very expensive since, in pairing-based SNARKs with very short proofs [54,67,50], the CRS size grows linearly with the number of multiplication gates in the arithmetic circuit. Inevitably, the computational cost of the prover grows (at least) linearly with the circuit size as well. In this paper, we aim at proving smallness more efficiently than by generically using a SNARK for all NP statements.

1.1 Our Contributions

We revisit the vector commitment scheme of Libert and Yung [81] and propose a technique allowing to argue the smallness of committed vectors without changing the scheme. Using a very small number of group elements (typically 2 or 3), we can prove that a committed vector is binary, ternary or that it has small infinity norm. By slightly increasing the proof length, we can also prove that a committed vector has small Euclidean norm or a small Hamming weight.

As a key building block, we describe a technique of generating a short proof that a committed $\boldsymbol{m}$ has binary entries using only two group elements. This argument of binarity is proven knowledge-sound in the combined algebraic group model (AGM) [48] and random oracle model. In addition, the scheme retains the useful properties of the original vector commitment [81]. In particular, its CRS size remains linear in the dimension n of committed vectors and it remains possible to succinctly open the commitment for individual vector positions. As in [80], it is also possible to prove that a committed (binary) $\boldsymbol{m} \in \mathbb{Z}_p$ satisfies a linear equation $\langle \boldsymbol{m}, \boldsymbol{t} \rangle = x$ for a public $\boldsymbol{t} \in \mathbb{Z}_p^n$ and a public $x \in \mathbb{Z}_p$. Finally, it retains the aggregation properties [80,61] that make it possible to generate a constant-size proof for a sub-vector opening.

As a first application of our arguments of binarity, we obtain a new construction of range proof featuring extremely short proofs. Regardless of the range magnitude, each proof consists of only 3 group elements, which matches the proof size of Groth's SNARK [67] and improves upon the shortest known range proof due to Boneh *et al.* [12]. The construction extends to simultaneously prove possibly distinct ranges for the individual entries of a vector $\boldsymbol{x} = (x_1, \ldots, x_m)$ without affecting the proof size. As a special case, it implies very short proofs that a committed $\boldsymbol{x} \in \mathbb{Z}^n$ has small infinity norm.

As a second main application, we provide short pairing-based non-interactive zero-knowledge (NIZK) arguments for many natural statements appearing in lattice-based cryptography. Specifically, we can argue knowledge of small-norm elements $s_1, \ldots, s_n$ of a cyclotomic ring $R = \mathbb{Z}[X]/(X^d + 1)$ that satisfy a linear relation $\sum_{i=1}^{M} \boldsymbol{a}_i \cdot s_i = \boldsymbol{t}$, for public vectors of ring elements $\boldsymbol{a}_1, \ldots, \boldsymbol{a}_M, \boldsymbol{t} \in R_q^N$, where $R_q = R/(qR)$. Using only 3 group elements, we can prove the validity of a ring LWE (RLWE) ciphertext [85], an RLWE public key, or even FHE ciphertexts [17,32]. We can also prove that a committed vector is a solution to an instance of the subset sum problem, which is useful for all the applications considered in [45]. For the specific task of proving the validity of a ciphertext in

the Lyubashevsky-Peikert-Regev cryptosystem [85], we provide efficiency comparisons with Groth's SNARK [67], which is the state-of-the art construction featuring the same proof size. We estimate that the size of the common reference string is reduced by a factor 2. While slower on the verifier's side, our scheme decreases the number of exponentiations at the prover by a factor 4. The reason is that, on the prover and verifier sides, the number of exponentiations only depends on the length of the witness and not on the size of the arithmetic circuit describing the relation. Our construction thus provides a more balanced tradeoff than SNARKs between the complexities of the prover and the verifier. As such, it can be useful in cloud or blockchain applications where it is desirable to minimize the overhead of the client even at the cost of increasing the workload of the server. For example, in FHE-based private smart contracts [37,93] (which explicitly require ZK proofs of input awareness), a resource-constrained client has to prove the validity of its input FHE ciphertexts before sending them to a computationally powerful server performing homomorphic operations.

Our NIZK arguments of range membership and ciphertext validity can be proven simulation-extractable in the algebraic group model [48] and the random oracle model (recall that all such succinct arguments have to rely on an idealized model [57]). Simulation-extractability guarantees knowledge-soundness even when the adversary can observe proofs generated by honest parties. It thus provides non-malleability [42] guarantees against a malicious prover attempting to create a proof of its own by mauling honestly generated proofs. As pointed out in, e.g., [53,51], it is an important security property in all applications where succinct arguments are easily observable in the wild. For example, if a malleable range proof is used to demonstrate the validity of confidential transactions (as in the use case of [19]), it may fail to ensure transaction independence.

Luckily, we can prove simulation-extractability without increasing the proof length while even the random-oracle-optimized variants [16,4] of Groth's SNARK have longer proofs. For the optimal proof length, existing SNARKs either provide a relaxed flavor of simulation-extractability [3] or they are significantly more demanding [68] than [67] in terms of CRS size and proving time.

1.2 Technical Overview

In asymmetric pairings $e : \mathbb{G} \times \hat{\mathbb{G}} \to \mathbb{G}_T$, the scheme of [81] uses a CRS containing group elements $(g, \{g_i = g^{(\alpha^i)}\}_{i \in [2n] \setminus \{n+1\}})$ and $(\hat{g}, \{\hat{g}_i = \hat{g}^{(\alpha^i)}\}_{i=1}^n)$. The sender commits to $\boldsymbol{m} = (m_1, \ldots, m_n) \in \mathbb{Z}_p^n$ by choosing $\gamma \xleftarrow{R} \mathbb{Z}_p$ and computing

$$C = g^\gamma \cdot \prod_{j=1}^{n} g_j^{m_j} = g^{\gamma + \sum_{j=1}^n m_i \cdot \alpha^j} \quad .$$

To open a position $i \in [n]$ of $\boldsymbol{m}$, the committer reveals a proof

$$\pi_i = g_{n+1-i}^{\gamma} \cdot \prod_{j=1, j \neq i}^{n} g_{n+1-i+j}^{m_j} = \left(C / g^{m_i \cdot \alpha^i}\right)^{\alpha^{n+1-i}}$$

which is verified by checking that $e(C, \hat{g}_{n+1-i}) = e(\pi_i, \hat{g}) \cdot e(g_1, \hat{g}_n)^{m_i}$.

To aggregate multiple proofs, PointProofs [61] uses the observation [80] that the commitment of [81] allows proving that a committed $\boldsymbol{m} \in \mathbb{Z}_p^n$ satisfies an inner product relation $\langle \boldsymbol{m}, \boldsymbol{t} \rangle = x$ for public $\boldsymbol{t} = (t_1, \ldots, t_n) \in \mathbb{Z}_p^n$ and $x \in \mathbb{Z}_p$. By raising the verification equation to the power $t_i \in \mathbb{Z}_p$ and taking the product over all $i \in [n]$, we obtain

$$e\big(C, \prod_{i=1}^{n} \hat{g}_{n+1-i}^{t_i}\big) = e\big(\prod_{i=1}^{n} \pi_i^{t_i}, \hat{g}\big) \cdot e(g_1, \hat{g}_n)^{\sum_{i=1}^{n} m_i \cdot t_i} \quad . \tag{1}$$

PointProofs [61] aggregates proofs $\{\pi_i\}_{i \in S}$ for a sub-vector $S \subseteq [n]$ by deriving aggregation coefficients $\{t_i\}_{i \in S}$ from a random oracle and defining the aggregated proof as the product $\pi_S = \prod_{i \in S} \pi_i^{t_i}$. Verification then proceeds by testing the equality $e(C, \hat{g}_{n+1-i})^{\sum_{i \in S} t_i} = e(\pi_S, \hat{g}) \cdot e(g_1, \hat{g}_n)^{\sum_{i \in S} m_i \cdot t_i}$. In the following, we further exploit the aggregation properties of PointProofs.

Proving Binarity. Let a commitment $\hat{C} = \hat{g}^{\gamma} \cdot \prod_{i=1}^{n} \hat{g}_i^{x_i}$ to $\boldsymbol{x} \in \{0,1\}^n$.[1] Using its proof aggregation properties, we prove that, for each $i \in [n]$, we have $x_i \cdot (x_i - 1) = 0$. To this end, we use a similar batching technique to BulletProofs [19] and show that $\sum_{i=1}^{n} y_i \cdot x_i \cdot (x_i - 1) = 0$, where $\boldsymbol{y} = (y_1, \ldots, y_n) \in \mathbb{Z}_p^n$ is a vector of random aggregation coefficients obtained by hashing $\boldsymbol{y} = H(\hat{C})$ using a random oracle $H : \{0,1\}^* \to \mathbb{Z}_p^n$. As long as $\boldsymbol{y} \in \mathbb{Z}_p^n$ is chosen uniformly *after* $\{x_i\}_{i=1}^n$, the probability to have $\sum_{i=1}^{n} y_i \cdot x_i \cdot (x_i - 1) = 0$ is only $1/p$ if there exists $i \in [n]$ such that $x_i \notin \{0,1\}$.

In order to prove the statement using a constant number of group elements, we first choose $\gamma_y \xleftarrow{R} \mathbb{Z}_p$ and generate an auxiliary commitment

$$C_y = g^{\gamma_y} \cdot \prod_{j=1}^{n} g_{n+1-j}^{y_j \cdot x_j}, \tag{2}$$

to the Hadamard product $\boldsymbol{y} \circ \boldsymbol{x} = (y_1 \cdot x_1, \ldots, y_n \cdot x_n)$ (in the reversed order). Then, we proceed in two steps.

In a first step, our prover has to demonstrate that it really computed C_y as a commitment to $(y_n \cdot x_n, \ldots, y_1 \cdot x_1)$. Since the commitment (2) satisfies

$$e(C_y, \hat{g}_i) = e\big(g_i^{\gamma_y} \cdot \prod_{j=1, j \neq i}^{n} g_{n+1-j+i}^{y_j \cdot x_j}, \hat{g}\big) \cdot e(g_1, \hat{g}_n)^{y_i \cdot x_i} \qquad \forall i \in [n] \tag{3}$$

and the initial commitment $\hat{C} = \hat{g}^{\gamma} \cdot \prod_{j=1}^{n} \hat{g}_j^{x_j}$ satisfies

$$e(g_{n+1-i}, \hat{C}) = e\big(g_{n+1-i}^{\gamma} \cdot \prod_{j=1, j \neq i}^{n} g_{n+1-i+j}^{x_j}, \hat{g}\big) \cdot e(g_1, \hat{g}_n)^{x_i} \qquad \forall i \in [n], \tag{4}$$

[1] For our applications, we will assume that the commitment is in $\hat{\mathbb{G}}$ rather than $\mathbb{G}$ in order for the proof of knowledge-soundness to work out.

we can choose random exponents $\boldsymbol{t} = (t_1, \ldots, t_n) \xleftarrow{R} \mathbb{Z}_p^n$ and use them to raise (4) to the power $t_i \cdot y_i$ and (3) to the power t_i, respectively. If we then take the products over all indices $i \in [n]$ and divide them, we find that

$$\pi_{eq} = \frac{\prod_{i=1}^{n} \left(g_{n+1-i}^{\gamma} \cdot \prod_{j \in [n] \setminus \{i\}} g_{n+1-i+j}^{x_j} \right)^{t_i \cdot y_i}}{\prod_{i=1}^{n} \left(g_i^{\gamma_y} \cdot \prod_{j \in [n] \setminus \{i\}} g_{n+1-j+i}^{y_j \cdot x_j} \right)^{t_i}}.$$

satisfies

$$\frac{e(\prod_{i=1}^{n} g_{n+1-i}^{t_i \cdot y_i}, \hat{C})}{e(C_y, \prod_{i=1}^{n} \hat{g}_i^{t_i})} = e(\pi_{eq}, \hat{g}), \tag{5}$$

The reason why π_{eq} is a convincing proof that the prover computed C_y as a commitment to $(y_n \cdot x_n, \ldots, y_1 \cdot x_1)$ is the following. Suppose that C_y is a commitment $C_y = g^{\gamma_y} \cdot \prod_{j=1}^{n} g_{n+1-j}^{z_{n+1-j}}$ to some $(z_1, \ldots, z_n)$. Then, (3) becomes

$$e(C_y, \hat{g}_i) = e\big(\pi_{z,i}, \hat{g}\big) \cdot e(g_1, \hat{g}_n)^{z_{n+1-i}} \qquad \forall i \in [n], \tag{6}$$

where $\pi_{z,i} = \prod_{j=1, j \neq i}^{n} g_{n+1-j+i}^{z_{n+1-j}}$ is the proof that a prover can compute to open the $(n+1-i)$-th position of C_y. Now, if we raise (6) to the power t_i and divide it from (4) raised to the power $t_i \cdot y_i$, we obtain

$$\frac{e(\prod_{i=1}^{n} g_{n+1-i}^{t_i \cdot y_i}, \hat{C})}{e(C_y, \prod_{i=1}^{n} \hat{g}_i^{t_i})} = e\Big(\prod_{i=1}^{n} \left(\pi_{x,i}^{y_i} / \pi_{z,i} \right)^{t_i} , \hat{g} \Big) \cdot e(g_1, \hat{g}_n)^{\sum_{i=1}^{n} t_i \cdot (y_i \cdot x_i - z_{n+1-i})},$$

where $\pi_{x,i} = \prod_{j \in [n] \setminus \{i\}} g_{n+1-i+j}^{x_j}$ is the computable proof that allows opening the i-th position of $\hat{C}$ in (4). If $\boldsymbol{t}$ is chosen uniformly after $(z_1, \ldots, z_n)$, $(y_1, \ldots, y_n)$ and $(x_1, \ldots, x_n)$, then the probability to have $\sum_{i=1}^{n} t_i \cdot (y_i \cdot x_i - z_{n+1-i}) = 0$ is $1/p$ if there exists $i \in [n]$ such that $z_{n+1-i} \neq y_i \cdot x_i$. In the construction, we derive $\boldsymbol{t} = (t_1, \ldots, t_n) = H(\boldsymbol{y}, \hat{C}, C_y) \in \mathbb{Z}_p^n$ from a random oracle to make sure that $\boldsymbol{t}$ is computed after $\boldsymbol{y}$, $(x_1, \ldots, x_n)$ and $(z_1, \ldots, z_n)$.

The proof π_{eq} of the first step implies that $C_y \cdot \prod_{j=1}^{n} g_{n+1-j}^{-y_j}$ is a commitment to the vector $(y_n \cdot (x_n - 1), \ldots, y_1 \cdot (x_1 - 1))$, where $(x_1, \ldots, x_n)$ is the vector committed in $\hat{C}$. In a second step, we prove that $(y_n \cdot (x_n - 1), \ldots, y_1 \cdot (x_1 - 1))$ is orthogonal to $(x_n, \ldots, x_1)$: i.e., $\sum_{i=1}^{n} y_i \cdot x_i \cdot (x_i - 1) = 0$. From (1), it can be shown that such a proof is computable as

$$\pi_y = (C_y \cdot \prod_{j=1}^{n} g_{n+1-j}^{-y_j})^{\gamma} \cdot \prod_{i=1}^{n} \Big(g_i^{\gamma_y} \cdot \prod_{j \in [n] \setminus \{i\}} g_{n+1-j+i}^{y_j \cdot (x_j - 1)} \Big)^{x_i}$$

and satisfies

$$e\big(C_y \cdot \prod_{j=1}^{n} g_{n+1-j}^{-y_j}, \hat{C}\big) = e(\pi_y, \hat{g}) \cdot e(g_1, \hat{g}_n)^{\sum_{i=1}^{n} y_i \cdot x_i \cdot (x_i - 1)} = e(\pi_y, \hat{g}) \tag{7}$$

In order to minimize the proof size, we will exploit the linearity of verification equations (5) and (7) to aggregate π_{eq} and π_y into a single group element $\pi = \pi_{eq}^{\delta_{eq}} \cdot \pi_y^{\delta_y}$ using random aggregation coefficients $(\delta_{eq}, \delta_y) \in \mathbb{Z}_p^2$.

Eventually, the proof $\boldsymbol{\pi} = (C_y, \pi) \in \mathbb{G}^2$ that $\hat{C}$ commits to a binary vector consists of the commitment C_y to $(y_n \cdot x_n, \ldots, y_1 \cdot x_1)$ and $\pi \in \mathbb{G}$.

Proving Range Membership. To obtain a constant-size range proof, we use the fact that the commitment scheme of [81] is also an inner product functional commitment. The prover has a Pedersen commitment [90] $\hat{V} = \hat{g}^r \cdot \hat{g}_1^x$ in the group $\hat{\mathbb{G}}$. In order to prove the statement $x \in [0, 2^\ell - 1]$, the prover considers the bit representation $(x_1, \ldots, x_\ell) \in \{0,1\}^\ell$ of x and computes a commitment $\hat{C} = \hat{g}^\gamma \cdot \prod_{j=1}^{\ell} g_j^{x_j}$, for a random $\gamma \in \mathbb{Z}_p$. Using the aggregation properties of the commitment, it will prove that the committed $\boldsymbol{x} = (x_1, \ldots, x_\ell \mid \mathbf{0}^{n-\ell}) \in \mathbb{Z}_p^n$ satisfies: (i) $\sum_{i=1}^{\ell} x_i \cdot 2^{i-1} = x$; (ii) $x_i \in \{0,1\}$ for each $i \in [n]$.

In order to prove (i), the prover can adapt (1) and generate a short proof $\prod_{i=1}^{n} \pi_i^{2^{i-1}} \in \mathbb{G}$ such that

$$e\big(\prod_{i=1}^{\ell} g_{n+1-i}^{2^{i-1}}, \hat{C}\big) = e\big(\prod_{i=1}^{\ell} \pi_i^{2^{i-1}}, \hat{g}\big) \cdot e(g_1, \hat{g}_n)^{\sum_{i=1}^{\ell} x_i \cdot 2^{i-1}} \tag{8}$$

and show that the exponent above $e(g_1, \hat{g}_n)$ in (8) is equal to the committed x in $\hat{V} = \hat{g}^r \cdot \hat{g}_1^x$. Since $\hat{V}$ satisfies $e(g_n, \hat{V}) = e(g_1, \hat{g}_n)^x \cdot e(g_n^r, \hat{g})$, the prover can actually compute $\pi_x = \prod_{i=1}^{\ell} \pi_i^{2^{i-1}} / g_n^r$ such that

$$\frac{e\big(\prod_{i=1}^{\ell} g_{n+1-i}^{2^{i-1}}, \hat{C}\big)}{e(g_n, \hat{V})} = e\big(\pi_x, \hat{g}\big) \ . \tag{9}$$

Proving (ii) is addressed as explained earlier. Note that we do not need to prove that the $n - \ell$ last positions of $\boldsymbol{x}$ are zeroes since the inner product in the right-hand-side member of (8) only involves the first ℓ positions of $\boldsymbol{x}$.

In order to minimize the proof size, we will exploit the linearity of verification equations (9), (5) and (7) to aggregate π_x, π_{eq} and π_y into a single group element. In order to ensure knowledge soundness in the algebraic group model, we also need to aggregate a proof element π_v showing that $\hat{V}$ is a commitment to a vector of the form $(x, 0, \ldots, 0)$. The entire proof $\boldsymbol{\pi} = (\hat{C}, C_y, \pi) \in \hat{\mathbb{G}} \times \mathbb{G}^2$ eventually consists of the commitment $\hat{C}$ to the bits of x, the auxiliary commitment C_y to $(y_n \cdot x_n, \ldots, y_1 \cdot x_1)$ and the aggregated proof $\pi \in \mathbb{G}$.

Batching Range Proofs. The above technique extends to prove multiple range membership statements at once about the entries of a committed vector. For a commitment $\hat{V} = \hat{g}^r \cdot \prod_{k=1}^{m} \hat{g}_k^{x_k}$, the prover will convince the verifier that $x_k \in [0, 2^\ell - 1]$ for each $k \in [m]$ using only 3 group elements (we assume for now that the same range is proven for each x_k but distinct ranges can be handled).

To this end, we can use the same aggregation technique as BulletProofs [19, Section 4.3] and compute $\hat{C}$ as a commitment to a vector of dimension

$n = \bar{\ell} \cdot m$ (where $\bar{\ell}$ is an upper bound for ℓ) obtained by appending the binary expansions of all $\{x_k\}_{k=1}^m$. Then, we can use a single group element to prove that, for each $k \in [m]$, the k-th sub-vector $\boldsymbol{x}_k = (x_{k,1}, \ldots, x_{k,\ell}, 0, \ldots, 0)$ hidden by the commitment $\hat{C}$ is a binary vector satisfying $x_k = \sum_{i=1}^{\ell} x_{k,i} \cdot 2^{i-1}$.

Proving Relations in Lattices. Here, we build on an approach considered by del Pino, Lyubashevsky and Seiler [41] to prove lattice-related statements assuming the hardness of computing discrete logarithms. The difference that we replace the BulletProofs component [19] by our more compact proof that a committed vector is binary. We also exploit the fact that the underlying vector commitment [81] allows proving inner-product relations as in (1).

Let the polynomial rings $R = \mathbb{Z}[X]/(\Phi)$, for some cyclotomic polynomial Φ of degree d, and $R_q = R/(qR)$. As in [41], we aim at proving the existence of small-norm ring elements $\boldsymbol{s} = (s_1, \ldots, s_M) \in R^M$ such that $\sum_{i=1}^M \boldsymbol{a}_i \cdot s_i = \boldsymbol{t} \bmod (q, \Phi)$, for public $\boldsymbol{t} \in R_q^N$ and $\boldsymbol{a}_1, \ldots, \boldsymbol{a}_M \in R_q^N$. To this end, we proceed as in [41] and re-write the relation as the following equality over $\mathbb{Z}[X]/(\Phi)$

$$\sum_{i=1}^{M} \boldsymbol{a}_i \cdot s_i = \boldsymbol{t} + \boldsymbol{r} \cdot q \bmod (\Phi) \ , \tag{10}$$

where $\boldsymbol{r} \in R^N$ is a vector of polynomials of degree $\leq d-1$ and the components of $\{\boldsymbol{a}_i\}_{i=1}^M$ and $\boldsymbol{t}$ are interpreted as polynomials with integer coefficients in $\{-\lfloor q/2 \rfloor, \ldots, \lfloor q/2 \rfloor\}$. If $\|s_i\|_\infty \leq B_i$ for each $i \in [M]$, $\boldsymbol{r}$ contains polynomials with coefficients of magnitude $\|\boldsymbol{r}\|_\infty \leq dM \cdot \max_{i \in [M]}(B_i)/2$.

If we denote by $\phi : R \to \mathbb{Z}^d$ the coefficient embedding that maps $s_i = \sum_{j=1}^d s_{i,j} \cdot X^{j-1}$ to its coefficient vector $\phi(s_i) = (s_{i,1}, \ldots, s_{i,d}) \in \mathbb{Z}^d$, we can re-write (10) as a matrix-vector product over $\mathbb{Z}$

$$[\mathbf{A}_1 \mid \ldots \mid \mathbf{A}_M \mid \ -q \cdot \mathbf{I}_{Nd} \ ] \cdot \underbrace{[\phi(s_1) \mid \ldots \mid \phi(s_M) \mid \phi(\boldsymbol{r})]^\top}_{\triangleq \ \boldsymbol{x}} = \phi(\boldsymbol{t}) \tag{11}$$

for structured matrices $\mathbf{A}_1, \ldots, \mathbf{A}_M \in \mathbb{Z}_q^{Nd \times d}$ interpreted as integer matrices over $\{-\lfloor q/2 \rfloor, \ldots, \lfloor q/2 \rfloor\}$. In order to prove (11), the prover can commit to the vector $\boldsymbol{x} \in \mathbb{Z}^{Md+Nd}$ using a vector commitment. Then, it can generate short proof that $\|\phi(s_i)\|_\infty \leq B_i$ for each $i \in [M]$ and $\|\phi(\boldsymbol{r})\|_\infty \leq dM \cdot \max_{i \in [M]}(B_i)/2$. Finally, it can prove that (11) holds over $\mathbb{Z}_p$, where p is the order of pairing-friendly groups. If $p > 2Mqd \max_i(B_i)$, this ensures that (11) also holds over the integers. In order to optimize the proof size, we commit to the binary decomposition of $(\phi(s_1), \ldots, \phi(s_M), \phi(\boldsymbol{r}))$ and prove a relation that implies (11).

In order to minimize the number of exponentiations, we apply the Schwartz-Zippel lemma in a different way than [41]: Instead of proving (10) by considering evaluations of degree-$2d$ polynomials,[2] we compress (11) by left-multiplying both

[2] More precisely, [41] proceeds by proving a relation $\sum_{i=1}^M \boldsymbol{a}_i \cdot s_i - \boldsymbol{r}_1 \cdot q - \boldsymbol{r}_2 \cdot \Phi = \boldsymbol{t}$ over $\mathbb{Z}[X]$, where $\boldsymbol{r}_1$ and $\boldsymbol{r}_2$ contain polynomials of degree $2(d-1)$ and $d-2$, respectively.

members with a random vector $\boldsymbol{\theta} \in \mathbb{Z}_p^{Nd}$, which allows processing all the rows of (11) using a short proof for a single inner product relation.

Just like [41], our protocol does not preserve soundness against quantum adversaries. However, both protocols still provide viable solutions in applications that only need to guarantee soundness at the moment of the protocol execution (i.e., today and assuming that the adversary is not quantum). In particular, they do not affect the post-quantum security of the encryption scheme as their zero-knowledge property does not rely on any assumption.

Achieving Simulation-Extractability. In our security proofs, one of the main difficulties is to properly simulate proofs for adversarially-chosen statements while remaining able to extract a witness (or break some assumption) from a proof generated by the adversary. As noticed in, e.g. [51], the simulator cannot use the trapdoor $\alpha \in \mathbb{Z}_p$ of the CRS since it would be incompatible with a reduction from a q-type assumption in the AGM.

To address this problem, we build a trapdoor-less simulator [51] that can simulate proofs for adversarially-chosen statements by programming the random oracles and without using α. To do this, we exploit the fact that our range proofs and our proof of valid RLWE encryption are obtained by aggregating various sub-proofs satisfying verifications of the form (5), (7) or (8). In each simulated proof $\boldsymbol{\pi} = (\hat{C}, C_y, \pi)$, we compute $\hat{C}$ and C_y as commitments to vectors which are programmed (as functions of previously chosen aggregation coefficients) in such a way that the unique corresponding valid proof π is computable without knowing the missing element $g^{(\alpha^{n+1})}$ of the CRS. At the same time, we can argue that the adversary cannot fake a proof using the simulator's strategy. We show that, with overwhelming probability, it can only come up with a proof π whose representation depends on $g^{(\alpha^{n+1})}$ if knowledge extraction fails.

1.3 Related Work

Vector commitments with logarithmic-size proofs are known since the Merkle-tree-based construction [87]. In the last decade, a number of number-theoretic candidates have emerged and offered useful advantages such as additive homomorphism, very short proofs [81], stateless updatability [26], or sub-vector openings [78,9,95]. The first candidate with constant-size proofs was put forth by Libert and Yung [81] under a q-type assumption. Constructions based on the standard Diffie-Hellman assumption (in pairing-friendly groups) and the RSA assumption appeared in the work of Catalano and Fiore [26]. Lattice-based schemes were suggested by Peikert *et al.* [91]. While more versatile than their hash-based counterparts, algebraic VCs also seem to require more fancy mathematical tools. Indeed, Catalano *et al.* [27] recently proved negative results on the possibility of discrete-log-based vector commitments without pairings.

Polynomial and Functional Commitments. Polynomial commitments [74] allow one to commit to a polynomial and subsequently prove evaluations of this polynomial on specific inputs via a short proof (i.e., of length sub-linear in the

degree of the committed polynomial). Succinct polynomial commitments were used in a number of SNARKs realizations (see, e.g., [86,50,20,11]). As shown in, e.g. [23, Section 3.1], polynomial commitments imply vector commitments.

Functional commitments (FC) for inner products [71,80] generalize both vector commitments and polynomial commitments by allowing the sender to commit to a vector $\boldsymbol{m}$ and succinctly prove linear functions of the committed vector. The first flavor of inner product functional commitment was considered in the interactive setting [71] while non-interactive solutions with constant-size proofs are enabled by SNARKs. Libert, Ramanna and Yung [80] generalized the vector commitment of [81] into a non-interactive inner product FC in the standard model while preserving its short proof size. Constructions with short public parameters in hidden-order groups were put forth in [33,2]. Lai and Malavolta [78] proposed the notion of linear map commitments that allows a prover to reveal a linear map evaluation, instead of just an inner product. At the expense of losing the homomorphic property, Lipmaa and Pavlyk [83] provided an FC candidate for sparse polynomials. Boneh *et al.* [13] considered the dual notion of function-hiding FC schemes (where the committer commits to a function instead of a message) for arithmetic circuits, which generalizes vector commitments and other flavors of commitments.

Among lattice-based realizations, Gorbunov *et al.* [62] implicitly showed non-succinct functional commitments for bounded-depth circuits. Peikert *et al.* [91] proposed a succinct realization while relying on an online trusted authority to generate proofs. Albrecht *et al.* described [1] a construction for constant-degree polynomials over the integers as a building block for lattice-based SNARKs. Succinct FC candidates for circuits recently appeared in the work of Wee and Wu [97]. Independently, de Casto and Peikert [40] proposed a lattice-based function-hiding FC for circuits, but without fully succinct evaluation proofs.

Vector commitments with succinct proofs of smallness can be seen as a special case of functional commitments for Boolean predicates, where the smallness bound is hard-wired in the circuit. However, functional commitments for general circuits [40,97] seem ill-suited to our purposes since we aim at computationally efficient schemes with very short proofs. Indeed, the function-hiding FC scheme proposed by de Castro and Peikert [40] does not provide succinct openings (i.e., the opening size grows with the input length). While succinct, the construction of Wee and Wu [97] would not compete with ours in terms of proof length and CRS size (which is quadratic in the dimension of committed vectors in [97]). Moreover, in our application to NIZK arguments, the scheme of [97] would require the use of ad hoc knowledge assumptions in lattices for lack of a well-defined lattice analogue of the algebraic group model. Balbás *et al.* [5] suggested an alternative realization of FC for arithmetic circuits. However, its proof length grows at least linearly with the depth of the arithmetic circuit, which would translate into much longer proofs than ours.

In an earlier work, Catalano, Fiore and Tucker [28] proposed additively homomorphic FCs for constant-degree polynomials and monotone span programs. While their construction for polynomials and the Lipmaa-Pavlyk construction

[83] are both amenable to proving smallness statements, they would be less efficient than our constructions, as discussed in Supplementary Material A. Moreover, their more complex CRS structure would make it harder to prove knowledge-soundness in our setting, where the evaluation-binding property considered in [83,28] would not suffice.

Aggregation and Sub-vector Openings. On several occasions, we rely on sub-vector openings and proof aggregation in the vector commitment of [81].

The notion of sub-vector openings was independently introduced and realized by Lai and Malavolta [78] and by Boneh, Bünz and Fisch [9]. It allows a sender to generate a short proof π_S that opens a sub-vector $\boldsymbol{m}_S$ of $\boldsymbol{m}$, for a subset $S \subseteq [n]$. Sub-vector openings are implied by the proof aggregation property considered in [9,96,95,24,61,94], which allows anyone (and not only the committer) to aggregate n individual proofs $\{\pi_i\}_{i \in S}$ for a committed sub-vector $\boldsymbol{m}_S$ into a constant-size proof π_S. Boneh, Bünz and Fisch [9] and Tomescu *et al.* [95] realized same-commitment aggregation in hidden-order groups and under q-type assumptions in pairing-friendly groups, respectively. Campanelli *et al.* [24] introduced incrementally aggregatable vector commitments, which allow different sub-vector openings to be merged into a shorter opening for the union of their sub-vectors. Moreover, aggregated proofs support further aggregation.

By leveraging the linearity properties of the vector commitment from [81], Gorbunov *et al.* [61] obtained a VC scheme enabling cross-commitment aggregation, which is useful in blockchain applications. The same-commitment variant of their aggregation method is obtained by introducing a random oracle in the inner product functional commitment of [80]. Our technique of proving that a committed vector is a reversed Hadamard product of another committed vector $\boldsymbol{x}$ and a public vector $\boldsymbol{y}$ is inspired by the randomized aggregation technique of PointProofs [61]. The difference is that, while [61] uses proof aggregation to succinctly prove sub-vector openings, we use it to prove linear relations between related positions in distinct committed vectors.

Using aggregation techniques, Campanelli *et al.* [25] described a compiler building linear map commitments from inner product functional commitments.

By instantiating vector commitments from polynomial commitments, Boneh *et al.* [10,11] obtained an alternative VC system supporting cross-commitment aggregation. Hyperproofs *et al.* [94] is yet another VC scheme allowing cross-commitment aggregation with the additional feature that all proofs can be updated in sub-linear time when the vector changes.

Other Proofs of Binarity. Prior works on pairing-based commitments [59,58] considered the problem of constructing constant-size proofs that a committed string is binary. However, these techniques apply to variants of Groth-Sahai commitments [69] that are *not* succinct vector commitments: i.e., either the commitment or partial openings (or both) do not have constant size. The first candidate [59] was designed for perfectly-binding commitments, where the commitment is longer than the committed message. The case of perfectly hiding (compressing) commitments was considered in [58, Section 4.2] but the under-

lying commitments do not natively support constant-size partial openings. As briefly alluded to in [58, Section 4.2.1], it is actually possible to build a succinct vector commitment to bitstrings on top of the perfectly hiding commitments from [58, Chapter 4]. However, the resulting construction has several limitations: (i) The CRS has quadratic size in the dimension of committed vectors (like the Diffie-Hellman-based vector commitment of [26]); (ii) It does not seem to support constant-size proofs that the committed $\boldsymbol{m} \in \mathbb{Z}_p^n$ satisfies inner product relations $\langle \boldsymbol{m}, \boldsymbol{t} \rangle = x$ for public $\boldsymbol{t} \in \mathbb{Z}_p^n$ and $x \in \mathbb{Z}_p$; (iii) Proofs are somewhat long and contain more than 20 group elements (according to Table 4.1 in [58]).

Das *et al.* [39] recently proposed another constant-size argument showing that a committed vector is binary. While their construction can be modified to build an alternative range proof to ours, it would result in longer proofs.

Range Proofs. Range proofs were introduced by Brickell *et al.* [18] and investigated in a large body of work [30,22,15,82,66,29,36,60] since then.

A standard approach [18,22,66,60,19] consists in breaking integers into bits and committing to these bits using homomorphic commitments. When it comes to proving membership of a range $[0, 2^\ell - 1]$, the resulting proofs generally contain $O(\ell)$ group elements (and thus $O(\lambda \cdot \ell)$ bits, where λ is the security parameter) although somewhat shorter proofs [22,66,60] are achievable using pairings. Using a clever recursive folding technique, Bulletproofs [19] decreased the communication complexity to $O(\log \ell)$ group elements (i.e., $O(\lambda \cdot \log \ell)$ bits) in general discrete-logarithm-hard groups without a bilinear map.

Another approach [15,82,63,36] relies on integer commitments in hidden-order groups, by decomposing positive integers as a sum of squares. The sum-of-squares method was transposed [35,34] to groups of (sufficiently large) public prime order. It was also adapted to class groups and lattices. For some parameters in the standard discrete logarithm setting, the constructions of [35,34] were shown to compare favorably with BulletProofs.

For some applications where the proof size is the primary concern (e.g., confidential transactions in the blockchain [19]), it may be desirable to have even shorter proofs than [19,35,34], even at the expense of losing the transparent setup property. Using polynomial commitments, Boneh *et al.* [12] suggested another range proof inspired by SNARK arithmetization techniques [50]. Their construction can be realized from a variety of polynomial commitments [74,20,79]. In instantiations from pairing-based polynomial commitments [74,80], it provides the smallest communication cost to date, with proofs as short as 3 group elements and 3 scalars. In our range proof construction, we further decrease the proof length to that of the shortest known SNARKs [67]. A detailed comparison with [12] is given in Section 4.4.

Discrete-Log-Based Proofs for Lattice Relations. The use of specialized pairing-based arguments to prove lattice relations was considered to prove the correct evaluation of FHE ciphertexts [47]. However, the modulus of the leveled FHE scheme had to match the group order of the pairing. This limitation does not appear in the del Pino *et al.* approach [41] nor in our construction. We

note that the motivation of [47] was different since, in their setting, the prover was the server while the verifier was a computationally constrained client. Here, we consider use cases like [93] where the prover is the client (generating the proof on its browser using a single thread) and the verifier runs on a computationally powerful machine that can afford the use of multiple threads.

In applications to private FHE-based private smart contracts [93], the protocol of [41] was actually preferred to SNARKs in order to obtain faster prover. Our system can offer a similarly fast prover with the benefit of shorter proofs.

1.4 Organization

We first present our argument of binarity in Section 3. Our constant-size range proof is described in Section 4. Its batched multi-range extension is detailed in Supplementary Material D. Due to space limitation, our NIZK argument for ring LWE ciphertexts is deferred to Supplementary Material G.

2 Background and Definitions

2.1 Hardness Assumptions

Let groups $(\mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T)$ of prime order p with a bilinear map $e : \mathbb{G} \times \hat{\mathbb{G}} \rightarrow \mathbb{G}_T$.

We rely on the hardness of computing a discrete logarithm $\alpha \in \mathbb{Z}_p$ given $\{g^{\alpha^i}\}_{i\in[2n]}$ and $\{\hat{g}^{\alpha^i}\}_{i\in[n]}$. This assumption is similar to the n-discrete logarithm assumption considered in, e.g. [48], except that powers α^i are given in the exponents in both groups $\mathbb{G}$ and $\hat{\mathbb{G}}$.

Definition 1 ([48]). *Let $(\mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T)$ be asymmetric bilinear groups of prime order p. For integers m, n, the* (m, n)**-Discrete Logarithm** *$((m, n)$-DLOG) problem is, given $(g, g^{\alpha}, g^{(\alpha^2)}, \ldots, g^{(\alpha^m)}, \hat{g}, \hat{g}^{\alpha}, \ldots, \hat{g}^{(\alpha^n)})$, where $\alpha \xleftarrow{R} \mathbb{Z}_p$, $g \xleftarrow{R} \mathbb{G}$, $\hat{g} \xleftarrow{R} \hat{\mathbb{G}}$, to compute $\alpha \in \mathbb{Z}_p$.*

2.2 Non-interactive Arguments

Let $\{\mathcal{R}_\lambda\}_\lambda$ a family of NP relations. A NIZK argument for $\{\mathcal{R}_\lambda\}_\lambda$ consists of algorithms $\Pi = (\mathsf{CRS\text{-}Gen}, \mathsf{Prove}, \mathsf{Verify})$ with the following specifications. On input of a security parameter $\lambda \in \mathbb{N}$ and, optionally, language-dependent parameters lpp, algorithm $\mathsf{CRS\text{-}Gen}$ generates a common reference string pp and a simulation trapdoor τ. We allow pp to parameterize the proven relation (when it specifies the public parameters of a commitment scheme), which then becomes $\mathcal{R}_{\mathsf{pp}} \in \{\mathcal{R}_\lambda\}_\lambda$. Algorithm Prove takes as input the common reference string pp, a statement x and a witness w and outputs a proof π when $(x, w) \in \mathcal{R}_{\mathsf{pp}}$. Verify takes in pp, a statement x and a proof π and returns 0 or 1. Correctness requires that, for any $\mathcal{R} \in \{\mathcal{R}_\lambda\}_\lambda$ and any $(x, w) \in \mathcal{R}_{\mathsf{pp}}$, honestly generated proofs are always (or at least with overwhelming probability) accepted by the verifier.

NIZK arguments should satisfy two security properties. The *zero-knowledge* property requires that proofs leak no information about the witness. *Knowledge-soundness* property requires that there exists an extractor that can compute a witness whenever the adversary generates a valid proof. The extractor has access to the adversary's internal state, including its random coins. Let the universal relation $\mathcal{R}^*$ for $\{\mathcal{R}_\lambda\}_\lambda$ that inputs $(\mathcal{R}_{\mathsf{pp}}, x, w)$ and outputs 1 iff $\mathcal{R}_{\mathsf{pp}} \in \{\mathcal{R}_\lambda\}_\lambda$ and $(x, w) \in \mathcal{R}_{\mathsf{pp}}$. We say that $\Pi = (\mathsf{CRS\text{-}Gen}, \mathsf{Prove}, \mathsf{Verify})$ is a NIZK argument for $\mathcal{R}^*$ if it satisfies the properties defined as follows.

Completeness: For any $\lambda \in \mathbb{N}$, any (not necessarily efficient) adversary $\mathcal{A}$, there is a negligible function $\mathsf{negl} : \mathbb{N} \to \mathbb{N}$ such that

$$\Pr\left[\mathsf{Verify}_{\mathsf{pp}}(x,\pi) = 1 \ \wedge \ (x,w) \in \mathcal{R}_{\mathsf{pp}} \ \mid \ (\mathsf{pp},\tau) \leftarrow \mathsf{CRS\text{-}Gen}(1^\lambda, \mathsf{lpp}), \ (x,w) \leftarrow \mathcal{A}(\mathsf{pp}) \ , \ \pi \leftarrow \mathsf{Prove}_{\mathsf{pp}}(x,w)\right] = 1 - \mathsf{negl}(\lambda).$$

Knowledge-soundness: For any PPT adversary $\mathcal{A}$, there is a PPT extractor $\mathcal{E}_{\mathcal{A}}$ that has access to $\mathcal{A}$'s internal state and random coins ρ such that

$$\Pr\left[\mathsf{Verify}_{\mathsf{pp}}(x,\pi) = 1 \ \wedge \ (x,w) \notin \mathcal{R}_{\mathsf{pp}} \ \mid \ (\mathsf{pp},\tau) \leftarrow \mathsf{CRS\text{-}Gen}(1^\lambda, \mathsf{lpp}), \ (x,\pi) \leftarrow \mathcal{A}(\mathsf{pp};\rho), \ w \leftarrow \mathcal{E}_{\mathcal{A}}(\mathsf{pp},(x,\pi),\rho)\right] = \mathsf{negl}(\lambda).$$

(Statistical) Zero-knowledge: There is a PPT simulator Sim such that, for any $\lambda \in \mathbb{N}$ and any (not necessarily efficient) adversary $\mathcal{A}$ and any $b \in \{0,1\}$,

$$\Pr\left[b \leftarrow \mathcal{A}^{\mathcal{O}_b}(\mathsf{pp}) \mid (\mathsf{pp},\tau) \leftarrow \mathsf{CRS\text{-}Gen}(1^\lambda, \mathsf{lpp})\right] = 1/2 + \mathsf{negl}(\lambda).$$

where $\mathcal{O}_1$ is an oracle that inputs (x, w) and returns $\pi \leftarrow \mathsf{Prove}_{\mathsf{pp}}(x, w)$ if $(x, w) \in \mathcal{R}_{\mathsf{pp}}$ and $\perp$ otherwise; $\mathcal{O}_0$ is oracle that inputs (x, w) and returns $\pi \leftarrow \mathsf{Sim}(\mathsf{pp}, \tau, x)$ if $(x, w) \in \mathcal{R}_{\mathsf{pp}}$ and $\perp$ otherwise.

For many applications, it is desirable to strengthen knowledge-soundness by considering an adversary that can observe simulated proofs (for possibly false statements) and exploit some malleability of these proofs to generate a fake proof of its own. The notion of simulation-extractability prevent such attacks.

Simulation-Extractability: For any PPT adversary $\mathcal{A}$, there is a PPT extractor $\mathcal{E}_{\mathcal{A}}$ that has access to $\mathcal{A}$'s internal state/randomness ρ such that

$$\Pr\left[\mathsf{Verify}_{\mathsf{pp}}(x,\pi) = 1 \ \wedge \ (x,w) \notin \mathcal{R}_{\mathsf{pp}} \ \wedge \ (x,\pi) \notin Q \ \mid \ (\mathsf{pp},\tau) \leftarrow \mathsf{CRS\text{-}Gen}(1^\lambda, \mathsf{lpp}), \ (x,\pi) \leftarrow \mathcal{A}^{\mathsf{SimProve}}(\mathsf{pp};\rho), \ w \leftarrow \mathcal{E}_{\mathcal{A}}(\mathsf{pp},(x,\pi),\rho,Q)\right] = \mathsf{negl}(\lambda),$$

where $\mathsf{SimProve}(\mathsf{pp}, \tau, \cdot)$ is an oracle that returns a simulated proof $\pi \leftarrow \mathsf{Sim}(\mathsf{pp}, \tau, x)$ for a given statement x and $Q = \{(x_i, \pi_i)\}_i$ denotes the set of queried statements and the simulated proofs returned by $\mathsf{SimProve}$.

In the following sections, we extend the syntax with an algorithm Com that inputs a vector $\boldsymbol{x} \in D^n$ over a domain D and outputs a commitment C. This commitment will be incorporated in the specific relation $\mathcal{R}_{\mathsf{pp}}$ defined by $\mathsf{CRS\text{-}Gen}$.

2.3 Algebraic Group Model

The algebraic group model (AGM) [48] is an idealized model, where all algorithms are assumed to be algebraic. Algebraic algorithms [14,89] generalize the notion of a generic algorithm [92] in that, whenever they compute a group element, they do it using generic operations, by taking linear combinations of available group elements so far. Hence, whenever they output a group element $X \in \mathbb{G}$, they also output a representation $\{\alpha_i\}_{i=1}^N$ of $X = \prod_{i=1}^N g_i^{\alpha_i}$ as a function of previously observed group elements $(g_1, \ldots, g_N) \in \mathbb{G}^N$ in the same group.

In contrast with generic algorithms, algebraic algorithms can exploit the structure of the group and obtain more information than they would in the generic group model. Although its relation with the generic group model is unclear [75], the AGM provides a powerful framework to analyze the security of efficient protocols via reductions. In particular, it has been widely used in the context of SNARKs [48,86,50,11,53,51].

3 Short Proofs That a Committed Vector is Binary

Our construction for binary strings is defined for the relation

$$\mathcal{R}_{\mathsf{pp}} = \left\{ (\mathbf{x}, \mathbf{w}) = \left(\hat{V} = \hat{g}^{\gamma} \cdot \prod_{i=1}^{n} \hat{g}_1^{x_i} \in \hat{\mathbb{G}}, \quad \left(\gamma, (x_1, \ldots, x_n)\right) \in \mathbb{Z}_p \times \{0,1\}^n \right) \right\}$$

where pp denotes the CRS containing the commitment key $(\hat{g}, \{\hat{g}_i\}_{i=1}^n)$ and the description of groups $(\mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T)$. Since the commitment is perfectly-hiding, the proven relation is trivially satisfied because, for any group element $\hat{V}$, there exists a string $\boldsymbol{x} \in \{0,1\}^n$ and a corresponding $\gamma \in \mathbb{Z}_p$ such that $\hat{V} = \hat{g}^{\gamma} \cdot \prod_{i=1}^n \hat{g}_i^{x_i}$. However, we can prove that the scheme is an argument of knowledge.

$\mathsf{CRS\text{-}Gen}(1^\lambda, 1^n)$: On input of a security parameter λ and the maximal dimension $n \in \mathsf{poly}(\lambda)$ of committed vectors, do the following:

1. Choose asymmetric bilinear groups $(\mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T)$ of prime order $p > 2^{l(\lambda)}$, for some function $l : \mathbb{N} \to \mathbb{N}$, and $g \xleftarrow{R} \mathbb{G}$, $\hat{g} \xleftarrow{R} \hat{\mathbb{G}}$.
2. Pick $\alpha \xleftarrow{R} \mathbb{Z}_p$. Compute $g_1, \ldots, g_n, g_{n+2}, \ldots, g_{2n} \in \mathbb{G}$ and $\hat{g}_1, \ldots, \hat{g}_n \in \hat{\mathbb{G}}$, where $g_i = g^{(\alpha^i)}$ for each $i \in [2n] \setminus \{n+1\}$ and $\hat{g}_i = \hat{g}^{(\alpha^i)}$ for each $i \in [n]$.
3. Choose hash functions $H, H_t : \{0,1\}^* \to \mathbb{Z}_p^n$ and $H_{\mathrm{agg}} : \{0,1\}^* \to \mathbb{Z}_p^2$.

The public parameters are

$$\mathsf{pp} = \left((\mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T), g, \hat{g}, \{g_i\}_{i \in [2n] \setminus \{n+1\}}, \{\hat{g}_i\}_{i \in [n]}, \mathbf{H} = \{H, H_t, H_{\mathrm{agg}}\} \right).$$

$\mathsf{Com}_{\mathsf{pp}}(\boldsymbol{x})$: To commit to a vector $\boldsymbol{x} = (x_1, \ldots, x_n) \in \mathbb{Z}_p^n$, choose a random $\gamma \xleftarrow{R} \mathbb{Z}_p$ and compute $\hat{C} = \hat{g}^{\gamma} \cdot \prod_{j=1}^n \hat{g}_j^{x_j}$. Return $\hat{C} \in \hat{\mathbb{G}}$ and the opening information $\mathsf{aux} = \gamma \in \mathbb{Z}_p$.

$\mathsf{Prove}_{\mathsf{pp}}\big(\hat{C}, (\boldsymbol{x}, \mathsf{aux})\big)$: given a commitment $\hat{C}$ and witnesses $\big(\boldsymbol{x}; \mathsf{aux}\big)$ consisting of a vector $\boldsymbol{x} = (x_1, \ldots, x_n) \in \mathbb{Z}_p^n$ and randomness $\mathsf{aux} = \gamma \in \mathbb{Z}_p$, return $\perp$ if $(x_1, \ldots, x_n) \notin \{0,1\}^n$. Otherwise, do the following:

1. Compute $\boldsymbol{y} = (y_1, \ldots, y_n) = H(\hat{C}) \in \mathbb{Z}_p^n$. Choose $\gamma_y \xleftarrow{R} \mathbb{Z}_p$ and compute

$$C_y = g^{\gamma_y} \cdot \prod_{j=1}^{n} g_{n+1-j}^{y_j \cdot x_j}$$

   Then, compute $\boldsymbol{t} = (t_1, \ldots, t_n) = H_t(\boldsymbol{y}, \hat{C}, C_y) \in \mathbb{Z}_p^n$.
2. Generate a proof

$$\pi_{eq} = \frac{\prod_{i=1}^{n} \Big(g_{n+1-i}^{\gamma} \cdot \prod_{j \in [n] \setminus \{i\}} g_{n+1-i+j}^{x_j} \Big)^{t_i \cdot y_i}}{\prod_{i=1}^{n} \Big(g_i^{\gamma_y} \cdot \prod_{j \in [n] \setminus \{i\}} g_{n+1-j+i}^{y_j \cdot x_j} \Big)^{t_i}} \tag{12}$$

   which satisfies

$$\frac{e(\prod_{i=1}^{n} g_{n+1-i}^{t_i \cdot y_i}, \hat{C})}{e(C_y, \prod_{i=1}^{n} \hat{g}_i^{t_i})} = e(\pi_{eq}, \hat{g}) \ , \tag{13}$$

   and argues that C_y commits to $(y_n \cdot x_n, \ldots, y_1 \cdot x_1) \in \mathbb{Z}_p^n$.
3. Compute a proof

$$\pi_y = g^{\gamma \cdot \gamma_y} \cdot \prod_{j=1}^{n} g_{n+1-j}^{\gamma \cdot y_j \cdot (x_j - 1)} \cdot \prod_{i=1}^{n} \big(g_i^{\gamma_y} \cdot \prod_{j \in [n] \setminus \{i\}} g_{n+1-j+i}^{y_j \cdot (x_j - 1)} \big)^{x_i} \tag{14}$$

   showing that $\sum_{i=1}^{n} y_i \cdot x_i \cdot (x_i - 1) = 0$ and satisfying

$$e\big(C_y \cdot \prod_{j=1}^{n} g_{n+1-j}^{-y_j}, \hat{C}\big) = e(\pi_y, \hat{g}) \tag{15}$$

4. Compute $(\delta_{eq}, \delta_y) = H_{\mathrm{agg}}(\hat{C}, C_y) \in \mathbb{Z}_p^2$ and then $\pi = \pi_{eq}^{\delta_{eq}} \cdot \pi_y^{\delta_y}$.

Output the final proof $\boldsymbol{\pi} := \big(C_y, \pi\big) \in \mathbb{G}^2$.

$\mathsf{Verify}_{\mathsf{pp}}\big(\hat{C}, \boldsymbol{\pi}\big)$: Given $\hat{C} \in \hat{\mathbb{G}}$ and a purported proof $\boldsymbol{\pi} = \big(C_y, \pi\big) \in \mathbb{G}^2$,

1. Compute $\boldsymbol{y} = H(\hat{C}) \in \mathbb{Z}_p^n$, $(\delta_{eq}, \delta_y) = H_{\mathrm{agg}}(\hat{C}, C_y) \in \mathbb{Z}_p^2$ and $\boldsymbol{t} = H_t(\boldsymbol{y}, \hat{C}, C_y) \in \mathbb{Z}_p^n$.
2. Return 1 if the following equations is satisfied and 0 otherwise:

$$\frac{e\big(C_y^{\delta_y} \cdot \prod_{i=1}^{n} g_{n+1-i}^{(\delta_{eq} \cdot t_i - \delta_y) \cdot y_i}, \hat{C}\big)}{e\big(C_y, \prod_{i=1}^{n} \hat{g}_i^{\delta_{eq} \cdot t_i}\big)} = e(\pi, \hat{g}). \tag{16}$$

Correctness follows from the observation that equation (16) is obtained by aggregating (13)-(15), for which a detailed proof of correctness can be found in Supplementary Material C.1.

In the algebraic group model, the construction can be proven zero-knowledge and knowledge-sound (the zero-knowledge simulator actually needs an algebraic representation of the adversarially-chosen commitment $\hat{C}$ but this requirement can be removed by swapping the groups where $\hat{C}$ and C_y live). The proof of knowledge-soundness can be inferred from the proof of Theorem 2 (in Section 4), of which it is a sub-case. In the upcoming sections, we will combine the system with other components in such a way that the combined arguments satisfy the stronger notion of simulation-extractability.

In Supplementary Material B, we provide a detailed comparison with the construction of Das *et al.* [39] and show that our scheme yields more compact range proofs. In Supplementary Material F, we also explain how to prove the exact Hamming weight (or an upper bound thereof) of committed binary/ternary vectors using 4 group elements.

4 A Range Proof With Very Short Proofs

Using the non-interactive argument for binary vectors from Section 3, we can build range arguments made of a constant number of group elements.

In the description below, we assume ranges $[0, B]$ such that $B+1$ is a power of 2 but the approach easily extends to general ranges. The standard approach to this problem is to consider the integer $\ell \in \mathbb{N}$ such that $2^{\ell-1} \leq B < 2^{\ell}$ and generate two range proofs showing that $x \in [0, 2^{\ell}-1]$ and $x+(2^{\ell}-1-B) \in [0, 2^{\ell}-1]$, where the second part is proven by leveraging the additive homomorphic property of the commitment. Instead of generating two independent range proofs, we can double the size of the CRS (by setting $n = 2\bar{\ell}$, where $\bar{\ell} \geq \ell$ is the maximal bitlength of the range) and avoid increasing the proof size. In Supplementary Material D.4, we provide more details on the treatment of general ranges.

4.1 Description

We assume that the initial Pedersen commitment $\hat{V} = \hat{g}^r \cdot \hat{g}_1^x$ to the witness $x \in [0, 2^{\ell}-1]$ lives in the second source group $\hat{\mathbb{G}}$ of the pairing.[3]

The range membership relation is formally defined as

$$\mathcal{R}_{\mathsf{pp}} = \Big\{ (\mathbf{x}, \mathbf{w}) = \big((\hat{V} = \hat{g}^r \cdot \hat{g}_1^x, \ell) \in \hat{\mathbb{G}} \times \mathbb{N},\ (r, x) \in \mathbb{Z}_p \times [0, 2^{\ell}-1] \big) \Big\}$$

where the CRS pp specifies the commitment key $(\hat{g}, g_1)$ and the groups $(\mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T)$.

CRS-Gen$(1^{\lambda}, 1^n)$**:** On input of a security parameter λ and the maximal bitlength $n \in \mathsf{poly}(\lambda)$ of ranges, do the following:

[3] Committing x to a different, pairing-free, group $\mathbb{G}$ would not strengthen security since an adversary that would be able to compute α from pp would still break knowledge-soundness. The construction of [12] similarly assumes that the integer x is committed as a polynomial $f[X]$ such that $f(1) = x$.

1. Choose asymmetric bilinear groups $(\mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T)$ of prime order $p > 2^{l(\lambda)}$, for some polynomial function $l : \mathbb{N} \to \mathbb{N}$, and $g \xleftarrow{R} \mathbb{G}$, $\hat{g} \xleftarrow{R} \hat{\mathbb{G}}$.
2. Pick a random $\alpha \xleftarrow{R} \mathbb{Z}_p$ and compute $g_1, \ldots, g_n, g_{n+2}, \ldots, g_{2n} \in \mathbb{G}$ as well as $\hat{g}_1, \ldots, \hat{g}_n \in \hat{\mathbb{G}}$, where $g_i = g^{(\alpha^i)}$ for each $i \in [2n] \setminus \{n+1\}$ and $\hat{g}_i = \hat{g}^{(\alpha^i)}$ for each $i \in [n]$.
3. Choose hash functions $H, H_t : \{0,1\}^* \to \mathbb{Z}_p^n$, $H_s : \{0,1\}^* \to \mathbb{Z}_p$ and $H_{\text{agg}} : \{0,1\}^* \to \mathbb{Z}_p^4$ that will be modeled as random oracles.

The public parameters are defined to be

$$\mathsf{pp} = \Big((\mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T), g, \hat{g}, \{g_i\}_{i \in [2n]\setminus\{n+1\}}, \{\hat{g}_i\}_{i \in [n]}, \mathbf{H} = \{H, H_s, H_t, H_{\text{agg}}\}\Big)$$

$\mathsf{Com}_{\mathsf{pp}}(x)$: To commit to an integer $x \in \mathbb{Z}$, choose a random $r \xleftarrow{R} \mathbb{Z}_p$ and compute a Pedersen commitment $\hat{V} = \hat{g}^r \cdot \hat{g}_1^x \in \hat{\mathbb{G}}$. Return $\mathsf{com} = \hat{V} \in \hat{\mathbb{G}}$ and the opening information $\mathsf{aux} = r \in \mathbb{Z}_p$.

$\mathsf{Prove}_{\mathsf{pp}}\big(\mathsf{com}, (x, \mathsf{aux})\big)$: given $\mathsf{com} = \hat{V}$ and witnesses $\big(x; \mathsf{aux}\big)$ consisting of an integer $x \in [0, 2^\ell - 1]$ with binary expansion $(x_1, \ldots, x_\ell) \in \{0,1\}^\ell$, where $\ell \leq n$, and $\mathsf{aux} = r \in \mathbb{Z}_p$ such that $\hat{V} = \hat{g}^r \cdot \hat{g}_1^x$, do the following:

1. Set $(x_{\ell+1}, \ldots, x_n) = \mathbf{0}^{n-\ell}$. Choose $\gamma \xleftarrow{R} \mathbb{Z}_p$ and compute

$$\hat{C} = \hat{g}^\gamma \cdot \prod_{j=1}^{\ell} \hat{g}_j^{x_j}$$

   together with a proof $\pi_x \in \mathbb{G}$ that $\hat{C}$ commits to $(x_1, \ldots, x_n) \in \mathbb{Z}_p^n$ such that $\sum_{i=1}^{\ell} x_i \cdot 2^{i-1} = x$. This proof π_x satisfies

$$\frac{e(\prod_{i=1}^{\ell} g_{n+1-i}^{2^{i-1}}, \hat{C})}{e(g_n, \hat{V})} = e(\pi_x, \hat{g}) \tag{17}$$

   and is obtained as

$$\pi_x = g_n^{-r} \cdot \prod_{i=1}^{\ell} \Big(g_{n+1-i}^{\gamma} \cdot \prod_{j \in [\ell]\setminus\{i\}} g_{n+1-i+j}^{x_j}\Big)^{2^{i-1}}.$$

2. Compute $\boldsymbol{y} = (y_1, \ldots, y_n) = H(\hat{V}, \hat{C}) \in \mathbb{Z}_p^n$. Pick $\gamma_y \xleftarrow{R} \mathbb{Z}_p$ and compute

$$C_y = g^{\gamma_y} \cdot \prod_{j=1}^{\ell} g_{n+1-j}^{y_j \cdot x_j}$$

   Then, compute $\boldsymbol{t} = (t_1, \ldots, t_n) = H_t(\boldsymbol{y}, \hat{V}, \hat{C}, C_y) \in \mathbb{Z}_p^n$.
3. Prove that C_y commits to $(y_1 \cdot x_1, \ldots, y_n \cdot x_n) \in \mathbb{Z}_p^n$ by computing a short $\pi_{eq} \in \mathbb{G}$ (as specified in (12)) satisfying

$$\frac{e(\prod_{i=1}^{n} g_{n+1-i}^{t_i \cdot y_i}, \hat{C})}{e(C_y, \prod_{i=1}^{n} \hat{g}_i^{t_i})} = e(\pi_{eq}, \hat{g}). \tag{18}$$

4. Prove that $\sum_{i=1}^{n} y_i \cdot x_i \cdot (x_i - 1) = 0$ by computing $\pi_y \in \mathbb{G}$ via (14), which satisfies

$$e\big(C_y \cdot \prod_{j=1}^{n} g_{n+1-j}^{-y_j}, \hat{C}\big) = e(\pi_y, \hat{g}) \tag{19}$$

5. Generate an aggregated proof that $\hat{V} = \hat{g}^r \cdot \hat{g}_1^x$ is a commitment to a vector that contains 0 in its last $n-1$ coordinates. Namely, compute $\pi_v = \prod_{i=2}^{n} \left(g_{n+1-i}^r \cdot g_{n+2-i}^x\right)^{s_i} \in \mathbb{G}$ such that

$$e\big(\prod_{i=2}^{n} g_{n+1-i}^{s_i}, \hat{V}\big) = e(\pi_v, \hat{g}). \tag{20}$$

   where $s_i = H_s(i, [2,n], \hat{V}, \hat{C}, C_y) \in \mathbb{Z}_p$ for each $i \in [2,n]$.
6. Compute $(\delta_x, \delta_{eq}, \delta_y, \delta_v) = H_{\mathrm{agg}}(\hat{V}, \hat{C}, C_y) \in \mathbb{Z}_p^4$ and an aggregated proof

$$\pi = \pi_x^{\delta_x} \cdot \pi_y^{\delta_y} \cdot \pi_{eq}^{\delta_{eq}} \cdot \pi_v^{\delta_v}.$$

Output the final range argument which consists of

$$\boldsymbol{\pi} := \big(\hat{C}, C_y, \pi\big). \tag{21}$$

$\mathsf{Verify}_{\mathsf{pp}}\big(\mathsf{com}, \boldsymbol{\pi}\big)$: Given $\mathsf{com} = \hat{V} \in \hat{\mathbb{G}}$ and a purported proof $\boldsymbol{\pi} = \big(\hat{C}, C_y, \pi\big)$,

1. Compute $\boldsymbol{y} = H(\hat{V}, \hat{C}) \in \mathbb{Z}_p^n$, $(\delta_x, \delta_{eq}, \delta_y, \delta_v) = H_{\mathrm{agg}}(\hat{V}, \hat{C}, C_y) \in \mathbb{Z}_p^4$, $\boldsymbol{t} = H_t(\boldsymbol{y}, \hat{V}, \hat{C}, C_y) \in \mathbb{Z}_p^n$. Set $s_1 = 0$ and $s_i = H_s(i, [2,n], \hat{V}, \hat{C}, C_y)$ for all indices $i \in [2,n]$.
2. Return 1 if and only if

$$\frac{e\big(C_y^{\delta_y} \cdot \prod_{i=1}^{n} g_{n+1-i}^{\delta_{x,i} \cdot 2^{i-1} + (\delta_{eq} \cdot t_i - \delta_y) \cdot y_i}, \hat{C}\big)}{e\big(g_n^{\delta_x} \cdot \prod_{i=2}^{n} g_{n+1-i}^{-\delta_v \cdot s_i}, \hat{V}\big) \cdot e\big(C_y, \prod_{i=1}^{n} \hat{g}_i^{\delta_{eq} \cdot t_i}\big)} = e(\pi, \hat{g}), \tag{22}$$

   where $\delta_{x,i} = \delta_x$ if $i \in [\ell]$ and $\delta_{x,i} = 0$ if $i \in [\ell+1, n]$.

Correctness. The verification equation (22) is obtained by raising equalities (17), (18), (19) and (20) to the powers δ_x, δ_{eq}, δ_y, and δ_v, respectively, and multiplying the results together. In Supplementary Material C.1, we provide detailed proofs of correctness for individual verification equations (17)-(20).

Efficiency. The cost of the prover is dominated by $3n$ exponentiations in $\mathbb{G}$ and two exponentiations in $\hat{\mathbb{G}}$. Indeed, computing $\hat{C}$ at step 1 only requires one exponentiation and a subset product (which is cheaper than an exponentiation) in $\hat{\mathbb{G}}$. Step 2 requires $n+1$ exponentiations in $\mathbb{G}$. Instead of individually computing the proof elements $(\pi_x, \pi_{eq}, \pi_y, \pi_v)$, the prover can directly compute the entire product π at step 6 using only $2n$ exponentiations since the aggregation

coefficients $(\delta_x, \delta_{eq}, \delta_y, \delta_v)$, $\boldsymbol{y}$ and $\boldsymbol{t}$ only depend on the commitments $(\hat{V}, \hat{C}, C_y)$. This allows the prover to obtain the coefficients allowing to compute π from $\{g_i\}_{i\neq n+1}$ via 3 polynomial products (which are implicitly computed in the exponent by the pairings in the left-hand-side member of (22)) by interpreting each commitment as a polynomial evaluated in α in the exponent. Overall, the prover's overhead amounts to $3n$ exponentiations in $\mathbb{G}$, 2 exponentiations in $\hat{\mathbb{G}}$, and $O(n \log n)$ multiplications over $\mathbb{Z}_p$. The verifier's work is dominated by $2n+1$ exponentiations in $\mathbb{G}$, n exponentiations in $\hat{\mathbb{G}}$ and 4 pairings.

In terms of proof length, $\boldsymbol{\pi}$ only requires one element of $\hat{\mathbb{G}}$, and 2 element of $\mathbb{G}$, which matches the optimal size of simulation-extractable pairing-based SNARKs [68]. Using the KSS18 family of pairing-friendly curves suggested by Kachisa *et al.* [73], each element of $\mathbb{G}$ (resp. $\hat{\mathbb{G}}$) can have a 348-bit (resp. 1044-bit) representation at the 128-bit security level according to [44]. Assuming that elements of $\hat{\mathbb{G}}$ are three times as large as those of $\mathbb{G}$, the overall proof length does not exceed the equivalent of 5 elements of $\mathbb{G}$, which amounts to 1740 bits.

In Section 4.4, we give a detailed comparison among existing constant-size range proofs. As shown in Table 1, our scheme provides the shortest proof length and the smallest computational cost at the prover.

4.2 Security in the AGM & ROM

We first prove the zero-knowledge property in the random oracle model.

Theorem 1. *The construction provides statistical zero-knowledge in the ROM.* (The proof is given in Supplementary Material C.2.)

The simulator in the proof of Theorem 1 proceeds by programming the random oracles and also uses the trapdoor of the CRS. On the other hand, it works for any given $\hat{V} \in \hat{\mathbb{G}}$ without knowing an algebraic representation of $\hat{V}$. If we restrict $\hat{V}$ to be chosen by an algebraic adversary, it is possible to build an algebraic simulator that does not rely on random oracles.

In the proof of simulation-extractability, we need to build a trapdoor-less simulator, which does not use the trapdoor α of the common reference string.

Theorem 2. *Under the $(2n, n)$-DLOG assumption, the scheme is simulation-extractable in the algebraic group model and in the random oracle model.*

Proof. In the AGM+ROM model, we show that, unless the $(2n, n)$-DLOG assumption is false, there exists an extractor that can extract a witness from any adversarially-generated proof $\boldsymbol{\pi} = \big(\hat{C}, C_y, \pi\big)$ and statement $(\hat{V}, [0, 2^\ell - 1])$. Specifically, we give an algorithm $\mathcal{B}$ that can either extract a witness (x, r) with $x \in [0, 2^\ell - 1]$ or solve an $(2n, n)$-DLOG instance by computing $\alpha \in \mathbb{Z}_p$ from $\{(g, g_1, \ldots, g_{2n}), (\hat{g}_1, \ldots, \hat{g}_n)\}$, where $g_i = g^{(\alpha^i)}$ and $\hat{g}_i = \hat{g}^{(\alpha^i)}$ for all i.

The given problem instance $\{(g, g_1, \ldots, , g_{2n}), (\hat{g}_1, \ldots, \hat{g}_n)\}$ is used to define the CRS pp. Note that $g_{n+1} = g^{(\alpha^{n+1})}$ is *not* included in pp and will never be used by $\mathcal{B}$. Our reduction/extractor $\mathcal{B}$ then interacts with $\mathcal{A}$ as follows.

Queries: At each random oracle query, $\mathcal{B}$ returns a random element in the appropriate range. When $\mathcal{A}$ queries a hash value $H_{\mathrm{agg}}(\hat{V}, \hat{C}, C_y)$, $\mathcal{B}$ makes the corresponding queries $\boldsymbol{y} = H(\hat{V}, \hat{C})$, $\boldsymbol{t} = H_t(\boldsymbol{y}, \hat{V}, \hat{C}, C_y)$, $\{s_i = H_s(i, \hat{V}, [2, n])\}_{i=2}^n$ for itself before returning a tuple $(\delta_x, \delta_{eq}, \delta_y, \delta_v)$. At the first query involving a group element, $\mathcal{A}$ provides a representation of this group element as a linear combination of all the group elements that it observed so far in the same group.

At any time, $\mathcal{A}$ can choose a commitment $\mathsf{com} = \hat{V}$ and ask for a simulated proof that $\hat{V}$ is a commitment to some integer in $[0, 2^\ell - 1]$ for some $\ell \leq n$ of its choice. Since $\mathcal{A}$ is algebraic, it provides a representation of $\hat{V}$ w.r.t. $\{\hat{g}_i\}_{i \in [0,n]}$ and the commitments $\hat{C}$ contained in earlier simulated proofs. However, the simulator used by $\mathcal{B}$ is itself algebraic and always simulates proofs by computing $\hat{C}$ as a linear combination of $\{\hat{g}_i\}_{i \in [0,n]}$ for coefficients of its choice. Hence, for any $\hat{V}$ chosen by $\mathcal{A}$, $\mathcal{B}$ can always compute a representation $\{v_i\}_{i=0}^n$ such that $\hat{V} = \hat{g}^{v_0} \cdot \prod_{i=1}^n \hat{g}_i^{v_i}$. We assume w.l.o.g. that either $v_1 \notin [0, 2^\ell - 1]$ or $(v_2, \ldots, v_n) \neq \mathbf{0}$ since, otherwise, $\mathcal{B}$ can generate a real proof using (v_1, v_0). Then, $\mathcal{B}$ simulates a proof as follows without using g_{n+1}:

1. Choose random vectors $\boldsymbol{\delta} = (\delta_x, \delta_{eq}, \delta_y, \delta_v) \xleftarrow{R} \mathbb{Z}_p^4$, $\boldsymbol{y} = (y_1, \ldots, y_n) \xleftarrow{R} \mathbb{Z}_p^n$, $\boldsymbol{t} = (t_1, \ldots, t_n) \xleftarrow{R} \mathbb{Z}_p^n$.
2. Let $f_{n+1} = \sum_{i=2}^n v_i \cdot s_i$ for random $s_2, \ldots, s_n \xleftarrow{R} \mathbb{Z}_p$. Define $z_n = y_1$ and

$$a_1 = v_1 - \frac{\delta_v \cdot f_{n+1}}{\delta_x} \qquad\qquad \forall i \in [2, n] \ : \ a_i = 0$$

   Note that $a_1 \notin \{0, 1\}$ w.h.p. if $v_1 \notin [0, 2^\ell - 1]$ or $(v_2, \ldots, v_n) \neq \mathbf{0}$. Then, compute an arbitrary vector $(z_1, \ldots, z_{n-1}) \in \mathbb{Z}_p^{n-1}$ satisfying the equality $\sum_{i=2}^n t_i \cdot z_{n+1-i} = t_1 \cdot (a_1 \cdot y_1 - y_1)$.
3. Choose random $a_0, z_0 \xleftarrow{R} \mathbb{Z}_p$ and compute simulated commitments

$$\hat{C} = \hat{g}^{a_0} \cdot \prod_{i=1}^n \hat{g}_i^{a_i} = \hat{g}^{a_0} \cdot \hat{g}^{a_1}, \qquad\qquad C_y = g^{z_0} \cdot \prod_{i=1}^n g_i^{z_i}.$$

4. If one of the random oracle values $H_{\mathrm{agg}}(\hat{V}, \hat{C}, C_y)$, $H(\hat{V}, \hat{C})$, $H_t(\boldsymbol{y}, \hat{V}, \hat{C}, C_y)$ or $\{H_s(i, [2, n], \hat{V}, \hat{C}, C_y)\}_{i=2}^n$ was already defined, then abort and report failure. Otherwise, set $\boldsymbol{y} = H(\hat{V}, \hat{C})$, $\boldsymbol{t} = H_t(\boldsymbol{y}, \hat{V}, \hat{C}, C_y)$, $\boldsymbol{\delta} = H_{\mathrm{agg}}(\hat{V}, \hat{C}, C_y)$ and $s_i = H_s(i, [2, n], \hat{V}, \hat{C}, C_y)$ for each $i \in [2, n]$.
5. Define the polynomials

$$Q_x[X] = \Big(\sum_{i=0}^n a_i \cdot X^i\Big) \cdot \Big(\sum_{i=1}^\ell 2^{i-1} \cdot X^{n+1-i}\Big) - \Big(\sum_{i=0}^n v_i \cdot X^{i+n}\Big) = \sum_{i=0}^{n+\ell} q_i \cdot X^i,$$

$$Q_y[X] = \Big(\sum_{i=0}^n z_i \cdot X^i - \sum_{i=1}^n y_i \cdot X^{n+1-i}\Big) \cdot \Big(\sum_{i=0}^n a_i \cdot X^i\Big) = \sum_{i=0}^{2n} \sigma_i \cdot X^i$$

$$Q_{eq}[X] = \Big(\sum_{i=0}^{n} a_i \cdot X^i\Big) \cdot \Big(\sum_{i=1}^{n} t_i \cdot y_i \cdot X^{n+1-i}\Big) - \Big(\sum_{i=0}^{n} z_i \cdot X^i\Big) \cdot \Big(\sum_{i=1}^{n} t_i \cdot X^i\Big) = \sum_{j=0}^{2n} e_j \cdot X^j,$$

$$Q_v[X] = \Big(\sum_{i=0}^{n} v_i \cdot X^i\Big) \cdot \Big(\sum_{i=2}^{n} s_i \cdot X^{n+1-i}\Big) = \sum_{j=0}^{2n} f_j \cdot X^j.$$

Their degree-$(n+1)$ coefficients are $f_{n+1} = \sum_{i=2}^{n} v_i \cdot s_i$ and

$$q_{n+1} = -v_1 + \sum_{i=1}^{\ell} a_i \cdot 2^{i-1} = -v_1 + a_1 = -\frac{\delta_v \cdot f_{n+1}}{\delta_x},$$

$$\sigma_{n+1} = \sum_{i=1}^{n} a_i \cdot (z_{n+1-i} - y_i) = a_1 \cdot (z_n - y_1) = 0$$

$$e_{n+1} = \sum_{i=1}^{n} t_i \cdot (a_i \cdot y_i - z_{n+1-i}) = t_1 \cdot (a_1 \cdot y_1 - y_1) - \sum_{i=2}^{n} t_i \cdot z_{n+1-i} = 0$$

due to the definition of $\boldsymbol{a} = (a_1, \ldots, a_n)$ and $\boldsymbol{z} = (z_1, \ldots, z_n)$. Note that

$$\delta_x \cdot q_{n+1} + \delta_{eq} \cdot e_{n+1} + \delta_y \cdot \sigma_{n+1} + \delta_v \cdot f_{n+1} = 0 \tag{23}$$

6. Define the polynomial

$$Q_{\mathrm{agg}}[X] = \delta_x \cdot Q_x[X] + \delta_{eq} \cdot Q_{eq}[X] + \delta_y \cdot Q_y[X] + \delta_v \cdot Q_v[X] = \sum_{i=0}^{2n} \eta_i \cdot X^i$$

for which $\eta_{n+1} = 0$ by construction. Compute $\pi = \prod_{i=1, i\neq n+1}^{2n} g_i^{\eta_i}$ using $(g, \{g_i\}_{i\in[2n]\setminus\{n+1\}})$ and return the simulated proof $\boldsymbol{\pi} = (\hat{C}, C_y, \pi)$.

Note that the simulated π satisfies the verification equation

$$\frac{e\big(C_y^{\delta_y} \cdot \prod_{i=1}^{\ell} g_{n+1-i}^{\delta_x \cdot 2^{i-1} + (\delta_{eq} \cdot t_i - \delta_y) \cdot y_i} \cdot \prod_{i=\ell+1}^{n} g_{n+1-i}^{(\delta_{eq} \cdot t_i - \delta_y) \cdot y_i}, \hat{C}\big)}{e\big(g_n^{\delta_x} \cdot \prod_{i=2}^{n} g_{n+1-i}^{-\delta_v \cdot s_i}, \hat{V}\big) \cdot e\big(C_y, \prod_{i=1}^{n} \hat{g}_i^{\delta_{eq} \cdot t_i}\big)} = e(\pi, \hat{g}). \tag{24}$$

and $\boldsymbol{\pi}$ has the same distribution as a proof generated by the zero-knowledge simulator in the proof of Theorem 1. Indeed, π is uniquely determined by the commitments $(\hat{C}, \hat{V}, C_y)$ and the $\mathbb{Z}_p$-elements $\boldsymbol{y}, \boldsymbol{t}$, $\{s_i\}_{i=2}^{n}$, and $\boldsymbol{\delta}$ in (24). Also, while the committed vectors $\boldsymbol{a}, \boldsymbol{z} \in \mathbb{Z}_p^n$ are programmed in a special way, they are perfectly hidden by the randomness a_0 and z_0 in $\hat{C}$ and C_y.

Consequently, the simulation is perfect, unless a collision occurs when random oracles are programmed in one of the simulation queries. If Q_S (reps. Q_H) denotes the number of queries made by $\mathcal{A}$ to the simulator (resp. to random oracles), this happens with probability at most $(Q_S + Q_H) \cdot Q_H / p$.

Output: When $\mathcal{A}$ halts, it outputs a statement $(\hat{V}, [0, 2^{\ell-1}])$, for some $\ell \in [1, n]$, together with a verifying proof $\boldsymbol{\pi} = \big(\hat{C}, C_y, \pi\big)$.

Any malicious algebraic prover that comes up with a commitment $\mathsf{com} = \hat{V}$ and a proof $\boldsymbol{\pi} = \big(\hat{C}, C_y, \pi\big)$ also gives a representation of each group element w.r.t. the group elements that have been observed so far.[4] In particular, $\mathcal{A}$ must provide a representation of C_y w.r.t to $(g, \{g_i\}_{i\in[2n]\setminus\{n+1\}})$ and the group elements $\{C_y^{(i)}, \pi^{(i)}\}_{i\in[Q_S]}$ contained in simulated proofs $\{\pi^{(i)}\}_{i\in[Q_S]}$. Likewise, $\mathcal{A}$ must provide a representation of $\hat{C}$ w.r.t $(\hat{g}, \{\hat{g}_i\}_{i\in[n]})$ and the commitments $\{\hat{C}^{(i)}\}_{i\in[Q_S]}$ contained in $\{\pi^{(i)}\}_{i\in[Q_S]}$. However, for each $i \in [Q_S]$, $\mathcal{B}$ knows a representation of $\hat{C}^{(i)}$ w.r.t. $(\hat{g}, \{\hat{g}_i\}_{i\in[n]})$ and a representation of C_y w.r.t. $(g, \{g_i\}_{i=1}^n)$. It also knows a representation of each $\pi^{(i)}$ w.r.t $(g, \{g_i\}_{i\in[2n]\setminus\{n+1\}})$. From $\mathcal{A}$'s output and the random coins of the simulation, $\mathcal{B}$ can compute scalars $\{(\theta_i, z_i) \in \mathbb{Z}_p^2\}_{i\in[0,2n]\setminus\{n+1\}}$, $\{(a_i, v_i) \in \mathbb{Z}_p^2\}_{i\in[0,n]}$ such that

$$\hat{C} = \prod_{i=0}^{n} \hat{g}_i^{a_i}, \qquad C_y = \prod_{i=0, i\neq n+1}^{2n} g_i^{z_i}, \qquad \hat{V} = \prod_{i=0}^{n} \hat{g}_i^{v_i}, \qquad \pi = \prod_{i=0, i\neq n+1}^{2n} g_i^{\theta_i},$$

where we define $g_0 = g$ and $\hat{g}_0 = \hat{g}$ for convenience.

If the representation $(v_0, v_1, \ldots, v_n) \in \mathbb{Z}_p^2$ of $\hat{V}$ is such that $v_1 \in [0, 2^\ell - 1]$ and $v_i = 0$ for all $i \in [2, n]$, then $\mathcal{B}$ is done as it can simply output $(v_1, v_0) \in \mathbb{Z}_p^2$ as a valid opening of the Pedersen commitment $\hat{V}$ to an integer v_1 in the proper range. We now assume that either $v_1 \notin [0, 2^\ell - 1]$ or $(v_2, \ldots, v_n) \neq \mathbf{0}^{n-1}$.

Solving $(2n, n)$-DLOG: By hypothesis, $\mathcal{A}$'s statement $(\mathsf{com} = \hat{V}, [0, 2^{\ell-1}])$ and proof $\boldsymbol{\pi} = \big(\hat{C}, C_y, \pi\big)$ satisfy (24), where $\boldsymbol{y} = H(\hat{V}, \hat{C})$, $\boldsymbol{t} = H_t(\boldsymbol{y}, \hat{V}, \hat{C}, C_y)$, $s_0 = 0$, $s_i = H_s(i, [2, n], \hat{V}, \hat{C}, C_y)$ for $i \in [2, n]$, and $(\delta_x, \delta_{eq}, \delta_y, \delta_v) = H_{\text{agg}}(\hat{V}, \hat{C}, C_y)$.

We first note that a non-trivial valid $\boldsymbol{\pi}$ cannot recycle $(\hat{V}, \hat{C}, C_y)$ obtained from the simulation oracle (namely, we must have $(\hat{V}, \hat{C}, C_y) \neq (\hat{V}^{(i)}, \hat{C}^{(i)}, C_y)$ for all $i \in [Q_S]$) since the left-hand-side member of (24) is uniquely determined by $(\hat{V}^{(i)}, \hat{C}^{(i)}, C_y^{(i)})$ and it in turn determines a unique valid $\pi^{(i)}$. Consequently, the hash values $H_{\text{agg}}(\hat{V}, \hat{C}, C_y)$, $H_t(\boldsymbol{y}, \hat{V}, \hat{C}, C_y)$ and $\{H_s(i, [2, n], \hat{V}, \hat{C}, C_y)\}_{i=2}^n$ were *not* programmed by the simulator in a simulation query.

We also note that the left-hand-side member of (24) is obtained by raising those of (17)-(20) to the powers $(\delta_x, \delta_{eq}, \delta_y, \delta_v)$ and multiplying the results together. Hence, it can be written $e(g, \hat{g})^{P_{\text{agg}}(\alpha)}$, where $P_{\text{agg}}[X]$ is the polynomial

$$P_{\text{agg}}[X] = \delta_x \cdot P_x[X] + \delta_y \cdot P_y[X] + \delta_{eq} \cdot P_{eq}[X] + \delta_v \cdot P_v[X]$$

[4] These representations are supplied by $\mathcal{A}$ at the first query involving the corresponding group elements.

obtained as a linear combination of the polynomials

$$P_x[X] = \Big(\sum_{i=0}^{n} a_i \cdot X^i\Big) \cdot \Big(\sum_{i=1}^{\ell} 2^{i-1} \cdot X^{n+1-i}\Big) - \big(\sum_{i=0}^{n} v_i \cdot X^{n+i}\big) = \sum_{i=0}^{n+\ell} \omega_i \cdot X^i,$$

$$P_y[X] = \Big(z_0 + \sum_{i=1}^{n} \big(z_{n+1-i} - y_i\big) \cdot X^{n+1-i} + \sum_{i=n+2}^{2n} z_i \cdot X^i\Big) \cdot \Big(\sum_{i=0}^{n} a_i \cdot X^i\Big) = \sum_{i=0}^{3n} \gamma_i \cdot X^i$$

$$P_{eq}[X] = \Big(\sum_{i=0}^{n} a_i \cdot X^i\Big) \cdot \Big(\sum_{i=1}^{n} t_i \cdot y_i \cdot X^{n+1-i}\Big) - \Big(\sum_{i=0, i\neq n+1}^{2n} z_i \cdot X^i\Big) \cdot \Big(\sum_{i=1}^{n} t_i \cdot X^i\Big) = \sum_{j=0}^{3n} \beta_j \cdot X^j,$$

$$P_v[X] = \Big(\sum_{i=0}^{n} v_i \cdot X^i\Big) \cdot \Big(\sum_{i=2}^{n} s_i \cdot X^{n+1-i}\Big) = \sum_{j=0}^{2n} \mu_j \cdot X^j$$

for which the left-hand-side members of (17)-(20) can be written $e(g,\hat{g})^{P_x(\alpha)}$, $e(g,\hat{g})^{P_{eq}(\alpha)}$, $e(g,\hat{g})^{P_y(\alpha)}$, and $e(g,\hat{g})^{P_v(\alpha)}$, respectively.

Letting $P_{\text{agg}}[X] = \sum_{i=0}^{3n} \nu_i \cdot X^i$, the coefficient of its degree-$(n+1)$ term is

$$\nu_{n+1} = \delta_x \cdot \underbrace{(\sum_{i=1}^{\ell} a_i \cdot 2^{i-1} - v_1)}_{\triangleq\ \omega_{n+1}} + \delta_y \cdot \underbrace{\sum_{i=1}^{n} \big(z_{n+1-i} - y_i\big) \cdot a_i}_{\triangleq\ \gamma_{n+1}} + \delta_{eq} \cdot \underbrace{\sum_{i=1}^{n} t_i \cdot (a_i \cdot y_i - z_{n+1-i})}_{\triangleq\ \beta_{n+1}} + \delta_v \cdot \underbrace{\sum_{i=2}^{n} v_i \cdot s_i}_{\triangleq\ \mu_{n+1}},$$

where $(\omega_{n+1}, \gamma_{n+1}, \beta_{n+1}, \mu_{n+1})$ are the coefficients of the degree-$(n+1)$ terms of $(P_x[X], P_y[X], P_{eq}[X], P_v[X])$, respectively. We argue that, if $v_1 \notin [0, 2^\ell - 1]$ or $(v_2, \ldots, v_n) \neq \mathbf{0}^{n-1}$, we cannot have $\nu_{n+1} = 0$, except with negligible probability. This follows from the following two arguments:

- The probability that $\boldsymbol{\rho} \triangleq (\omega_{n+1}, \gamma_{n+1}, \beta_{n+1}, \mu_{n+1}) = \mathbf{0}$ is negligible if $v_1 \notin [0, 2^\ell - 1]$ or $(v_2, \ldots, v_n) \neq \mathbf{0}^{n-1}$. Indeed, when $(v_2, \ldots, v_n) \neq \mathbf{0}^{n-1}$, we have $\mu_{n+1} = 0$, with probability $1/p$ over the random choice of $\{s_i\}_{i=2}^n$ since $\{s_i = H_s(i, [2,n], \hat{V}, \hat{C}, C_y)\}_{i=2}^n$ are derived uniformly *after* the choice of $\{v_i\}_{i=2}^n$. Likewise, when $z_{n+1-i} \neq a_i \cdot y_i$ for some $i \in [n]$, we have $\beta_{n+1} = 0$ with probability $1/p$ since $\boldsymbol{t} = H_t(\boldsymbol{y}, \hat{V}, \hat{C}, C_y)$ is derived *after* the choice of $\boldsymbol{y}$, $\{a_i\}_{i=0}^n$ and $\{z_i\}_{i\in[0,2n]\setminus\{n+1\}}$. Then, if $z_{n+1-i} = a_i \cdot y_i$ for all $i \in [n]$, we have $\gamma_{n+1} = \sum_{i=1}^n y_i \cdot (a_i - 1) \cdot a_i$, which cancels with probability $1/p$ if there exists $i \in [n]$ such that $a_i \notin \{0, 1\}$. To see this, we distinguish two cases:

a. If $\boldsymbol{y} = H(\hat{V}, \hat{C})$ was programmed in a simulation query, we only have $\gamma_{n+1} = 0$ with probability $1/p$ since $\mathcal{B}$ chose $(a_1, \ldots, a_n)$ so as to have $\gamma_{n+1} = \sum_{i=1}^{n} y_i \cdot a_i \cdot (a_i - 1) = y_1 \cdot a_1 \cdot (a_1 - 1)$ with $y_1 \in_R \mathbb{Z}_p$ and $a_1 \notin \{0, 1\}$. This covers the case of $\mathcal{A}$ attempting to recycle $(\hat{V}, \hat{C}) = (\hat{V}^{(i)}, \hat{C}^{(i)})$ from a simulated $\boldsymbol{\pi}^{(i)} = (\hat{C}^{(i)}, C_y^{(i)}, \pi^{(i)})$, with a modified $C_y \neq C_y^{(i)}$.
b. If $H(\hat{V}, \hat{C})$ was not programmed by the simulator, then $\boldsymbol{y} = H(\hat{V}, \hat{C})$ was defined after $\mathcal{B}$ obtained the representation $\{a_i\}_{i=0}^{n}$ of $\hat{C}$. Over the choice of $\boldsymbol{y}$, we have $\sum_{i=1}^{n} y_i \cdot (a_i - 1) \cdot a_i = 0$ with probability $1/p$.

If none of the above events occurs and $\omega_{n+1} = 0$, we have $v_1 = \sum_{i=1}^{\ell} a_i \cdot 2^{i-1}$ and $a_i \in \{0, 1\}$ for all $i \in [\ell]$, which contradicts the hypothesis $v_1 \notin [0, 2^{\ell} - 1]$.

- If $\boldsymbol{\rho} \neq \mathbf{0}$, then $\nu_{n+1} \neq 0$ with probability $1 - 1/p$ since $\boldsymbol{\delta} = H_{\mathrm{agg}}(\hat{V}, \hat{C}, C_y)$ is derived *after* the choice of $\{(a_i, v_i)\}_{i=0}^{n}$, and $\{z_i\}_{i \in [0,2n] \setminus \{n+1\}}$, which determine $\boldsymbol{\rho}$. So, a random $\boldsymbol{\delta} \in \mathbb{Z}_p^4$ satisfies $\langle \boldsymbol{\delta}, \boldsymbol{\rho} \rangle = 0$ with probability $1/p$.

If $\nu_{n+1} \neq 0$, $\mathcal{B}$ can compute $\alpha \in \mathbb{Z}_p$ by observing that the aggregated verification equation (24) implies

$$\pi = g_{n+1}^{\nu_{n+1}} \cdot \prod_{i \in [0,3n] \setminus \{n+1\}} g_i^{\nu_i}, \tag{25}$$

where $g_{2n+1} = g^{(\alpha^{2n+1})}, \ldots, g_{3n} = g^{(\alpha^{3n})}$ are not available. However, $\mathcal{B}$ knows $\{\nu_i\}_{i=0}^{3n}$. Since $\nu_{n+1} \neq 0$, we are guaranteed that the representation (25) of π differs from its representation $\pi = \prod_{i=0, i \neq n+1}^{2n} g_i^{\theta_i}$ revealed by $\mathcal{A}$ as part of its output. This means that $\alpha \in \mathbb{Z}_p$ can be found among the roots of the non-zero

$$R[X] = \sum_{i \in [0,2n] \setminus \{n+1\}} (\nu_i - \theta_i) \cdot X^i + \nu_{n+1} \cdot X^{n+1} + \sum_{i=2n+1}^{3n} \nu_i,$$

□

4.3 Batched Range Proofs and Proving the Smallness of Vectors

As detailed in Supplementary Material D, the construction extends to prove multiple ranges at once for a committed vector $\hat{V} = \hat{g}^r \cdot \prod_{k=1}^{m} \hat{g}_k^{x_k}$ of integers. Namely, $\hat{C}$ commits to concatenation of the binary decompositions of all $\{x_k\}_{k=1}^{m}$. Then, a single group element allows proving that, for each $k \in [m]$, the k-th subvector $\boldsymbol{x}_k = (x_{k,1}, \ldots, x_{k,\ell}, 0, \ldots, 0)$ hidden by $\hat{C}$ is a binary vector satisfying $x_k = \sum_{i=1}^{\ell} x_{k,i} \cdot 2^{i-1}$. For the k-th slot, the prover computes $\pi_k \in \mathbb{G}$ such that

$$e\Big(\prod_{i=1}^{\ell} g_{n+1-((k-1)\bar{\ell}+i)}^{2^{i-1}}, \hat{C}\Big) = e(g_1, \hat{g}_n)^{x_k} \cdot e(\pi_k, \hat{g}) \tag{26}$$

Since $\hat{V}$ is itself a vector commitment, the prover can compute $\pi_{v,k}$ such that

$$e\big(g_k, \hat{V}\big) = e(g_1, \hat{g}_n)^{x_k} \cdot e(\pi_{v,k}, \hat{g}) \tag{27}$$

Then, by dividing (27) from (26), raising the result to a random power $\xi_k \in \mathbb{Z}_p$ and taking the product over all indices $k \in [m]$, we find that the prover is able to compute a short $\pi = \prod_{k=1}^{m}(\pi_k/\pi_{v,k})^{\xi_k}$ such that

$$\frac{e\big(\prod_{k=1}^{m}\big(\prod_{i=1}^{\ell} g_{n+1-((k-1)\bar{\ell}+i)}^{2^{i-1}}\big)^{\xi_k}, \hat{C}\big)}{e\big(\prod_{k=1}^{m} g_k^{\xi_k}, \hat{V}\big)} = e(\pi, \hat{g}), \tag{28}$$

which argues that $x_k = \sum_{i=1}^{\ell} x_{k,i} \cdot 2^{i-1}$ for all $k \in [m]$. Indeed, otherwise, we have $\sum_{k=1}^{m} \xi_k \cdot (x_k - \sum_{i=1}^{\ell} x_{k,i} \cdot 2^{i-1}) = 0$ with negligible probability $1/p$ as long as $(\xi_1, \ldots, \xi_m)$ are chosen uniformly after the commitments $\hat{V}$ and $\hat{C}$.

The remaining proof elements are computed exactly as in the single-slot setting, so that the final proof $\boldsymbol{\pi}$ still lives in $\hat{\mathbb{G}} \times \mathbb{G}^2$. This immediately provides a short proof that a committed vector has small infinity norm. By introducing a few more group elements in the proof, we can also prove small Euclidean norms, as explained in Supplementary Material E.

4.4 Comparisons

Our construction assumes that the witness x is committed using a Pedersen commitment in the pairing-friendly group specified by the CRS of the range proof. The range proof of [12] similarly requires x to be committed as a constant polynomial using the CRS of a polynomial commitment scheme.

The BFGW range proofs [12] were the shortest ones so far and they also feature constant verification time (whereas our verifier computes $O(n)$ exponentiations, where n is the maximal bitlength of the range, as in BulletProofs). When instantiated with KZG commitments [74] and the cross-commitment evaluation techniques of [11, Section 4.1], BFGW proofs consist of 2 commitments to polynomials (each of which takes an element of $\mathbb{G}$), 3 elements of $\mathbb{Z}_p$ representing evaluations of committed polynomials, and a batched evaluation proof comprised of a group element and at least one scalar.[5] If their construction is instantiated with the polynomial commitment of [80, Section 4.1][6] and the batched evaluation protocol of [11, Section 4.1], the communication cost decreases to 2 elements of $\mathbb{G}$ (which commit to polynomials), 3 scalars (for polynomial evaluations) and a single element of $\mathbb{G}$ for the batched evaluation proof. In the latter case, the range proof of [12] only requires 3 elements of $\mathbb{G}$ and 3 elements of $\mathbb{Z}_p$. On the downside, combining [12,80] induces $2n$ exponentiations in $\hat{\mathbb{G}}$ at the verifier (instead of $O(1)$ using KZG commitments) and increases the prover's overhead to $7n$ exponentiations in $\mathbb{G}$.

[5] In randomized versions of the KZG commitment (described in [74, Section 3.3], [11, Appendix B.2] and [99]), each evaluation proof consists of an element of $\mathbb{G}$ and at least one scalar or an additional element of $\mathbb{G}$.

[6] In order to prove the knowledge soundness of the range proof of [12] when the polynomial commitment of [80] is used, it is necessary to rely on the latter's knowledge soundness in the AGM (as defined in [11, Appendix C.1.3]) but we believe this property holds under the $(2n, n)$-DLOG assumption.

Not only does our construction ensure simulation-extractability in the AGM, it also features the smallest number of exponentiations at the prover (which is reduced by at least 40%) while matching the shortest proof length of SNARKs.

In terms of space, our construction also improves upon BulletProofs [19], which requires the prover to send $2\lceil \log \ell \rceil + 4$ group elements and 5 elements of $\mathbb{Z}_p$. If we compare with SNARKs, we obtain the same proof size as optimally short candidates [67,68] with the advantage that our CRS size is much shorter: It only depends on the maximal bitlength n of a range rather than the size of a circuit representation of the statement. Also, our prover only needs to compute $O(n)$ exponentiations instead of a number of exponentiations growing with the size of an arithmetic circuit that computes a commitment opening (which would be very large as the circuit would have to compute modular exponentiations).

In Table 1, we compare our constant-size range proofs with existing pairing-based solutions featuring similarly short proofs. Several instantiations of [10] are considered for different polynomial commitment schemes that are known to provide constant-size evaluation proofs. Among schemes that do not generically rely on SNARKs, we only consider those where the CRS size is at most logarithmic in the maximal range magnitude $N = 2^n$ (i.e., linear in n). For example, Table 1 does not include range proofs based on lookup arguments [49,98] as they would require a CRS of size $O(N) = O(2^n)$. For a range $[0, 2^{30}]$, this would translate into a CRS of about 30 Gb instead of 6 Kb in our construction.

Table 1. Efficiency comparisons between constant-size range proofs

Schemes	Proof size	CRS size	Prover cost	Verifier cost
BFGW [12]	$3 \times \lvert\mathbb{G}\rvert$	$(4n+2) \times \lvert\mathbb{G}\rvert$	$5n\ \mathrm{exp}_{\mathbb{G}}$	$3\mathrm{P} + 4\,\mathrm{exp}_{\hat{\mathbb{G}}}$
+ KZG [74, Section 3.3]	$4 \times \lvert\mathbb{Z}_p\rvert$	$4 \times \lvert\hat{\mathbb{G}}\rvert$		$1\,\mathrm{exp}_{\mathbb{G}}$
BFGW	$4 \times \lvert\mathbb{G}\rvert$	$(2n+1) \times \lvert\mathbb{G}\rvert$	$5n\ \mathrm{exp}_{\mathbb{G}}$	$3\mathrm{P} + 4\,\mathrm{exp}_{\hat{\mathbb{G}}}$
+ Zhang *et al.* [99]	$3 \times \lvert\mathbb{Z}_p\rvert$	$3 \times \lvert\hat{\mathbb{G}}\rvert$		$1\,\mathrm{exp}_{\mathbb{G}}$
BFGW	$3 \times \lvert\mathbb{G}\rvert$	$4n \times \lvert\mathbb{G}\rvert$	$7n\ \mathrm{exp}_{\mathbb{G}}$	$3\mathrm{P} + 2n\,\mathrm{exp}_{\hat{\mathbb{G}}}$
+ LRY [80]	$3 \times \lvert\mathbb{Z}_p\rvert$	$2n \times \lvert\hat{\mathbb{G}}\rvert$		$2\,\mathrm{exp}_{\mathbb{G}}$
Groth16 [67]	$1 \times \lvert\hat{\mathbb{G}}\rvert$	$3\lvert\mathcal{C}\rvert \times \lvert\mathbb{G}\rvert$	$4\lvert\mathcal{C}\rvert\ \mathrm{exp}_{\mathbb{G}}$	$3\mathrm{P} + O(1)\,\mathrm{exp}_{\mathbb{G}}$
	$2 \times \lvert\mathbb{G}\rvert$	$\lvert\mathcal{C}\rvert \times \lvert\hat{\mathbb{G}}\rvert$	$\lvert\mathcal{C}\rvert\ \mathrm{exp}_{\hat{\mathbb{G}}}$	
New construction	$1 \times \lvert\hat{\mathbb{G}}\rvert$	$2n \times \lvert\mathbb{G}\rvert$	$3n\ \mathrm{exp}_{\mathbb{G}}$	$4\mathrm{P} + 2n\,\mathrm{exp}_{\mathbb{G}}$
(Section 4)	$2 \times \lvert\mathbb{G}\rvert$	$n \times \lvert\hat{\mathbb{G}}\rvert$	$1\,\mathrm{exp}_{\hat{\mathbb{G}}}$	$n\,\mathrm{exp}_{\hat{\mathbb{G}}}$
			$n\ \mathrm{mult}_{\hat{\mathbb{G}}}$	

$\mathrm{exp}_{\mathbb{G}}$ and $\mathrm{exp}_{\hat{\mathbb{G}}}$ denote exponentiations in $\mathbb{G}$ and $\hat{\mathbb{G}}$ while $\mathrm{mult}_{\hat{\mathbb{G}}}$ denotes a multiplication in $\hat{\mathbb{G}}$; n denotes the bitlength of the range; P stands for a pairing computation; $|\mathcal{C}|$ is the size of the arithmetic circuit verifying a commitment opening.

References

1. M. Albrecht, V. Cini, R.-F. Lai, G. Malavolta, and S.-A. Thyagarajan. Lattice-based SNARKs: Publicly verifiable, preprocessing, and recursively composable. In *Crypto*, 2022.

2. A. Arun, C. Ganesh, S. Lokam, T. Mopuri, and S. Sridhar. A transparent constant-sized polynomial commitment scheme. In *PKC*, 2023.
3. K. Baghery, M. Kohlweiss, J. Siim, and M. Volkhov. Another look at extraction and randomization of Groth's zk-SNARK. Cryptology ePrint Archive Report 2020/811.
4. K. Baghery, Z. Pindado, and C. Ràfols. Simulation extractable versions of Groth's zk-SNARK revisited. In *CANS*, 2020.
5. D. Balbas, D. Catalano, D. Fiore, and R.-F. Lai. Chainable functional commitments: From quadratic polynomials to unbounded-depth circuits. In *TCC*, 2023.
6. R. Barbulescu and S. Duquesne. Updating key size estimations for pairings. *J. of Cryptology*, 4(32), 2019.
7. P. Barreto, B. Lynn, and M. Scott. Constructing elliptic curves with prescribed embedding degrees. In *SCN*, 2002.
8. E. Ben-Sasson, A. Chiesa, M. Riabzev, N. Spooner, M. Virza, and N. Ward. Aurora: Transparent succinct arguments for R1CS. In *Eurocrypt*, 2019.
9. D. Boneh, B. Bünz, and B. Fisch. Batching techniques for accumulators with applications to IOPs and stateless blockchains. In *Crypto*, 2019.
10. D. Boneh, J. Drake, B. Fisch, and A. Gabizon. Efficient polynomial commitmentschemes for multiple points and polynomials. Cryptology ePrint Archive Report 2020/81, 2020.
11. D. Boneh, J. Drake, B. Fisch, and A. Gabizon. Halo infinite: Recursive zk-SNARKs from any additive polynomial commitment scheme. In *Crypto*, 2021.
12. D. Boneh, B. Fisch, A. Gabizon, and Z. Williamson. A simple range proof from polynomial commitments. `https://hackmd.io/@dabo/B1U4kx8XI`, 2020.
13. D. Boneh, W. Nguyen, and A. Ozdemir. Efficient functional commitments: How to commit to a private function. Cryptology ePrint Archive Report 2021/1342.
14. D. Boneh and V. Venkatesan. Breaking RSA may not be equivalent to factoring. In *Eurocrypt*, 1998.
15. F. Boudot. Efficient proofs that a committed number lies in an interval. In *Eurocrypt*, 2000.
16. S. Bowe and A. Gabizon. Making Groth's zk-SNARK simulation extractable in the random oracle model. Cryptology ePrint Archive Report 2018/187.
17. Z. Brakerski, C. Gentry, and V. Vaikuntanathan. (Leveled) Fully Homomorphic Encryption without Bootstrapping. In *ITCS*, 2012.
18. E. Brickell, D. Chaum, I. Damgård, and J. van de Graaf. Gradual and verifiable release of a secret. In *Crypto*. 1988.
19. B. Bünz, J. Bootle, D. Boneh, A. Poelstra, P. Wuille, and G. Maxwell. Bulletproofs: Short proofs for confidential transactions and more. In *IEEE S&P*, 2018.
20. B. Bünz, B. Fisch, and A. Szepieniec. Transparent SNARKs from DARK compilers. In *Eurocrypt*, 2020.
21. B. Bünz, M. Maller, P. Mishra, N. Tyagi, and P. Vesely. Proofs for inner pairing products and applications. In *Asiacrypt*, 2021.
22. J. Camenisch, R. Chaabouni, and A. shelat. Efficient protocols for set membership and range proofs. In *Asiacrypt*, 2008.
23. J. Camenisch, M. Dubovitskaya, K. Haralambiev, and M. Kohlweiss. Composable and modular anonymous credentials: Definitions and practical constructions. In *Asiacrypt*, 2015.
24. M. Campanelli, D. Fiore, N. Greco, D. Kolonelos, and L. Nizzardo. Incrementally aggregatable vector commitments and applications to verifiable decentralized storage. In *Asiacrypt*, 2020.
25. M. Campanelli, A. Nitulsecu, C. Ràfols, A. Zacharakis, and A. Zapico. Linear-map vector commitments and their practical applications. In *Asiacrypt*, 2022.

26. D. Catalano and D. Fiore. Vector commitments and their applications. In *PKC*, 2013.
27. D. Catalano, D. Fiore, R. Gennaro, and E. Giunta. On the impossibility of algebraic vector commitments in pairing-free groups. In *TCC*, 2022.
28. D. Catalano, D. Fiore, and I. Tucker. Additive-homomorphic functional commitments and applications to homomorphic signatures. In *Asiacrypt*, 2022.
29. R. Chaabouni, H. Lipmaa, and B. Zhang. A non-interactive range proof with constant communication. In *Financial Cryptography*, 2012.
30. A. Chan, Y. Frankel, and Y. Tsiounis. Easy come – easy go divisible cash. In *Eurocrypt*, 1998.
31. J.-H. Cheon. Security analysis of the Strong Diffie-Hellman problem. In *Eurocrypt*, 2006.
32. I. Chillotti, N. Gama, M. Georgieva, and M. Izabachène. TFHE: Fast fully homomorphic encryption over the torus. *Journal of Cryptology*, 33(1), 2020.
33. H. Chu, D. Fiore, D. Kolonelos, and D. Schröder. Inner product functional commitments with constant-size public parameters and openings. In *SCN*, 2022.
34. G. Couteau, D. Goudarzi, M. Klooß, and M. Reichle. Sharp: Short Relaxed Range Proofs. In *ACM-CCS*, 2022.
35. G. Couteau, M. Klooß, H. Lin, and M. Reichle. Efficient range proofs with transparent setup from bounded integer commitments. In *Eurocrypt*, 2021.
36. G. Couteau, T. Peters, and D. Pointcheval. Removing the strong RSA assumption from arguments over the integers. In *Eurocrypt*, 2017.
37. W. Dai. PESCA: a privacy-enhancing smart-contract architecture. Manuscript.
38. I. Damgård, J. Luo, S. Oechsner, P. Scholl, and M. Simkin. Compact zero-knowledge proofs of small Hamming weight. In *PKC*, 2018.
39. S. Das, P. Camacho, Z. Xiang, J. Nieto, B. Bünz, and L. Ren. Threshold signatures from inner product argument: Succinct, weighted, and multi-threshold. In *ACM-CCS*, 2023.
40. L. de Castro and C. Peikert. Functional commitments for all functions, with transparent setup. In *Eurocrypt*, 2023.
41. R. del Pino, V. Lyubashevsky, and G. Seiler. Short discrete log proofs for FHE and Ring-LWE ciphertexts. In *PKC*, 2019.
42. D. Dolev, C. Dwork, and M. Naor. Non-malleable cryptography. In *STOC*, 1991.
43. Y. El Housni. The arithmetic of pairing-based proof systems. PhD Thesis, Nov. 2022.
44. Y. El Housni and A. Guillevic. Optimized and secure pairing-friendly elliptic curves suitable for one layer proof composition. In *CANS*, 2020.
45. T. Feneuil, J. Maire, M. Rivain, and D. Vergnaud. Zero-knowledge protocols for the subset sum problem from MPC-in-the-head with rejection. In *Asiacrypt*, 2022.
46. A.-L. Ferrara, M. Green, S. Hohenberger, and M.-O. Pedersen. Practical short signature batch verification. In *CT-RSA*, 2009.
47. D. Fiore, A. Nitulescu, and D. Pointcheval. Boosting verifiable computation on encrypted data. In *PKC*, 2020.
48. G. Fuchsbauer, E. Kiltz, and J. Loss. The algebraic group model and its applications. In *Crypto*, 2018.
49. A. Gabizon and Z. Williamson. Plookup: A simplified polynomial protocol for lookup tables. Cryptology ePrint Archive Report 2020/315.
50. G. Gabizon, Z. Williamson, and O. Ciobotaru. PLONK: Permutations over Lagrange-bases for oecumenical noninteractive arguments of knowledge. Cryptology ePrint Archive, Report 2019/953,2019.

51. C. Ganesh, H. Khoshakhlagh, M. Kohlweiss, A. Nitulescu, and M. Zajac. What makes Fiat-Shamir zkSNARKs (Updatable SRS) simulation extractable? In *SCN*, 2022.
52. C. Ganesh, A. Nitulescu, and E. Soria-Vazquez. Rinocchio: SNARKs for ring arithmetic. Cryptology ePrint Archive Report 2021/322.
53. C. Ganesh, C. Orlandi, M. Pancholi, A. Takahashi, and D. Tschudi. Fiat–Shamir Bulletproofs are Non-Malleable (in the Algebraic Group Model). In *Eurocrypt*, 2022.
54. R. Gennaro, C. Gentry, B. Parno, and M. Raykova. Quadratic span programs and succinct NIZKs without PCPs. In *Eurocrypt*, 2013.
55. C. Gentry, C. Peikert, and V. Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *STOC*, 2008.
56. C. Gentry, A. Sahai, and B. Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In *Crypto*, 2013.
57. C. Gentry and D. Wichs. Separating succinct non-interactive arguments from all falsifiable assumptions. In *STOC*, 2011.
58. A. González. Efficient non-interactive zero-knowledge proofs. PhD thesis Universidad De Chile, Santiago, Mar. 2017.
59. A. González, A. Hevia, and C. Ràfols. QA-NIZK arguments in asymmetric groups: New tools and new constructions. In *Asiacrypt*, 2015.
60. A. Gonzalez and C. Ràfols. New techniques for non-interactive shuffle and range arguments. In *ACNS*, 2017.
61. S. Gorbunov, L. Reyzin, H. Wee, and Z. Zhang. PointProofs: Aggregating Proofs for Multiple Vector Commitments. In *ACM-CCS*, 2020.
62. S. Gorbunov, V. Vaikuntanathan, and D. Wichs. Leveled fully homomorphic signatures from standard lattices. In *STOC*, 2015.
63. J. Groth. Non-interactive zero-knowledge arguments for voting. In *ACNS*, 2005.
64. J. Groth. Simulation-sound NIZK proofs for a practical language and constant size group signatures. In *Asiacrypt*, 2006.
65. J. Groth. Short pairing-based non-interactive zero-knowledge arguments. In *Asiacrypt*, 2010.
66. J. Groth. Efficient zero-knowledge arguments from two-tiered homomorphic commitments. In *Asiacrypt*, 2011.
67. J. Groth. On the size of pairing-based non-interactive arguments. In *Eurocrypt*, 2016.
68. J. Groth and M. Maller. Snarky signatures: Minimal signatures of knowledge from simulation-extractable SNARKs. In *Crypto*, 2017.
69. J. Groth and A. Sahai. Efficient non-interactive proof systems for bilinear groups. In *Eurocrypt*, 2008.
70. A. Guillevic and S. Singh. On the alpha value of polynomials in thetower number field sieve algorithm. Cryptology ePrint Archive Report 2019/885, 2019.
71. Y. Ishai, E. Kushilevitz, and R. Ostrovsky. Efficient arguments without short PCPs. In *CCC*, 2007.
72. M. Joye. TFHE public key encryption revisited. Cryptology ePrint Archive Report 2023/603, Apr. 2023.
73. E. Kachisa, E. Schaefer, and M. Scott. Constructing Brezing-Weng pairing-friendly elliptic curves using elements in the cyclotomic field. In *Pairing*, 2008.
74. A. Kate, G. Zaverucha, and I. Goldberg. Constant-size commitments to polynomialsand applications. In *Asiacrypt*, 2010.

75. J. Katz, C. Zhang, and H.-S. Zhou. An analysis of the algebraic group model. In *Asiacrypt*, 2022.
76. A. Kosba, C. Papamanthou, and E. Shi. xJsnark: A Framework for Efficient Verifiable Computation. In *IEEE S&P*, 2018.
77. J. Krupp, D. Schröder, M. Simkin, D. Fiore, G. Ateniese, and S. Nuernberger. Nearly optimal verifiable data streaming. In *PKC*, 2016.
78. R.-W. Lai and G. Malavolta. Subvector commitments with application to succinct arguments. In *Crypto*, 2019.
79. J. Lee. Dory: Efficient, Transparent arguments for Generalised Inner Products and Polynomial Commitments. In *TCC*, 2021.
80. B. Libert, S. Ramanna, and M. Yung. Functional commitment schemes: From polynomial commitments to pairing-based accumulators from simple assumptions. In *ICALP*, 2016.
81. B. Libert and M. Yung. Concise mercurial vector commitments and independent zero-knowledge sets with short proofs. In *TCC*, 2010.
82. H. Lipmaa. On Diophantine complexity and statistical zero-knowledge arguments. In *Asiacrypt*, 2003.
83. H. Lipmaa and K. Pavlyk. Succinct functional commitment for a large class of arithmetic circuits. In *Asiacrypt*, 2020.
84. V. Lyubashevsky. Lattice signatures without trapdoors. In *Eurocrypt*, 2012.
85. V. Lyubashevsky, C. Peikert, and O. Regev. On ideal lattices and learning with errors over rings. In *Eurocrypt*, 2010.
86. M. Maller, S. Bowe, M. Kohlweiss, and S. Meiklejohn. Sonic: Zero-knowledge SNARKs from linear-size universal and updateable structured reference strings. In *ACM-CCS*, 2019.
87. R. Merkle. A certified digital signature. In *Crypto*, 1989.
88. S. Micali. Computationally sound proofs. *SIAM J. of Computing*, 30(4), 2000.
89. P. Paillier and D. Vergnaud. Discrete-log-based signatures may not be equivalent to discrete log. In *Asiacrypt*, 2005.
90. T. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *Crypto*, 1991.
91. C. Peikert, Z. Z. Pepin, and C. Sharp. Vector and functional commitments from lattices. In *TCC*, 2021.
92. V. Shoup. Lower bounds for discrete logarithms and related problems. In *Eurocrypt*, 1997.
93. R. Solomon, R. Weber, and G. Almashaqbeh. smartFHE: Privacy-Preserving Smart Contracts from Fully Homomorphic Encryption. In *EuroS&P*, 2023.
94. S. Srinivasan, A. Chepurnoy, C. Papamanthou, A. Tomescu, and Y. Zhang. Hyperproofs: Aggregating and maintaining proofs in vector commitments. In *USENIX Security*, 2022.
95. A. Tomescu, I. Abraham, V. Buterin, J. Drake, D. Feist, and D. Khovratovich. Aggregatable subvector commitments for stateless cryptocurrencies. In *SCN*, 2020.
96. A. Tomescu, Y. Xia, and Z. Newman. Authenticated dictionaries with cross-incremental proof (dis)aggregation. Cryptology ePrint Archive Report 2020/1239.
97. H. Wee and D. Wu. Succinct vector, polynomial, and functional commitments from lattices. In *Eurocrypt*, 2023.
98. A. Zapico, V. Buterin, D. Khovratovich, M. Maller, A. Nitulescu, and M. Simkin. Caulk: Lookup arguments in sublinear time. In *ACM-CCS*, 2022.
99. Y. Zhang, D. Genkin, J. Katz, D. Papadopoulos, and C. Papamanthou. A zero-knowledge version of vSQL. Cryptology ePrint Archive Report 2017/1146.

Supplementary Material

A On Proving Smallness via Functional Commitments for Constant-Degree Polynomials

Catalano, Fiore and Tucker [28] recently built additively homomorphic FCs for constant-degree multivariate polynomials and monotone span programs. The former could be used to build short proofs of binarity by showing that the degree-2 polynomial $f(x_1, \ldots, x_n) = \sum_{i=1}^{n} y_i \cdot x_i \cdot (x_i - 1)$ evaluates to 0 for random coefficients $\{y_i\}_{i=1}^{n}$. Nevertheless, their construction for degree-d polynomials has a CRS size $O(n^d)$. If we were to use it as is to prove that a committed vector is binary, we would end up with a quadratic-size CRS (instead of linear in our construction) and longer commitments containing two group elements. Moreover, the shape of the CRS would make it harder to prove knowledge-soundness in the algebraic group model (note that their notion of evaluation-binding[7] would not suffice for our purposes). The reason is that their CRS contains elements of the form $(g^{\alpha^j}, g^{\beta \cdot \alpha^j})_{j \in [n^2]}$, for some secret $\beta, \alpha \in \mathbb{Z}_p$, while some components of honestly generated commitments are of the form $g^{\sum_{j=1}^{n} x_j \cdot (\alpha^j)}$ and only depend on $\{g^{\alpha^j}\}_{j \in [n]}$. Hence, it is not clear how the AGM would enable knowledge extraction from an adversarially-generated commitment/proof since the commitment can depend on all generators contained in the CRS, including $\{g^{\beta \cdot \alpha^j}\}_{j \in [n^2]}$.

To avoid these difficulties and decrease the CRS size to $O(n)$ group elements, it is tempting to exploit the sparsity of the polynomial $\sum_i y_i \cdot x_i \cdot (x_i - 1)$. Then, in the closest adaptation of the technique from [28, Section 4] that we can think of, either the commitment or the opening is longer than ours by at least one group element: The prover would include a commitment $\bar{C} \in \mathbb{G}$ to the product $\boldsymbol{x} \circ \boldsymbol{x} = (x_1^2, \ldots, x_n^2)$ in the opening before proving that $\boldsymbol{x} \circ \boldsymbol{x} - \boldsymbol{x}$ satisfies an inner product relation $\langle \boldsymbol{x} \circ \boldsymbol{x} - \boldsymbol{x}, \boldsymbol{y} \rangle = 0$ and that $\bar{C}$ is consistent with the initial commitment $\hat{C} = \hat{g}^{\gamma} \cdot \prod_{j=1}^{n} g_j^{x_j}$ to $\boldsymbol{x}$, which is part of the statement. To do this, the prover would have to include at least one additional group element (typically, an auxiliary commitment C to a reversed version of $\boldsymbol{y} \circ \boldsymbol{x}$ in $\mathbb{G}$ if the initial commitment $\hat{C}$ lives in $\hat{\mathbb{G}}$) either in the commitment or in the opening. Then, it would have to prove that C, and $\bar{C}$ and $\hat{C}$ are consistent with one another by computing a pairing $e(C, \hat{C})$ and a pairing of $\bar{C}$ with some public encoding of $\boldsymbol{y}$. Hence, the auxiliary commitment C would have to be part of either the initial commitment or the opening, thus increasing the global communication overhead (besides the main commitment $\hat{C}$) to 3 group elements $(C, \bar{C}, \pi)$ if π is an aggregated proof showing the consistency of all commitments. In our applications to range proofs and short proofs for ring LWE ciphertexts, this would increase the proof length by at least one group element. Our approach avoids this overhead

[7] In short, evaluation-binding means that no PPT adversary can prove distinct evaluations for a given function of the committed vector.

since, instead of including a commitment to $(x_1^2, \ldots, x_n^2)$ in the proof, we include a commitment to the reversed Hadamard product $(y_n \cdot x_n, \ldots, y_1 \cdot x_1)$ so that we only need two group elements to argue that $\sum_i y_i \cdot x_i \cdot (x_i - 1) = 0$. This allows us to reach the smallest proof length of SNARKs [67] in our proofs of smallness and valid ring LWE encryption.

We also note that the technique of [28, Section 4.1] could be used to prove that a committed vector has infinity norm $\leq B$ by showing that the polynomial $P_y(x_1, \ldots, x_n) = \sum_{i=1} y_i \cdot \prod_{j \in [-B,B]}(x_i - j)$ evaluates to 0 for a random $\boldsymbol{y} \in \mathbb{Z}_p^n$. However, the commitment size would grow with B (since it grows with the degree of the polynomial) while the proof length would grow with $\log B$. In contrast, both sizes are constant in our construction of Section D.1.

In an earlier work [83], Lipmaa and Pavlyk used the arithmetization of SNARKs [54] to construct succinct FC for sparse polynomials, where the number monomials is small w.r.t. the number n of variables. Their construction could be used as well to prove that a committed vector $(x_1, \ldots, x_n)$ satisfies $\sum_{i=1}^n y_i \cdot x_i \cdot (x_i - 1) = 0$, for a random $\boldsymbol{y} \in \mathbb{Z}_p$ derived from a random oracle. While their openings only consist of one group element, their scheme is more complex and using it in our setting would be significantly less efficient than our construction from Section 3 in other metrics. First, their commitments are larger and contain element of both sources groups $\mathbb{G}$ and $\hat{\mathbb{G}}$ (concretely, 2 elements of $\mathbb{G}$ and one element of $\hat{\mathbb{G}}$). In our applications of sections 4, D and G, this would lengthen the proofs by at least one element of $\mathbb{G}$. Also, their CRS is more complex and contains $2\nu + \mu$ elements of $\mathbb{G}$ and ν elements of $\hat{\mathbb{G}}$, where ν is the number of multiplication gates in the arithmetic circuit that computes the polynomial (which would be $\nu = 2n$ in our setting) and μ is the number of wires (here, we would have $\mu \geq 2n$). Their prover is more expensive as well and computes more than $\nu + \mu + \mu_\alpha + 2\mu_\beta$ exponentiations in $\mathbb{G}$, where μ_α and μ_α denote the lengths of private and public inputs (in our setting, this would amount to at least $7n$ exponentiations in $\mathbb{G}$). Moreover, their verification algorithm computes a product of 5 pairings (instead of 3 in Section 3) and $\mu_\beta = n$ exponentiations in both source groups.

Finally, the complex structure of their CRS would make it harder to prove knowledge-soundness in our context as it contains multiple monomials $\alpha^i y^j$ in the exponent (with $j > 1$), while "valid" commitments have components that only depend on monomials $\alpha^i y$, which have degree one in y. In the AGM, this would complicate the task of the knowledge extractor since maliciously generated commitments come with a representation that possibly depends on all group elements contained in the CRS.

B Das *et al.*'s Binarity Argument

Das *et al.* [39, Section 4.2] recently described a constant-size argument showing that a committed vector is binary. Their construction builds on KZG commitments and inherits their constant verification time. We consider the problem of building an alternative constant-size range proof using their binarity argument.

In [39, Section 4.2], a commitment to a vector $\boldsymbol{x} = (x_1, \ldots, x_n) \in \{0,1\}^n$ is obtained by committing to a polynomial $b[X]$ such that $b(\omega^i) = x_i$ for each $i \in [n]$, where ω is a primitive n-th root of unity. In asymmetric pairings, the CRS contains $(g, \hat{g}, \{g^{(\alpha^i)}, \hat{g}^{(\alpha^i)}\}_{i=1}^n)$ and a commitment is of the form $\hat{C} = \hat{g}^{b(\alpha)}$. In order to show that $b(\omega^i) \in \{0,1\}$ for each $i \in [n]$, the prover shows that the product $b[X] \cdot (1 - b[X])$ vanishes everywhere on the domain $H = \{\omega, \ldots, \omega^n\}$, which is equivalent to showing that it is divisible by the vanishing polynomial $Z_H = \prod_{i=1}^n (X - \omega^i)$.

To prove the statement, the prover has to reveal $\boldsymbol{\pi} = (C = g^{b(\alpha)}, \pi_b) \in \mathbb{G}^2$ such that $e(C, \hat{g}/\hat{C}) = e(\pi_b, \hat{g}^{Z_H(\alpha)})$. As described in [39], the construction has the same proof size as our argument system of Section 3. However, it is not zero-knowledge since the underlying KZG commitment is deterministic.

A standard trick (used in, e.g., [50]) to achieve zero-knowledge is to add a random multiple of the vanishing polynomial to the committed $b[X]$ as it does not change its evaluations on H. One can compute the initial commitment to $b[X]$ (which has degree $n-1$ since $b(\omega^i) = x_i$ for each $i \in [n]$) as $\hat{C} = \hat{g}^{b(\alpha)+r \cdot Z_H(\alpha)}$ for a random $r \xleftarrow{R} \mathbb{Z}_p$.[8] In order to generate a proof that $x_i \in \{0,1\}$ for each $i \in [n]$, the prover can compute $C = g^{b(\alpha)+s \cdot Z_H(\alpha)}$ for a random $s \xleftarrow{R} \mathbb{Z}_p$ and π_b such that $e(C, \hat{g}/\hat{C}) = e(\pi_b, \hat{g}^{Z_H(\alpha)})$ since $(b[X]+s \cdot Z_H[X]) \cdot (1-b[X]+r \cdot Z_H[X])$ is divisible by $Z_H[X]$. However, the prover has to additionally prove that $\hat{C}$ and C commit to the same $b[X]$. This can be done by introducing an additional proof component $\pi_{eq} = g^{r-s}$ that satisfies $e(C, \hat{g})/e(g, \hat{C}) = e(\pi_{eq}, \hat{g}^{Z_H(\alpha)})$ and shows that C and $\hat{C}$ are deterministic KZG commitments to polynomials $b[X] + r \cdot Z_H[X]$ and $b[X] + s \cdot Z_H[X]$ that differ by a multiple of $Z_H[X]$. If the commitment of the statement is $\hat{C}$ and the simulator knows an algebraic representation of $\hat{C}$, it can simulate a proof using α as a trapdoor.[9] In this version, each proof requires 3 elements of $\mathbb{G}$. This can be reduced to 2 elements of $\mathbb{G}$ by computing an aggregated proof $\pi_0 = \pi_b \cdot \pi_{eq}^{\delta}$ satisfying

$$\frac{e(C^{1+\delta}, \hat{g})}{e(C \cdot g^{\delta}, \hat{C})} = e(\pi_0, \hat{g}^{Z_H(\alpha)}) \tag{29}$$

where δ is obtained by hashing C and $\hat{C}$.

In order to obtain a range proof, the argument of binarity should be combined with an inner product argument showing that $\sum_{i=1}^n x_i \cdot 2^{i-1}$ is equal to the committed integer x. In [39], Das *et al.* use the observation from [8] that, if $a[X]$ and $b[X]$ are polynomials such that $a(\omega^i) = a_i$ and $b(\omega^i) = b_i$ for each $i \in [n]$,

[8] In the proof of knowledge-soundness, the AGM-enabled knowledge extractor can recover $b[X]$ from the algebraic representation of $\hat{C}$ by performing an Euclidean division (of which $b[X]$ can be interpreted as the remainder since it has degree $n-1$ while the divisor $Z_H[X]$ has degree n).

[9] If the commitment of the statement is $C \in \mathbb{G}$ and the second commitment $\hat{C}$ is part of the proof, the simulator can even simulate a proof for a given C without knowing an algebraic representation of it.

then we have

$$a[X] \cdot b[X] = q[X] \cdot Z_H[X] + X \cdot r[X] + \langle \boldsymbol{a}, \boldsymbol{b} \rangle \cdot n^{-1}$$

for unique polynomials $q[X], r[X]$ of degree $n-2$. Therefore, if $a(\omega^i) = 2^{i-1}$ for each $i \in [n]$, the prover can compute proof elements $\pi_q = g^{q(\alpha)}$, $\pi_r = g^{r(\alpha)}$ such that

$$e(C, \hat{g}^{a(\alpha)}) = e(\pi_q, \hat{g}^{Z_H(\alpha)}) \cdot e(\pi_r, \hat{g}^{\alpha}) \cdot e(g^{1/n}, \hat{g})^x \tag{30}$$

While $e(g^{1/n}, \hat{g})^x$ cannot be revealed (as it would not be zero-knowledge), the prover can compute its initial commitment to the integer $x \in [0, 2^n - 1]$ as $V = g^{x + t \cdot Z_H(\alpha)}$, for a random $t \xleftarrow{R} \mathbb{Z}_p$, which satisfies

$$e(V, \hat{g}^{1/n}) = e(g, \hat{g})^{x/n} \cdot e(g^{t/n}, \hat{g}^{Z_H(\alpha)}).$$

If we divide the latter equation out of (30), we obtain

$$\frac{e(C, \hat{g}^{a(\alpha)})}{e(V, \hat{g}^{1/n})} = e\big(\underbrace{\pi_q / g^{t/n}}_{\triangleq\ \pi'_q}, \hat{g}^{Z_H(\alpha)}\big) \cdot e(\pi_r, \hat{g}^{\alpha}) \tag{31}$$

If we now aggregate the verification equations (31) and (29), it is then possible to obtain a compressed proof $\pi = \pi_b \cdot \pi_{eq}^{\delta} \cdot {\pi'_q}^{\delta^2}$ satisfying

$$\frac{e(C, \hat{g}^{1+\delta+\delta^2 \cdot a(\alpha)})}{e(V^{\delta^2}, \hat{g}^{1/n}) \cdot e(C \cdot g^{\delta}, \hat{C})} = e\big(\pi, \hat{g}^{Z_H(\alpha)}\big) \cdot e(\pi_r, \hat{g}^{\delta^2 \cdot \alpha})$$

where $\delta = H(V, C, \hat{C}) \in \mathbb{Z}_p$. The final proof that $V = g^{x + t \cdot Z_H(\alpha)}$ commits to an integer $x \in [0, 2^n - 1]$ should contain at least the group elements $(\hat{C}, C, \pi, \pi_r)$ (which is already longer than our proofs) and cost at least 5 pairings to verify.

Moreover, in order to ensure knowledge-soundness in the AGM, the prover should also convince the verifier that V was really computed as a commitment of the form $g^{x + t \cdot Z_H(\alpha)}$ (i.e., it commits to a constant polynomial x). Since the commitment V is randomized by adding a random multiple of $Z_H(\alpha)$ in the exponent, it is not clear how this can be done using the degree check techniques of [86]. One option is to use a Schnorr-like Σ-protocol proving knowledge of the underlying (x, t), which introduces two scalars in the proof.

C Deferred Material for the Range Proof of Section 4

C.1 Proof of Correctness

The first verification equation (17) is satisfied because

$$\begin{aligned}
e(\prod_{i=1}^{n} g_{n+1-i}^{2^{i-1}}, \hat{C}) &= \prod_{i=1}^{n} e(g_{n+1-i}, \hat{C})^{2^{i-1}} \\
&= \prod_{i=1}^{n} e\big(g_{n+1-i}, \hat{g}^{\gamma} \cdot \prod_{j=1}^{n} \hat{g}_j^{x_j}\big)^{2^{i-1}} = \prod_{i=1}^{n} e\big(g^{\gamma} \cdot \prod_{j=1}^{n} g_j^{x_j}, \hat{g}_{n+1-i}\big)^{2^{i-1}}, \\
&= \prod_{i=1}^{n} e\big(g_{n+1-i}^{\gamma} \cdot \prod_{j=1}^{n} g_{n+1+j-i}^{x_j}, \hat{g}\big)^{2^{i-1}} \\
&= e(g_1, \hat{g}_n)^{\sum_{i=1}^{n} x_i \cdot 2^{i-1}} \cdot e\Big(\prod_{i=1}^{n} \big(g_{n+1-i}^{\gamma} \cdot \prod_{j \in [n] \setminus \{i\}} g_{n+1-i+j}^{x_j}\big)^{2^{i-1}}, \hat{g}\Big)
\end{aligned}$$

and $e(g_n, \hat{V}) = e(g_1, \hat{g}_n)^{x} \cdot e(g_n^{r}, \hat{g})$, so that dividing out the two equations yields $e(\prod_{i=1}^{n} g_{n+1-i}^{2^{i-1}}, \hat{C}) / e(g_n, \hat{V}) = e(\pi_x, \hat{g})$ when $x = \sum_{i=1}^{n} x_i \cdot 2^{i-1}$.

Similarly, the second verification equation (18) follows by dividing the following two equalities:

$$\begin{aligned}
e(\prod_{i=1}^{n} g_{n+1-i}^{t_i \cdot y_i}, \hat{C}) &= \prod_{i=1}^{n} e(g_{n+1-i}, \hat{C})^{t_i \cdot y_i} \\
&= \prod_{i=1}^{n} \Big(e(g_1, \hat{g}_n)^{x_i} \cdot e\big(g, \hat{g}_{n+1-i}^{\gamma} \cdot \prod_{j \in [n] \setminus \{i\}} \hat{g}_{n+1+j-i}^{x_j}\big)\Big)^{t_i \cdot y_i} \\
&= \prod_{i=1}^{n} \Big(e(g_1, \hat{g}_n)^{x_i} \cdot e\big(g_{n+1-i}^{\gamma} \cdot \prod_{j \in [n] \setminus \{i\}} g_{n+1+j-i}^{x_j}, \hat{g}\big)\Big)^{t_i \cdot y_i} \\
&= e(g_1, \hat{g}_n)^{\sum_{i=1}^{n} y_i \cdot t_i \cdot x_i} \\
&\qquad \cdot e\Big(\prod_{i=1}^{n} \Big(g_{n+1-i}^{\gamma} \cdot \prod_{j \in [n] \setminus \{i\}} g_{n+1-i+j}^{x_j}\Big)^{t_i \cdot y_i}, \hat{g}\Big)
\end{aligned}$$

and

$$\begin{aligned}
e\big(C_y, \prod_{i=1}^{n} \hat{g}_i^{t_i}\big) &= \prod_{i=1}^{n} e(C_y, \hat{g}_i)^{t_i} = \prod_{i=1}^{n} e\big(g^{\gamma_y} \cdot \prod_{j=1}^{n} g_{n+1-j}^{y_j \cdot x_j}, \hat{g}_i\big)^{t_i} \\
&= \prod_{i=1}^{n} e\big(g_i^{\gamma_y} \cdot \prod_{j=1}^{n} g_{n+1-j+i}^{y_j \cdot x_j}, \hat{g}\big)^{t_i} \\
&= e(g_1, \hat{g}_n)^{\sum_{i=1}^{n} y_i \cdot t_i \cdot x_i} \cdot e\Big(\prod_{i=1}^{n} \big(g_i^{\gamma_y} \cdot \prod_{j \in [n] \setminus \{i\}} g_{n+1-j+i}^{y_j \cdot x_j}\big)^{t_i}, \hat{g}\Big).
\end{aligned}$$

As for equation (19), we have

$$
\begin{aligned}
e(C_y \cdot \prod_{j=1}^{n} g_{n+1-j}^{-y_j}, \hat{C}) &= e\Big(g^{\gamma_y} \cdot \prod_{j=1}^{n} g_{n+1-j}^{y_j \cdot (x_j - 1)}, \hat{g}^{\gamma} \cdot \prod_{i=1}^{n} \hat{g}_i^{x_i}\Big) \\
&= e(C_y \cdot \prod_{j=1}^{n} g_{n+1-j}^{-y_j}, \hat{g})^{\gamma} \cdot \prod_{i=1}^{n} e\Big(\big(g_i^{\gamma_y} \cdot \prod_{j=1}^{n} g_{n+1-j+i}^{y_j \cdot (x_j - 1)}\big)^{x_i}, \hat{g}\Big) \\
&= e(g_1, \hat{g}_n)^{\sum_{i=1}^{n} y_i \cdot x_i \cdot (x_i - 1)} \cdot e\big(g^{\gamma \cdot \gamma_y} \cdot \prod_{j=1}^{n} g_{n+1-j}^{\gamma \cdot y_j \cdot (x_j - 1)}, \hat{g}\big) \\
&\qquad \cdot \prod_{i=1}^{n} e\Big(\big(g_i^{\gamma_y} \cdot \prod_{j \in [n] \setminus \{i\}} g_{n+1-j+i}^{y_j \cdot (x_j - 1)}\big)^{x_i}, \hat{g}\Big) \\
&= e\Big(g^{\gamma \cdot \gamma_y} \cdot \prod_{j=1}^{n} g_{n+1-j}^{\gamma \cdot y_j \cdot (x_j - 1)} \cdot \prod_{i=1}^{n} \big(g_i^{\gamma_y} \cdot \prod_{j \in [n] \setminus \{i\}} g_{n+1-j+i}^{y_j \cdot (x_j - 1)}\big)^{x_i}, \hat{g}\Big)
\end{aligned}
$$

where the last equality holds because $x_i(x_i - 1) = 0$ for each $i \in [n]$.

Equation (20) is satisfied by $\pi_v = \prod_{i=2}^{n} \big(g_{n+1-i}^{r} \cdot g_{n+2-i}^{x}\big)^{s_i} \in \mathbb{G}$ since

$$
\begin{aligned}
e\big(\prod_{i=2}^{n} g_{n+1-i}^{s_i}, \hat{V}\big) &= \prod_{i=2}^{n} e\big(g_{n+1-i}, \hat{g}^{r} \cdot \hat{g}_1^{x}\big)^{s_i} \\
&= \prod_{i=2}^{n} e\big(g^{r} \cdot g_1^{x}, \hat{g}_{n+1-i}\big)^{s_i} = \prod_{i=2}^{n} e\big(g_{n+1-i}^{r} \cdot g_{n+2-i}^{x}, \hat{g}\big)^{s_i} = e(\pi_v, \hat{g}).
\end{aligned}
$$

C.2 Proof of Theorem 1

Proof. We describe a simulator that perfectly simulates proofs using a trapdoor $\mathsf{tk} = \alpha \in \mathbb{Z}_p$ and by programming the random oracles. Given a commitment $\hat{V} \in \hat{\mathbb{G}}$, the simulator computes $C_y = g^{\theta_y} \in \mathbb{G}$ for a randomly chosen $\theta_y \xleftarrow{R} \mathbb{Z}_p$. Next, it obtains $s_i = H_s(i, [2, n], \hat{V}, \hat{C}, C_y)$ for each index $i \in [2, n]$. It then uniformly chooses $\boldsymbol{y} = (y_1, \ldots, y_n) \xleftarrow{R} \mathbb{Z}_p^n$, $\boldsymbol{t} = (t_1, \ldots, t_n) \xleftarrow{R} \mathbb{Z}_p^n$, $\boldsymbol{\delta} = (\delta_x, \delta_{eq}, \delta_y, \delta_v) \xleftarrow{R} \mathbb{Z}_p^4$, sets $s_0 = 0$, and computes

$$
\lambda = \frac{\delta_x \cdot (\alpha^n) - \sum_{i=2}^{n} \delta_v \cdot s_i \cdot (\alpha^{n+1-i})}{\theta_y \cdot \delta_y + \sum_{i=1}^{n} (\delta_{x,i} \cdot 2^{i-1} + (\delta_{eq} \cdot t_i - \delta_y) \cdot y_i + \delta_0 \cdot u_i) \cdot (\alpha^{n+1-i})},
$$

where $\delta_{x,i} = \delta_x$ if $i \in [\ell]$ and $\delta_{x,i} = 0$ for all $i \in [\ell + 1, n]$.

Note that the denominator is uniformly distributed over $\mathbb{Z}_p$ and non-zero with probability $1 - 1/p$. Then, it chooses $\gamma \xleftarrow{R} \mathbb{Z}_p$ and computes a commitment $\hat{C} = \hat{V}^{\lambda} \cdot \hat{g}^{\gamma}$. It aborts if $\boldsymbol{y} = H(\hat{V}, \hat{C})$, or $H_{\mathrm{agg}}(\hat{V}, \hat{C}, C_y)$ or $H_t(\boldsymbol{y}, \hat{V}, \hat{C}, C_y)$ was already defined. If the simulator does not fail, it computes

$$
\pi = \Big(C_y^{\delta_y} \cdot \prod_{i=1}^{n} g_{n+1-i}^{\delta_{x,i} \cdot 2^{i-1} + (\delta_{eq} \cdot t_i - \delta_y) y_i}\Big)^{\gamma} \cdot \Big(\prod_{i=1}^{n} \hat{g}_i^{\delta_{eq} \cdot t_i}\Big)^{-\theta_y}. \tag{32}
$$

Then, it programs the random oracles to have $H_{\text{agg}}(\hat{V}, \hat{C}, C_y) = (\delta_x, \delta_{eq}, \delta_y, \delta_v)$, $\boldsymbol{y} = H(\hat{V}, \hat{C})$, $\boldsymbol{t} = H_t(\boldsymbol{y}, \hat{V}, \hat{C}, C_y)$ for each $i \in [n]$. This provides a valid proof $\boldsymbol{\pi} = (\hat{C}, C_y, \pi)$, where $\hat{C}$ and C_y are uniformly distributed over $\mathbb{G}$ and $\hat{\mathbb{G}}$, respectively. Moreover, π satisfies the verification equation (22) since we have

$$g_n^{\delta_x} \cdot \prod_{i=2}^{n} g_{n+1-i}^{-\delta_v \cdot s_i} = \Big(C_y^{\delta_y} \cdot \prod_{i=1}^{n} g_{n+1-i}^{\delta_{x,i} \cdot 2^{i-1} + (\delta_{eq} t_i - \delta_y) y_i} \Big)^{\lambda},$$

which implies

$$\begin{aligned}
&\frac{e\big(C_y^{\delta_y} \cdot \prod_{i=1}^{n} g_{n+1-i}^{\delta_{x,i} \cdot 2^{i-1} + (\delta_{eq} \cdot t_i - \delta_y) \cdot y_i}, \hat{C}\big)}{e\big(g_n^{\delta_x} \cdot \prod_{i=2}^{n} g_{n+1-i}^{-\delta_v \cdot s_i}, \hat{V}\big) \cdot e\big(C_y, \prod_{i=1}^{n} \hat{g}_i^{\delta_{eq} \cdot t_i}\big)} \\
&\qquad = \frac{e\big(C_y^{\delta_y} \cdot \prod_{i=1}^{n} g_{n+1-i}^{\delta_{x,i} \cdot 2^{i-1} + (\delta_{eq} \cdot t_i - \delta_y) \cdot y_i},\ \hat{V}^{\lambda} \cdot \hat{g}^{\gamma}\big)}{e\big(g_n^{\delta_x} \cdot \prod_{i=2}^{n} g_{n+1-i}^{-\delta_v \cdot s_i}, \hat{V}\big) \cdot e\big(C_y, \prod_{i=1}^{n} \hat{g}_i^{\delta_{eq} \cdot t_i}\big)} \\
&\qquad = \frac{e\big(C_y^{\delta_y} \cdot \prod_{i=1}^{n} g_{n+1-i}^{\delta_{x,i} \cdot 2^{i-1} + (\delta_{eq} \cdot t_i - \delta_y) \cdot y_i}, \hat{V}^{\lambda}\big)}{e\big(g_n^{\delta_x} \cdot \prod_{i=2}^{n} g_{n+1-i}^{-\delta_v \cdot s_i},\ \hat{V}\big)} \\
&\qquad\qquad \cdot \frac{e\big(\big(C_y^{\delta_y} \cdot \prod_{i=1}^{n} g_{n+1-i}^{\delta_{x,i} \cdot 2^{i-1} + (\delta_{eq} \cdot t_i - \delta_y) y_i}\big)^{\gamma}, \hat{g}\big)}{e\big(\big(\prod_{i=1}^{n} \hat{g}_i^{\delta_{eq} \cdot t_i}\big)^{\theta_y}, \hat{g}\big)} = e(\pi, \hat{g})
\end{aligned}$$

□

D Short Proofs that a Committed Vector is Small

We now show that the range proof of Section 4 can be batched in order to simultaneously prove possibly distinct ranges for the different slots of a multi-base Pedersen commitment. In particular, we can prove that a vector commitment commits to a vector of small infinity norm.

As explained in Supplementary Material E, the construction can be used to prove that a committed vector has small Euclidean norm.

D.1 Description

Given a commitment $\hat{V} = \hat{g}^r \cdot \prod_{k=1}^{m} \hat{g}_k^{x_k}$, the prover will convince the verifier that $x_k \in [0, 2^{\ell_k} - 1]$ for each $k \in [m]$ using only 3 group elements. The relation is now defined as

$$\begin{aligned}
\mathcal{R}_{\mathsf{pp}} = \Big\{ (\mathbf{x}, \mathbf{w}) = \Big(\big(\hat{V} = \hat{g}^r \cdot \prod_{k=1}^{m} \hat{g}_k^{x_k}, (\ell_1, \ldots, \ell_m)\big) \in \hat{\mathbb{G}} \times \mathbb{N}^m, \\
(r, x_1, \ldots, x_m) \in \mathbb{Z}_p \times \prod_{k=1}^{m} [0, 2^{\ell_k} - 1] \Big) \Big\}
\end{aligned}$$

and the construction proceeds as follows.

CRS-Gen$(1^\lambda, 1^m, \{1^{n_k}\}_{k=1}^m)$**:** On input of a security parameter λ, a number of slots $m \in \mathsf{poly}(\lambda)$, and the maximal bitlength $n_k \in \mathsf{poly}(\lambda)$ of ranges for each slot $k \in [m]$, set $n = \sum_{k=1}^m n_k$ and do the following:

1. Choose asymmetric bilinear groups $(\mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T)$ of prime order $p > 2^{\ell_p}$, where $\ell_p = \max(l(\lambda), n_1, \ldots, n_m)$ for some polynomial $l : \mathbb{N} \to \mathbb{N}$, and generators $g \xleftarrow{R} \mathbb{G}$, $\hat{g} \xleftarrow{R} \hat{\mathbb{G}}$.
2. Pick a random $\alpha \xleftarrow{R} \mathbb{Z}_p$ and compute $g_1, \ldots, g_n, g_{n+2}, \ldots, g_{2n} \in \mathbb{G}$ as well as $\hat{g}_1, \ldots, \hat{g}_n \in \hat{\mathbb{G}}$, where $g_i = g^{(\alpha^i)}$ for each $i \in [2n] \setminus \{n+1\}$ and $\hat{g}_i = \hat{g}^{(\alpha^i)}$ for each $i \in [n]$.
3. Choose hash functions $H, H_t : \{0,1\}^* \to \mathbb{Z}_p^n$, $H_s : \{0,1\}^* \to \mathbb{Z}_p$, $H_{\text{agg}} : \{0,1\}^* \to \mathbb{Z}_p^4$ and $H_\xi : \{0,1\}^* \to \mathbb{Z}_p^m$ modeled as random oracles.

The public parameters are defined to be

$$\mathsf{pp} = \Big((\mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T), n, g, \hat{g}, \{g_i\}_{i \in [2n] \setminus \{n+1\}}, \{\hat{g}_i\}_{i \in [n]}, \mathbf{H}\Big)$$

where $\mathbf{H} = \{H, H_s, H_t, H_{\text{agg}}, H_\xi\}$ are hash functions.

Com$_{\mathsf{pp}}(x)$ To commit to a vector of integers $\boldsymbol{x} = (x_1, \ldots, x_m) \in \mathbb{Z}^m$, choose a random $r \xleftarrow{R} \mathbb{Z}_p$ and compute $\hat{V} = \hat{g}^r \cdot \prod_{k=1}^m \hat{g}_k^{x_k} \in \hat{\mathbb{G}}$. Return the commitment $\mathsf{com} = \hat{V} \in \hat{\mathbb{G}}$ and the opening information $\mathsf{aux} = r \in \mathbb{Z}_p$.

Prove$_{\mathsf{pp}}\big((\mathsf{com}, \{1^{\ell_k}\}_{k=1}^m), (\boldsymbol{x}, \mathsf{aux})\big)$**:** given a commitment $\mathsf{com} = \hat{V}$ and witnesses $\big(\boldsymbol{x}; \mathsf{aux}\big)$ consisting of an integer vector $\boldsymbol{x} = (x_1, \ldots, x_m) \in \mathbb{Z}^m$ such that $x_k \in [0, 2^{\ell_k} - 1]$ for each $k \in [m]$, where $\ell_k \leq n_k$, and $\mathsf{aux} = r \in \mathbb{Z}_p$ is the randomness such that $\hat{V} = \hat{g}^r \cdot \prod_{k=1}^m \hat{g}_k^{x_k}$, do the following:

1. For each $k \in [m-1]$, set $j_k = n_1 + \cdots + n_{k-1}$ with $j_1 = 0$. For each $k \in [m]$, let the binary expansion $(x_{k,1}, \ldots, x_{k,\ell_k}) \in \{0,1\}^{\ell_k}$ of x_k. Set $x_{k,i} = 0$ for each $i \in [\ell_k + 1, n_k]$ and define $\bar{\boldsymbol{x}}_k = (x_{k,1}, \ldots, x_{k,n_k}) \in \{0,1\}^{n_k}$.
2. Choose $\gamma \xleftarrow{R} \mathbb{Z}_p$ and compute

$$\hat{C} = \hat{g}^\gamma \cdot \prod_{k=1}^m \prod_{j=1}^{\ell_k} \hat{g}_{j_k+j}^{x_{k,j}}.$$

Compute $\boldsymbol{y} = (y_1, \ldots, y_n) = H(\hat{V}, \hat{C}) \in \mathbb{Z}_p^n$. Then, choose $\gamma_y \xleftarrow{R} \mathbb{Z}_p$ and compute

$$C_y = g^{\gamma_y} \cdot \prod_{k=1}^m \prod_{j=1}^{\ell_k} g_{n+1-(j_k+j)}^{y_{j_k+j} \cdot x_{k,j}}$$

3. Compute $\boldsymbol{\xi} = (\xi_1, \ldots, \xi_m) = H_\xi(\hat{V}, \hat{C}, C_y)$ and generate a proof π_x that $\hat{C}$ commits to $(\bar{\boldsymbol{x}}_1 \mid \ldots \mid \bar{\boldsymbol{x}}_m) \in \mathbb{Z}_p^n$ such that $\sum_{i=1}^{\ell_k} x_{k,i} \cdot 2^{i-1} = x_k$ for each $k \in [m]$. This proof $\pi_x \in \mathbb{G}$ satisfies

$$\frac{e\big(\prod_{k=1}^m \big(\prod_{i=1}^{\ell_k} g_{n+1-(j_k+i)}^{2^{i-1}}\big)^{\xi_k}, \hat{C}\big)}{e\big(\prod_{k=1}^m g_{n+1-k}^{\xi_k}, \hat{V}\big)} = e(\pi_x, \hat{g}) \tag{33}$$

and is obtained as per formula (40) in Supplementary Material D.2.

4. Compute $\boldsymbol{t} = (t_1, \ldots, t_n) = H_t(\boldsymbol{y}, \hat{V}, \hat{C}, C_y) \in \mathbb{Z}_p^n$. Generate a proof π_{eq} (as per formula (43) in Supplementary Material D.2) satisfying

$$\frac{e\big(\prod_{k=1}^{m}\prod_{i=1}^{n_k} g_{n+1-(j_k+i)}^{t_{j_k+i}\cdot y_{j_k+i}}, \hat{C}\big)}{e\big(C_y, \prod_{k=1}^{m}\prod_{i=1}^{n_k} \hat{g}_{j_k+i}^{t_{j_k+i}}\big)} = e(\pi_{eq}, \hat{g}), \tag{34}$$

which shows that C_y is consistent with $\boldsymbol{y}$ and $\hat{C}$.

5. Prove that $\sum_{k=1}^{m}\sum_{i=1}^{n_k} y_{j_k+i}\cdot x_{k,i}\cdot(x_{k,i}-1) = 0$ by computing $\pi_y \in \mathbb{G}$ (as specified by (44) in Supplementary Material D.2) satisfying

$$e\big(C_y \cdot \prod_{k=1}^{m}\prod_{i=1}^{n_k} g_{n+1-(j_k+i)}^{-y_{j_k+i}}, \hat{C}\big) = e(\pi_y, \hat{g}). \tag{35}$$

6. Generate an aggregated proof that $\hat{V} = \hat{g}^r \cdot \prod_{k=1}^{m} \hat{g}_k^{x_k}$ is a commitment to a vector that contains 0 in its last $n-m$ coordinates. Namely, compute $\pi_v = \prod_{i=m+1}^{n}\big(g_{n+1-i}^{r}\cdot\prod_{k=1}^{m} g_{n+2-i+k}^{x_k}\big)^{s_i} \in \mathbb{G}$ such that

$$e\Big(\prod_{i=m+1}^{n} g_{n+1-i}^{s_i}, \hat{V}\Big) = e(\pi_v, \hat{g}). \tag{36}$$

where $s_i = H_s(i, [m+1, n], \hat{V}, \hat{C}, C_y) \in \mathbb{Z}_p$ for each $i \in [m+1, n]$.

7. Compute $(\delta_x, \delta_{eq}, \delta_y, \delta_v) = H_{\text{agg}}(\hat{V}, \hat{C}, C_y) \in \mathbb{Z}_p^4$ and compute an aggregated proof

$$\pi = \pi_x^{\delta_x}\cdot\pi_y^{\delta_y}\cdot\pi_{eq}^{\delta_{eq}}\cdot\pi_v^{\delta_v}.$$

Output the final range argument which consists of

$$\boldsymbol{\pi} := \big(\hat{C}, C_y, \pi\big). \tag{37}$$

$\mathsf{Verify}_{\mathsf{pp}}\big(\mathsf{com}, \boldsymbol{\pi}\big)$: Given a commitment $\mathsf{com} = \hat{V} \in \hat{\mathbb{G}}$ and a candidate proof $\boldsymbol{\pi}$, parse the latter as in (37).

1. Compute $(\delta_x, \delta_{eq}, \delta_y, \delta_v) = H_{\text{agg}}(\hat{V}, \hat{C}, C_y) \in \mathbb{Z}_p^4$, $\boldsymbol{y} = H(\hat{V}, \hat{C}) \in \mathbb{Z}_p^n$, $\boldsymbol{\xi} = (\xi_1, \ldots, \xi_m) = H_\xi(\hat{V}, \hat{C}, C_y) \in \mathbb{Z}_p^m$, $\boldsymbol{t} = H_t(\boldsymbol{y}, \hat{V}, \hat{C}, C_y) \in \mathbb{Z}_p^n$, and $s_i = H_s(i, [m+1, n], \hat{V}, \hat{C}, C)$ for all indices $i \in [m+1, n]$. Define the vector $(s_1, \ldots, s_n) = (\mathbf{0}^m, s_{m+1}, \ldots, s_n)$.
2. For each $k \in [m]$, define $\delta_{x,k,i} = \delta_x$ if $i \in [\ell_k]$ and $\delta_{x,k,i} = 0$ if $i \in [\ell_k + 1, n_k]$. Return 1 if

$$e\Big(C_y^{\delta_y}\prod_{k=1}^{m}\prod_{i=1}^{n_k} g_{n+1-(j_k+i)}^{\xi_k\cdot\delta_{x,k,i}\cdot 2^{i-1}+(\delta_{eq}\cdot t_{j_k+i}-\delta_y)\cdot y_{j_k+i}}, \hat{C}\Big)$$
$$\cdot\, e\Big(\prod_{k=1}^{m} g_{n+1-k}^{\xi_k}\prod_{i=m+1}^{n} g_{n+1-i}^{-\delta_v\cdot s_i}, \hat{V}\Big)^{-1}\cdot e\Big(C_y, \prod_{k=1}^{m}\prod_{i=1}^{n_k}\hat{g}_{j_k+i}^{\delta_{eq}\cdot t_{j_k+i}}\Big)^{-1} = e(\pi, \hat{g}) \tag{38}$$

and 0 otherwise.

The correctness of the scheme can be proven in the same way as in the base scheme of Section 4 and details are given in Supplementary Material D.2.

Efficiency. The proof size remains the same as in Section 4. If m denotes the number of simultaneously performed range proofs, the computational cost of the prover is now dominated by $3n$ exponentiations in $\mathbb{G}$ and $m+1$ exponentiations in $\hat{\mathbb{G}}$ since each subset product $\prod_{j=1}^{n_k} \hat{g}_{j_k+j}^{x_{k,j}}$ is cheaper to compute than an exponentiation in $\hat{\mathbb{G}}$ (recall that $n_k < \log p$).

The verifier's workload amounts to $2n+1$ exponentiations in $\mathbb{G}$, n exponentiations in $\hat{\mathbb{G}}$ and 4 pairings.

D.2 Correctness of Aggregated Range Proofs

The verification equation (38) is obtained by raising equalities (33), (34), (35) and (36) to the powers δ_x, δ_{eq}, δ_y and δ_v, and multiplying them together. We now consider the generation of proofs for individual verification equations (33)-(36).

The prover can compute $\pi_x \in \mathbb{G}$ satisfying equation (33) because, for each $k \in [m]$, we have

$$
\begin{aligned}
e\Big(\prod_{i=1}^{\ell_k} g_{n+1-(j_k+i)}^{2^{i-1}}\,,\,\hat{C}\Big) &= \prod_{i=1}^{\ell_k} e\big(g_{n+1-(j_k+i)},\hat{C}\big)^{2^{i-1}} \\
&= \prod_{i=1}^{\ell_k} e\Big(g_{n+1-(j_k+i)},\hat{g}^{\gamma}\cdot\prod_{\kappa=1}^{m}\prod_{j=1}^{\ell_\kappa}\hat{g}_{j_\kappa+j}^{x_{\kappa,j}}\Big)^{2^{i-1}} \\
&= \prod_{i=1}^{\ell_k} e\Big(g^{\gamma}\cdot\prod_{\kappa=1}^{m}\prod_{j=1}^{\ell_\kappa} g_{j_\kappa+j}^{x_{j_\kappa+j}},\hat{g}_{n+1-(j_k+i)}\Big)^{2^{i-1}}, \\
&= \prod_{i=1}^{\ell_k} e\Big(g_{n+1-(j_k+i)}^{\gamma}\cdot\prod_{\kappa=1}^{m}\prod_{j=1}^{\ell_k} g_{n+1+(j_\kappa+j)-(j_k+i)}^{x_{\kappa,j}},\hat{g}\Big)^{2^{i-1}} \\
&= e(g_1,\hat{g}_n)^{\sum_{i=1}^{\ell_k} x_{k,i}\cdot 2^{i-1}} \\
&\quad\cdot e\Big(\prod_{i=1}^{\ell_k}\big(g_{n+1-(j_k+i)}^{\gamma}\cdot\prod_{\kappa=1}^{m}\prod_{\substack{j\in[\ell_\kappa]\\(\kappa,j)\neq(k,i)}} g_{n+1-(j_k+i)+(j_\kappa+j)}^{x_{\kappa,j}}\big)^{2^{i-1}},\hat{g}\Big)
\end{aligned}
$$

and thus

$$
\begin{aligned}
\prod_{k=1}^{m} e\big(\prod_{i=1}^{\ell_k} g_{n+1-(j_k+i)}^{2^{i-1}},\hat{C}\big)^{\xi_k} &= e(g_1,\hat{g}_n)^{\sum_{k=1}^{m}\xi_k\cdot(\sum_{i=1}^{\ell_k} x_{k,i}\cdot 2^{i-1})} \\
&\quad\cdot e\Big(\prod_{k=1}^{m}\Big(\prod_{i=1}^{\ell_k}\big(g_{n+1-(j_k+i)}^{\gamma}\cdot\prod_{\kappa=1}^{m}\prod_{\substack{j\in[\ell_\kappa]\\(\kappa,j)\neq(k,i)}} g_{n+1-(j_k+i)+(j_\kappa+j)}^{x_{\kappa,j}}\big)^{2^{i-1}}\Big)^{\xi_k},\hat{g}\Big).
\end{aligned}
\tag{39}
$$

We also have

$$e(\prod_{k=1}^{m} g_{n+1-k}^{\xi_k}, \hat{V}) = e(g_1, \hat{g}_n)^{\xi_k \cdot x_k} \cdot e\Big(\prod_{k=1}^{m} \big(g_{n+1-k}^{r} \cdot \prod_{i=1, i\neq k}^{m} g_{n+1+i-k}^{x_i}\big)^{\xi_k}, \hat{g}\Big),$$

so that dividing the latter from (39) yields (33) when $x_k = \sum_{i=1}^{n} x_{k,i} \cdot 2^{i-1}$ for each $k \in [m]$ and

$$\pi_x = \frac{\prod_{k=1}^{m} \Big(\prod_{i=1}^{\ell_k} \big(g_{n+1-(j_k+i)}^{\gamma} \cdot \prod_{\kappa=1}^{m} \prod_{j\in[\ell_\kappa],(\kappa,j)\neq(k,i)} g_{n+1-(j_k+i)+(j_\kappa+j)}^{x_{\kappa,j}}\big)^{2^{i-1}}\Big)^{\xi_k}}{\prod_{k=1}^{m} \big(g_{n+1-k}^{r} \cdot \prod_{i=1,i\neq k}^{m} g_{n+1+i-k}^{x_i}\big)^{\xi_k}}. \tag{40}$$

The prover can also compute π_{eq} satisfying the second verification equation (34) by observing that, for each $k \in [m]$, we have

$$\begin{aligned}
e\big(\prod_{i=1}^{n_k} g_{n+1-(j_k+i)}^{t_{j_k+i}\cdot y_{j_k+i}}, \hat{C}\big) &= \prod_{i=1}^{n_k} e(g_{n+1-(j_k+i)}, \hat{C})^{t_{j_k+i}\cdot y_{j_k+i}} \qquad (41)\\
&= \prod_{i=1}^{n_k} e\big(g_{n+1-(j_k+i)}, \hat{g}^{\gamma} \prod_{\kappa=1}^{m}\prod_{j=1}^{n_\kappa} \hat{g}_{j_\kappa+j}^{x_{\kappa,j}}\big)^{t_{j_k+i}\cdot y_{j_k+i}}\\
&= \prod_{i=1}^{n_k} \Big(e(g_1, \hat{g}_n)^{x_{k,i}} \cdot e\big(g, \hat{g}_{n+1-(j_k+i)}^{\gamma} \cdot \prod_{\kappa=1}^{m} \prod_{\substack{j\in[n_\kappa]\\(\kappa,j)\neq(k,i)}} \hat{g}_{n+1+(j_\kappa+j)-(j_k+i)}^{x_{\kappa,j}}\big)\Big)^{t_{j_k+i}\cdot y_{j_k+i}}\\
&= \prod_{i=1}^{n_k} \Big(e(g_1, \hat{g}_n)^{x_{k,i}} \cdot e\big(g_{n+1-(j_k+i)}^{\gamma} \cdot \prod_{\kappa=1}^{m} \prod_{\substack{j\in[n_\kappa]\\(\kappa,j)\neq(k,i)}} g_{n+1+(j_\kappa+j)-(j_k+i)}^{x_{\kappa,j}}, \hat{g}\big)\Big)^{t_{j_k+i}\cdot y_{j_k+i}}\\
&= e(g_1, \hat{g}_n)^{\sum_{i=1}^{\ell_k} y_{j_k+i}\cdot t_{j_k+i}\cdot x_{k,i}}\\
&\qquad \cdot e\Big(\prod_{i=1}^{n_k} \Big(g_{n+1-(j_k+i)}^{\gamma} \cdot \prod_{\kappa=1}^{m} \prod_{\substack{j\in[n_\kappa]\\(\kappa,j)\neq(k,i)}} g_{n+1+(j_\kappa+j)-(j_k+i)}^{x_{\kappa,j}}\Big)^{t_{j_k+i}\cdot y_{j_k+i}}, \hat{g}\Big).
\end{aligned}$$

We also have

$$
\begin{aligned}
e\big(C_y, \prod_{k=1}^{m}\prod_{i=1}^{n_k} \hat{g}_{j_k+i}^{t_{j_k+i}}\big) &= \prod_{k=1}^{m}\prod_{i=1}^{n_k} e(C_y, \hat{g}_{j_k+i})^{t_{j_k+i}} \qquad (42)\\
&= \prod_{k=1}^{m}\prod_{i=1}^{n_k} e\big(g^{\gamma_y} \cdot \prod_{\kappa=1}^{m}\prod_{j=1}^{n_\kappa} g_{n+1-(j_\kappa+j)}^{y_{j_\kappa+j}\cdot x_{\kappa,j}}, \hat{g}_{j_k+i}\big)^{t_{j_k+i}}\\
&= \prod_{k=1}^{m}\prod_{i=1}^{n_k} e\big(g_{j_k+i}^{\gamma_y} \cdot \prod_{\kappa=1}^{m}\prod_{j=1}^{n_\kappa} g_{n+1-(j_\kappa+j)+(j_k+i)}^{y_{j_\kappa+j}\cdot x_{\kappa,j}}, \hat{g}\big)^{t_{j_k+i}}\\
&= e(g_1, \hat{g}_n)^{\sum_{k=1}^{m}\sum_{i=1}^{\ell_k} y_{j_k+i}\cdot t_{j_k+i}\cdot x_{k,i}}\\
&\quad \cdot e\Big(\prod_{k=1}^{m}\prod_{i=1}^{n_k}\big(g_{j_k+i}^{\gamma_y} \cdot \prod_{\kappa=1}^{m}\prod_{\substack{j\in[n_\kappa]\\(\kappa,j)\neq(k,i)}} g_{n+1-(j_\kappa+j)+(j_k+i)}^{y_{j_\kappa+j}\cdot x_{\kappa,j}}\big)^{t_{j_k+i}}, \hat{g}\Big).
\end{aligned}
$$

By taking the product of (41) for all $k \in [m]$ and dividing (42) out of the result, we obtain (34) when π_{eq} is computed as

$$
\pi_{eq} = \frac{\prod_{k=1}^{m}\prod_{i=1}^{n_k}\Big(g_{n+1-(j_k+i)}^{\gamma} \cdot \prod_{\kappa=1}^{m}\prod_{j\in[n_\kappa],(\kappa,j)\neq(k,i)} g_{n+1+(j_\kappa+j)-(j_k+i)}^{x_{\kappa,j}}\Big)^{t_{j_k+i}\cdot y_{j_k+i}}}{\prod_{k=1}^{m}\prod_{i=1}^{n_k}\big(g_{j_k+i}^{\gamma_y} \cdot \prod_{\kappa=1}^{m}\prod_{j\in[n_\kappa],(\kappa,j)\neq(k,i)} g_{n+1-(j_\kappa+j)+(j_k+i)}^{y_{j_\kappa+j}\cdot x_{\kappa,j}}\big)^{t_{j_k+i}}} \qquad (43)
$$

As for equation (35), the prover can compute

$$
\begin{aligned}
\pi_y = g^{\gamma\cdot\gamma_y} &\cdot \prod_{k=1}^{m}\prod_{i=1}^{n_k} g_{n+1-(j_k+i)}^{\gamma\cdot y_{j_k+i}\cdot(x_{k,i}-1)}\\
&\cdot \prod_{\kappa=1}^{m}\prod_{j=1}^{n_\kappa}\big(g_{j_\kappa+j}^{\gamma_y} \cdot \prod_{k=1}^{m}\prod_{\substack{i\in[n_k]\\(k,i)\neq(\kappa,j)}} g_{n+1-(j_k+i)+(j_\kappa+j)}^{y_{j_k+i}\cdot(x_{k,i}-1)}\big)^{x_{\kappa,j}}, \qquad (44)
\end{aligned}
$$

which satisfies (35) because we have

$$
\begin{aligned}
e(C_y \cdot \prod_{k=1}^{m}\prod_{i=1}^{n_k} g_{n+1-(j_k+i)}^{-y_{j_k+i}}, \hat{C}) &= e\Big(g^{\gamma_y} \cdot \prod_{k=1}^{m}\prod_{i=1}^{n_k} g_{n+1-(j_k+i)}^{y_{j_k+i}\cdot(x_{k,i}-1)}, \hat{g}^{\gamma} \cdot \prod_{\kappa=1}^{m}\prod_{j=1}^{n_\kappa} \hat{g}_{j_\kappa+j}^{x_{\kappa,j}}\Big) \\
&= e(g^{\gamma_y} \cdot \prod_{k=1}^{m}\prod_{i=1}^{n_k} g_{n+1-(j_k+i)}^{y_{j_k+i}\cdot(x_{k,i}-1)}, \hat{g})^{\gamma} \\
&\quad \cdot \prod_{\kappa=1}^{m}\prod_{j=1}^{n_\kappa} e\Big(\big(g_{j_\kappa+j}^{\gamma_y} \cdot \prod_{k=1}^{m}\prod_{i=1}^{n_k} g_{n+1-(j_k+i)+(j_\kappa+j)}^{y_{j_k+i}\cdot(x_{k,i}-1)}\big)^{x_{\kappa,j}}, \hat{g}\Big) \\
&= e(g_1, \hat{g}_n)^{\sum_{k=1}^{m}\sum_{i=1}^{n_k} y_{j_k+i}\cdot x_{k,i}\cdot(x_{k,i}-1)} \cdot e(g^{\gamma\cdot\gamma_y} \cdot \prod_{k=1}^{m}\prod_{i=1}^{n_k} g_{n+1-(j_k+i)}^{\gamma\cdot y_{j_k+i}\cdot(x_{k,i}-1)}, \hat{g}) \\
&\quad \cdot \prod_{\kappa=1}^{m}\prod_{j=1}^{n_\kappa} e\Big(\big(g_{j_\kappa+j}^{\gamma_y} \cdot \prod_{k=1}^{m} \prod_{\substack{i\in[n_k] \\ (k,i)\neq(\kappa,j)}} g_{n+1-(j_k+i)+(j_\kappa+j)}^{y_{j_k+i}\cdot(x_{k,i}-1)}\big)^{x_{\kappa,j}}, \hat{g}\Big) \\
&= e\Big(g^{\gamma\cdot\gamma_y} \cdot \prod_{k=1}^{m}\prod_{i=1}^{n_k} g_{n+1-(j_k+i)}^{\gamma\cdot y_{j_k+i}\cdot(x_{k,i}-1)} \\
&\qquad \cdot \prod_{\kappa=1}^{m}\prod_{j=1}^{n_\kappa} \big(g_{j_\kappa+j}^{\gamma_y} \cdot \prod_{k=1}^{m} \prod_{\substack{i\in[n_k] \\ (k,i)\neq(\kappa,j)}} g_{n+1-(j_k+i)+(j_\kappa+j)}^{y_{j_k+i}\cdot(x_{k,i}-1)}\big)^{x_{\kappa,j}}, \hat{g}\Big),
\end{aligned}
$$

where the last equality holds since $x_{k,i} \cdot (x_{k,i} - 1) = 0$ for all indices $k \in [m]$ and $i \in [n_k]$.

Equation (36) is satisfied by $\pi_v = \prod_{i=m+1}^{n} \big(g_{n+1-i}^{r} \cdot \prod_{k=1}^{m} g_{n+2-i+k}^{x_k}\big)^{s_i}$ since

$$
\begin{aligned}
e(\prod_{i=m+1}^{n} g_{n+1-i}^{s_i}, \hat{V}) &= \prod_{i=m+1}^{n} e\Big(g_{n+1-i}, \hat{g}^{r} \cdot \prod_{k=1}^{m} \hat{g}_k^{x_k}\Big)^{s_i} \\
&= \prod_{i=m+1}^{n} e\Big(g^{r} \cdot \prod_{k=1}^{m} g_k^{x_k}, \hat{g}_{n+1-i}\Big)^{s_i} \\
&= \prod_{i=m+1}^{n} e\Big(g_{n+1-i}^{r} \cdot \prod_{k=1}^{m} g_{n+1-i+k}^{x_k}, \hat{g}\Big)^{s_i} = e(\pi_v, \hat{g}).
\end{aligned}
$$

D.3 Security

Theorem 3. *The construction provides zero-knowledge in the ROM.*

Proof. The proof is identical to that of Theorem 1 and omitted. □

Theorem 4. *Under the $(2n, n)$-DLOG assumption, the scheme is simulation-extractable in the algebraic group model and in the random oracle model.*

Proof. The proof is similar to that of Theorem 2 and we only detail the changes in the interaction between the reduction/extractor $\mathcal{B}$ and the adversary $\mathcal{A}$.

Queries: At any time, $\mathcal{A}$ can choose a commitment $\mathsf{com} = \hat{V}$ and ask for a simulated proof that $\hat{V}$ commits to an integer vector $(x_1, \ldots, x_m) \in \mathbb{Z}^m$ such that $x_k \in [0, 2^{\ell_k} - 1]$ for some integers $\{\ell_k\}_{k\in[m]}$ of its choice such that $\ell_k \leq n_k$. Since $\mathcal{A}$ is algebraic, it must provide a representation of $\hat{V}$ with respect to the generators $\{\hat{g}_i\}_{i\in[0,n]}$ and the commitments $\hat{C}$ contained in previous simulated proofs. Since the simulator $\mathcal{B}$ is itself algebraic, for any $\hat{V}$ chosen by $\mathcal{A}$, $\mathcal{B}$ can always find a representation $\{v_i\}_{i=0}^n$ such that $\hat{V} = \hat{g}^{v_0} \cdot \prod_{i=1}^n \hat{g}_i^{v_i}$. We assume w.l.o.g. that either: (i) There exists $k \in [m]$ such that $v_k \notin [0, 2^{\ell_k} - 1]$; or (ii) $(v_{m+1}, \ldots, v_n) \neq \mathbf{0}$. Otherwise, $\mathcal{B}$ can faithfully generate a proof using $(v_0, v_1, \ldots, v_m)$ as witnesses. Then, $\mathcal{B}$ proceeds as follows to simulate a proof without using g_{n+1}:

1. Choose random vectors $\boldsymbol{\xi} = (\xi_1, \ldots, \xi_m) \xleftarrow{R} \mathbb{Z}_p^m$, $\boldsymbol{\delta} = (\delta_x, \delta_{eq}, \delta_y, \delta_v) \xleftarrow{R} \mathbb{Z}_p^4$, $\boldsymbol{y} = (y_1, \ldots, y_n) \xleftarrow{R} \mathbb{Z}_p^n$, $\boldsymbol{t} = (t_1, \ldots, t_n) \xleftarrow{R} \mathbb{Z}_p^n$.
2. Let $f_{n+1} = \sum_{i=m+1}^n v_i \cdot s_i$, for randomly chosen $s_i \xleftarrow{R} \mathbb{Z}_p$ for all indices $i \in [m+1, n]$. Let an arbitrary $k \in [m]$ such that

$$a_{j_k+1} \triangleq v_k + \frac{1}{\xi_k} \cdot \Big(- \frac{\delta_v \cdot f_{n+1}}{\delta_x} + \sum_{\kappa\in[m]\setminus\{k\}} \xi_\kappa v_\kappa \Big) \notin \{0, 1\}$$

   Such a $k \in [m]$ must exist w.h.p. since we assumed that $(v_{m+1}, \ldots, v_n) \neq \mathbf{0}$ or there exists $k \in [m]$ such that $v_k \notin [0, 2^{\ell_k} - 1]$. Then, set

$$\begin{aligned} a_i &= 0 && \forall i \in [n] \setminus \{j_k + 1\} \\ z_{n-j_k} &= y_{j_k+1} \end{aligned}$$

   Then, find arbitrary scalars $\{z_j\}_{j\in[n]\setminus\{n-j_k\}}$ such that

$$\sum_{i=2}^{n_k} t_{j_k+i} \cdot z_{n+1-(j_k+i)} + \sum_{\substack{\kappa=1 \\ \kappa\neq k}}^{m} \sum_{i=1}^{n_\kappa} t_{j_\kappa+i} \cdot z_{n+1-(j_\kappa+i)} = t_{j_k+1} \cdot (a_{j_k+1} \cdot y_{j_k+1} - y_{j_k+1}).$$

3. Choose random $a_0, z_0 \xleftarrow{R} \mathbb{Z}_p$ and compute simulated commitments

$$\hat{C} = \hat{g}^{a_0} \cdot \prod_{i=1}^n \hat{g}_i^{a_i} = \hat{g}^{a_0} \cdot \hat{g}_{j_k+1}^{a_{j_k+1}}, \qquad C_y = g^{z_0} \cdot \prod_{i=1}^n g_i^{z_i}.$$

4. If one of the hashes $H_\xi(\hat{V}, \hat{C}, C_y)$, $H_{\text{agg}}(\hat{V}, \hat{C}, C_y)$, $H(\hat{V}, \hat{C})$, $H_t(\boldsymbol{y}, \hat{V}, \hat{C}, C_y)$ or $\{s_i = H_s(i, [m+1, n], \hat{V}, \hat{C}, C_y)\}_{i=m+1}^n$ was already defined, abort. Otherwise, set $\boldsymbol{\delta} = H_{\text{agg}}(\hat{V}, \hat{C}, C_y)$,

$$\boldsymbol{\xi} = H_\xi(\hat{V}, \hat{C}, C_y), \qquad \boldsymbol{y} = H(\hat{V}, \hat{C}), \qquad \boldsymbol{t} = H_t(\boldsymbol{y}, \hat{V}, \hat{C}, C_y),$$

   and $s_i = H_s(i, [m+1, n], \hat{V}, \hat{C}, C_y)$ for each $i \in [m+1, n]$.

5. Define the polynomials

$$\begin{aligned}
Q_x[X] &= \Big(\sum_{i=0}^{n} a_i \cdot X^i\Big) \cdot \Big(\sum_{k=1}^{m}\sum_{i=1}^{\ell_k} 2^{i-1} \cdot \xi_k \cdot X^{n+1-(j_k+i)}\Big) \\
&\qquad -\Big(\sum_{i=0}^{n} v_i \cdot X^i\Big) \cdot \Big(\sum_{k=1}^{m} \xi_k \cdot X^{n+1-k}\Big) = \sum_{i=0}^{2n} q_i \cdot X^i, \\
Q_y[X] &= \Big(\sum_{i=0}^{n} z_i \cdot X^i - \sum_{\kappa=1}^{m}\sum_{i=1}^{n_\kappa} y_{j_\kappa+i} \cdot X^{n+1-(j_\kappa+i)}\Big) \cdot \Big(\sum_{i=0}^{n} a_i \cdot X^i\Big) \\
&= \Big(z_0 + \sum_{\kappa=1}^{m}\sum_{i=1}^{n_\kappa} \big(z_{n+1-(j_\kappa+i)} - y_{j_\kappa+i}\big) \cdot X^{n+1-(j_\kappa+i)}\Big) \cdot \Big(\sum_{i=0}^{n} a_i \cdot X^i\Big) \\
&= \sum_{i=0}^{2n} \sigma_i \cdot X^i \\
Q_{eq}[X] &= \Big(\sum_{i=0}^{n} a_i \cdot X^i\Big) \cdot \Big(\sum_{\kappa=1}^{m}\sum_{i=1}^{n_\kappa} t_{j_\kappa+i} \cdot y_{j_\kappa+i} \cdot X^{n+1-(j_\kappa+i)}\Big) \\
&\qquad -\Big(z_0 + \sum_{\kappa=1}^{m}\sum_{i=1}^{n_\kappa} z_{n+1-(j_\kappa+i)} \cdot X^{n+1-(j_\kappa+i)}\Big) \cdot \Big(\sum_{\kappa=1}^{m}\sum_{i=1}^{n_\kappa} t_{j_\kappa+i} \cdot X^{j_\kappa+i}\Big) \\
&= \sum_{j=0}^{2n} e_j \cdot X^j, \\
Q_v[X] &= \Big(\sum_{i=0}^{n} v_i \cdot X^i\Big) \cdot \Big(\sum_{i=m+1}^{n} s_i \cdot X^{n+1-i}\Big) = \sum_{j=0}^{2n} f_j \cdot X^j.
\end{aligned}$$

Their degree-$(n+1)$ coefficients are $f_{n+1} = \sum_{i=m+1}^{n} v_i \cdot s_i$ and

$$\begin{aligned}
q_{n+1} &= \sum_{\kappa=1}^{m} \xi_\kappa \cdot \left(-v_\kappa + \sum_{i=1}^{\ell_\kappa} a_{j_\kappa+i} \cdot 2^{i-1}\right) \\
&= \xi_k \cdot a_{j_k+1} - \sum_{\kappa\in[m]} \xi_\kappa \cdot v_\kappa = -\frac{\delta_v \cdot f_{n+1}}{\delta_x}, \\
\sigma_{n+1} &= \sum_{\kappa=1}^{m}\sum_{i=1}^{n_\kappa} a_{j_\kappa+i} \cdot (z_{n+1-(j_\kappa+i)} - y_{j_\kappa+i}) = a_{j_k+1} \cdot (z_{n-j_k} - y_{j_k+1}) = 0 \\
e_{n+1} &= \sum_{\kappa=1}^{m}\sum_{i=1}^{n_\kappa} t_{j_\kappa+i} \cdot (a_{j_\kappa+i} \cdot y_{j_\kappa+i} - z_{n+1-(j_\kappa+i)}) \\
&= t_{j_k+1} \cdot (a_{j_k+1} \cdot y_{j_k+1} - y_{j_k+1}) - \sum_{i=2}^{n_k} t_{j_k+i} \cdot z_{n+1-(j_k+i)} \\
&\qquad - \sum_{\substack{\kappa=1 \\ \kappa\neq k}}^{m}\sum_{i=1}^{n_\kappa} t_{j_\kappa+i} \cdot z_{n+1-(j_\kappa+i)} = 0
\end{aligned}$$

due to the definition of committed $\boldsymbol{a} = (a_1, \ldots, a_n)$ and $\boldsymbol{z} = (z_1, \ldots, z_n)$. Observe that

$$\delta_x \cdot q_{n+1} + \delta_{eq} \cdot e_{n+1} + \delta_y \cdot \sigma_{n+1} + \delta_v \cdot f_{n+1} = 0 \tag{45}$$

6. Define the polynomial

$$Q_{\text{agg}}[X] = \delta_x \cdot Q_x[X] + \delta_{eq} \cdot Q_{eq}[X] + \delta_y \cdot Q_y[X] + \delta_v \cdot Q_v[X] = \sum_{i=0}^{2n} \eta_i \cdot X^i$$

for which $\eta_{n+1} = 0$ by construction. Compute

$$\pi = \prod_{i=1, i \neq n+1}^{2n} g_i^{\eta_i} \tag{46}$$

using $(g, \{g_i\}_{i \in [2n] \setminus \{n+1\}})$ and return the simulated proof $\boldsymbol{\pi} = (\hat{C}, C_y, \pi)$.

We remark that the simulated π from (46) satisfies the verification equation (38) by construction. Moreover, the proof $\boldsymbol{\pi}$ has the same distribution as a proof generated by the zero-knowledge simulator. Indeed, π is uniquely determined by the commitments $(\hat{C}, \hat{V}, C_y)$ and the aggregation coefficients $\boldsymbol{\xi}$, $\boldsymbol{y}, \boldsymbol{t}$, $\{s_i\}_{i=m+1}^n$ and $\boldsymbol{\delta}$ in (38). Also, the committed vectors $\boldsymbol{a}, \boldsymbol{z} \in \mathbb{Z}_p^n$ are perfectly hidden by the randomness a_0 and z_0 in $\hat{C}$ and C_y, respectively.

Therefore the simulation is perfect, unless a collision occurs when random oracles are programmed in the simulation queries. If Q_S (reps. Q_H) is the number of queries made by $\mathcal{A}$ to the simulator (resp. to random oracles), this happens with probability $\leq (Q_S + Q_H) \cdot Q_H / p$.

Output: When $\mathcal{A}$ terminates, it outputs a statement $(\hat{V}, \{1^{\ell_k}\}_{k=1}^m)$, for some integers $\ell_1 \in [n_1], \ldots, \ell_m \in [n_m]$, together with a valid proof $\boldsymbol{\pi} = \big(\hat{C}, C_y, \pi\big)$.

Since we are in the AGM, $\mathcal{A}$ must provide a representation of C_y w.r.t to $(g, \{g_i\}_{i \in [2n] \setminus \{n+1\}})$ and the group elements $\{C_y^{(i)}, \pi^{(i)}\}_{i \in [Q_S]}$ contained in responses $\{\pi^{(i)}\}_{i \in [Q_S]}$ to simulation queries. Likewise, it must provide a representation of $\hat{C}$ w.r.t $(\hat{g}, \{\hat{g}_i\}_{i \in [n]})$ and the commitments $\{\hat{C}^{(i)}\}_{i \in [Q_S]}$ contained in simulated proofs $\{\pi^{(i)}\}_{i \in [Q_S]}$. Also, for each $i \in [Q_S]$, $\mathcal{B}$ knows a representation of $\hat{C}^{(i)}$ w.r.t. $(\hat{g}, \{\hat{g}_i\}_{i \in [n]})$ and a representation of C_y w.r.t. $(g, \{g_i\}_{i=1}^n)$. It also knows a representation of each simulated $\pi^{(i)}$ w.r.t $(g, \{g_i\}_{i \in [2n] \setminus \{n+1\}})$. From $\mathcal{A}$'s output and the randomness of the simulation, $\mathcal{B}$ can infer scalars $\{(\theta_i, z_i) \in \mathbb{Z}_p^2\}_{i \in [0,2n] \setminus \{n+1\}}$, $\{(a_i, v_i) \in \mathbb{Z}_p^2\}_{i \in [0,n]}$ such that

$$\hat{C} = \prod_{i=0}^{n} \hat{g}_i^{a_i}, \qquad C_y = \prod_{i=0, i \neq n+1}^{2n} g_i^{z_i}, \qquad \hat{V} = \prod_{i=0}^{n} \hat{g}_i^{v_i}, \qquad \pi = \prod_{i=0, i \neq n+1}^{2n} g_i^{\theta_i},$$

where we define $g_0 = g$ and $\hat{g}_0 = \hat{g}$ for convenience.

If the representation $(v_0, v_1, \ldots, v_n) \in \mathbb{Z}_p^2$ of $\hat{V}$ is such that $v_k \in [0, 2^{\ell_k} - 1]$

for all $k \in [m]$ and $v_i = 0$ for all $i \in [m+1, n]$, then $\mathcal{B}$ can simply output $(v_0, v_1, \ldots, v_m) \in \mathbb{Z}_p^{m+1}$ as a valid witness. We henceforth assume that either $(v_{m+1}, \ldots, v_n) \neq \mathbf{0}^{n-m}$ or there exists $k \in [m]$ such that $v_k \notin [0, 2^{\ell_k} - 1]$.

Solving $(2n, n)$-DLOG: We first note that a non-trivial valid proof $\boldsymbol{\pi}$ cannot recycle $(\hat{V}, \hat{C}, C_y)$ from an output of the simulation oracle (namely, we must have $(\hat{V}, \hat{C}, C_y) \neq (\hat{V}^{(i)}, \hat{C}^{(i)}, C_y)$ for all $i \in [Q_S]$) since the left-hand-side member of (38) is uniquely determined by $(\hat{V}^{(i)}, \hat{C}^{(i)}, C_y^{(i)})$ and it in turn determines a unique valid $\pi^{(i)} \in \mathbb{G}$. As a consequence, $H_{\mathrm{agg}}(\hat{V}, \hat{C}, C_y)$, $H_{\xi}(\hat{V}, \hat{C}, C_y)$, $H_t(\boldsymbol{y}, \hat{V}, \hat{C}, C_y)$ and $\{H_s(i, [m+1, n], \hat{V}, \hat{C}, C_y)\}_{i=m+1}^{n}$ are not part of the random oracle values that have been programmed by the simulator.

Since the left-hand-side member of (38) is obtained by raising the right-hand-side members of (33)-(36) to the powers $(\delta_x, \delta_{eq}, \delta_y, \delta_v)$ and multiplying the results, it can be written $e(g, \hat{g})^{P_{\mathrm{agg}}(\alpha)}$, where $P_{\mathrm{agg}}[X]$ is the polynomial

$$P_{\mathrm{agg}}[X] = \delta_x \cdot P_x[X] + \delta_y \cdot P_y[X] + \delta_{eq} \cdot P_{eq}[X] + \delta_v \cdot P_v[X]$$

obtained as a linear combination of the polynomials

$$\begin{aligned} P_x[X] = &\Big(\sum_{i=0}^{n} a_i \cdot X^i\Big) \cdot \Big(\sum_{k=1}^{m} \sum_{i=1}^{\ell_k} 2^{i-1} \cdot \xi_k \cdot X^{n+1-(j_k+i)}\Big) \\ &- \Big(\sum_{i=0}^{n} v_i \cdot X^i\Big) \cdot \Big(\sum_{k=1}^{m} \xi_k \cdot X^{n+1-k}\Big) = \sum_{i=1}^{2n} \omega_i \cdot X^i, \end{aligned}$$

$$\begin{aligned} P_y[X] &= \Big(\sum_{i=0, i\neq n+1}^{2n} z_i \cdot X^i - \sum_{\kappa=1}^{m} \sum_{i=1}^{n_\kappa} y_{j_\kappa+i} \cdot X^{n+1-(j_\kappa+i)}\Big) \cdot \Big(\sum_{i=0}^{n} a_i \cdot X^i\Big) \\ &= \Big(z_0 + \sum_{\kappa=1}^{m} \sum_{i=1}^{n_\kappa} \big(z_{n+1-(j_\kappa+i)} - y_{j_\kappa+i}\big) \cdot X^{n+1-(j_\kappa+i)} + \sum_{i=n+2}^{2n} z_i \cdot X^i\Big) \\ &\quad \cdot \Big(\sum_{i=0}^{n} a_i \cdot X^i\Big) = \sum_{i=0}^{3n} \gamma_i \cdot X^i \\ P_{eq}[X] &= \Big(\sum_{i=0}^{n} a_i \cdot X^i\Big) \cdot \Big(\sum_{\kappa=1}^{m} \sum_{i=1}^{n_\kappa} t_{j_\kappa+i} \cdot y_{j_\kappa+i} \cdot X^{n+1-(j_\kappa+i)}\Big) \\ &\quad - \Big(\sum_{i=0, i\neq n+1}^{2n} z_i \cdot X^i\Big) \cdot \Big(\sum_{\kappa=1}^{m} \sum_{i=1}^{n_\kappa} t_{j_\kappa+i} \cdot X^{j_\kappa+i}\Big) = \sum_{j=0}^{3n} \beta_j \cdot X^j, \\ P_v[X] &= \Big(\sum_{i=0}^{n} v_i \cdot X^i\Big) \cdot \Big(\sum_{i=m+1}^{n} s_i \cdot X^{n+1-i}\Big) = \sum_{j=0}^{2n} \mu_j \cdot X^j \end{aligned}$$

for which the left-hand-side members of (33)-(36) can be written $e(g, \hat{g})^{P_x(\alpha)}$, $e(g, \hat{g})^{P_{eq}(\alpha)}$, $e(g, \hat{g})^{P_y(\alpha)}$ and $e(g, \hat{g})^{P_v(\alpha)}$, respectively.

If we write $P_{\mathrm{agg}}[X] = \sum_{i=0}^{3n} \nu_i \cdot X^i$, the coefficient ν_{n+1} of its degree-$(n+1)$ term can be written

$$\nu_{n+1} = \delta_x \cdot \underbrace{\sum_{\kappa=1}^{m} \xi_\kappa \cdot \left(\sum_{i=1}^{\ell_\kappa} a_{j_\kappa+i} \cdot 2^{i-1} - v_\kappa \right)}_{\triangleq\ \omega_{n+1}} + \delta_y \cdot \underbrace{\sum_{\kappa=1}^{m} \sum_{i=1}^{n_\kappa} \left(z_{n+1-(j_\kappa+i)} - y_{j_\kappa+i} \right) \cdot a_{j_\kappa+i}}_{\triangleq\ \gamma_{n+1}}$$
$$+\ \delta_{eq} \cdot \underbrace{\sum_{\kappa=1}^{m} \sum_{i=1}^{n_\kappa} t_{j_\kappa+i} \cdot (a_{j_\kappa+i} \cdot y_{j_\kappa+i} - z_{n+1-(j_\kappa+i)})}_{\triangleq\ \beta_{n+1}} \ +\ \delta_v \cdot \underbrace{\sum_{i=m+1}^{n} v_i \cdot s_i}_{\triangleq\ \mu_{n+1}},$$

where $(\omega_{n+1}, \gamma_{n+1}, \beta_{n+1}, \mu_{n+1})$ are the coefficients of the degree-$(n+1)$ terms of $(P_x[X], P_y[X], P_{eq}[X], P_v[X])$.

We now argue that, if there exists $k \in [m]$ such that $v_k \notin [0, 2^{\ell_k} - 1]$ or if $(v_{m+1}, \ldots, v_n) \neq \mathbf{0}^{n-m}$, then we can only have $\nu_{n+1} = 0$ with negligible probability. This follows from the following arguments:

- The probability to have $\boldsymbol{\rho} \triangleq (\omega_{n+1}, \gamma_{n+1}, \beta_{n+1}, \mu_{n+1}, \zeta_{n+1}) = \mathbf{0}$ is negligible. Indeed, when $(v_{m+1}, \ldots, v_n) \neq \mathbf{0}^{n-m}$, we have $\mu_{n+1} = 0$, with probability $1/p$ over the choice of $\{s_i = H_s(i, [m+1, n], \hat{V}, \hat{C}, C_y)\}_{i=m+1}^{n}$. When $z_{n+1-(j_\kappa+i)} \neq a_{j_\kappa+i} \cdot y_{j_\kappa+i}$ for some $\kappa \in [m]$ and $i \in [n_\kappa]$, we have $\beta_{n+1} = 0$ with probability $1/p$ since $\boldsymbol{t} = H_t(\boldsymbol{y}, \hat{V}, \hat{C}, C_y)$ is derived *after* the choice of $\boldsymbol{y}$, $\{a_i\}_{i=0}^{n}$ and $\{z_i\}_{i \in [0,2n] \setminus \{n+1\}}$. Then, if $z_{n+1-(j_\kappa+i)} = a_{j_\kappa+i} \cdot y_{j_\kappa+i}$ for all $\kappa \in [m]$, $i \in [n_\kappa]$, we have $\gamma_{n+1} = \sum_{\kappa=1}^{m} \sum_{i=1}^{n_\kappa} y_{j_\kappa+i} \cdot (a_{j_\kappa+i} - 1) \cdot a_{j_\kappa+i}$, which cancels with probability $1/p$ if there exists $\kappa \in [m]$ and $i \in [\ell_\kappa]$ such that $a_{j_\kappa+i} \notin \{0, 1\}$. This can be seen by distinguishing two cases:

  a. If $\boldsymbol{y} = H(\hat{V}, \hat{C})$ was programmed when answering a simulation query, we can only have $\gamma_{n+1} = 0$ with probability $1/p$ since the simulator programmed $(a_1, \ldots, a_n)$ to have

  $$\gamma_{n+1} = a_{j_k+1} \cdot (z_{n-j_k} - y_{j_k+1}) = y_{j_k+1} \cdot a_{j_k+1} \cdot (a_{j_k+1} - 1)$$

  for some index $j_k \in [n]$ such that $a_{j_k+1} \notin \{0, 1\}$ and $y_{j_k+1} \in_R \mathbb{Z}_p$. This captures the case of $\mathcal{A}$ attempting to re-use $(\hat{V}, \hat{C}) = (\hat{V}^{(i)}, \hat{C}^{(i)})$ contained in an output $\boldsymbol{\pi}^{(i)} = (\hat{C}^{(i)}, C_y^{(i)}, \pi^{(i)})$ of the simulator, with a different $C_y \neq C_y^{(i)}$.

  b. If $H(\hat{V}, \hat{C})$ was not programmed by the simulator, then $\boldsymbol{y} = H(\hat{V}, \hat{C})$ was defined after $\mathcal{B}$ obtained the algebraic representation $\{a_i\}_{i=0}^{n}$ of $\hat{C}$. Over the choice of $\boldsymbol{y}$, we have $\sum_{\kappa=1}^{m} \sum_{i=1}^{n_\kappa} y_{j_\kappa+i} \cdot (a_{j_\kappa+i} - 1) \cdot a_{j_\kappa+i} = 0$ with probability $1/p$.

  If there exists $k \in [m]$ such that $v_k \neq \sum_{i=1}^{\ell_k} a_{j_k+i} \cdot 2^{i-1}$, the probability to have $\omega_{n+1} = 0$ is only $1/p$ since $\boldsymbol{\xi} = H_\xi(\hat{V}, \hat{C}, C_y)$ are chosen uniformly after $\{v_k\}_{k=1}^{m}$ and $\{a_i\}_{i=0}^{n}$. If none of the above events occurs, then we have

$v_k = \sum_{i=1}^{\ell_k} a_{j_k+i} \cdot 2^{i-1}$ for each $k \in [m]$ and $a_{j_k+i} \in \{0,1\}$ for each $i \in [\ell_k]$. This contradicts the hypothesis that $v_k \notin [0, 2^{\ell_k} - 1]$ for some $k \in [m]$.

- If $\boldsymbol{\rho} \neq \mathbf{0}$, then we have $\nu_{n+1} \neq 0$ with probability $1-1/p$ since the aggregation coefficients $\boldsymbol{\delta} \triangleq (\delta_x, \delta_{eq}, \delta_y, \delta_v) = H_{\text{agg}}(\hat{V}, \hat{C}, C_y)$ are derived *after* the choice of $\{(a_i, v_i)\}_{i=0}^n$, and $\{z_i\}_{i \in [0,2n] \setminus \{n+1\}}$, which determine the coordinates of $\boldsymbol{\rho}$. Hence, a random independent $\boldsymbol{\delta} \in \mathbb{Z}_p^5$ can only satisfy $\langle \boldsymbol{\delta}, \boldsymbol{\rho} \rangle = 0$ with probability $1/p$.

If $\nu_{n+1} \neq 0$, then $\mathcal{B}$ can compute $\alpha \in \mathbb{Z}_p$ by factoring a non-zero polynomial as in the proof of Theorem 2. □

D.4 Range Proofs for Non-Power-of-Two Ranges

In order to prove membership of a range $[0, B]$ where $B+1$ is not a power of 2, a common approach to to use an additively homomorphic commitment and consider the integer $\ell \in \mathbb{N}$ such that $2^{\ell-1} < B < 2^\ell$. Then, we generate two range proofs showing that $x \in [0, 2^\ell - 1]$ and $x + (2^\ell - 1 - B) \in [0, 2^\ell - 1]$.

To do this without increasing the proof size, we can commit to vectors of dimension $n = 2\bar{\ell}$ (where $\bar{\ell}$ is an upper bound on ℓ) containing a concatenation $(\boldsymbol{x} \mid \boldsymbol{x}')$ of the binary decompositions $\boldsymbol{x} = (x_1, \ldots, x_\ell, 0, \ldots, 0)$ and $\boldsymbol{x}' = (x'_1, \ldots, x'_\ell, 0, \ldots, 0)$ of x and $x + (2^\ell - 1 - B)$, respectively. Then, the prover can compute $\pi_x, \pi'_x \in \mathbb{G}$

$$\frac{e(\prod_{i=1}^{\bar{\ell}} g_{n+1-i}^{2^{i-1}}, \hat{C})}{e(g_n, \hat{V})} = e(\pi_x, \hat{g}) \qquad \frac{e(\prod_{i=\bar{\ell}+1}^{n} g_{n+1-i}^{2^{i-\bar{\ell}-1}}, \hat{C})}{e(g_n, \hat{V} \cdot \hat{g}^{2^\ell - 1 - B})} = e(\pi'_x, \hat{g}),$$

where $\hat{V} = \hat{g}^r \cdot \hat{g}_1^x$. In the above equalities, π_x and π'_x show that $x = \sum_{i=1}^{\bar{\ell}} x_i \cdot 2^{i-1}$ and $x + (2^\ell - 1 - B) = \sum_{i=1}^{\bar{\ell}} x'_i \cdot 2^{i-1}$. The rest of the proof follows the basic construction of Section 4. The two proofs π_x and π'_x can be aggregated (using additional randomization components derived from a random oracle) with other proof components to obtain a proof of the same form as in Section 4.

E Proving Small Euclidean Norms

In this section, we extend the construction of Section D to prove that a commitment $\hat{V} = \hat{g}^r \cdot \prod_{k=1}^m \hat{g}_k^{x_k}$ is a commitment to some $\boldsymbol{x} = (x_1, \ldots, x_m) \in \mathbb{Z}^m$ such that $\|\boldsymbol{x}\| \leq B$. In order to preserve the zero-knowledge property, we need to choose the group order p so that $\sqrt{p/m} > B$ for any proven norm bound B. When the CRS is generated, we thus assume a maximal value $\bar{B}$ for the norm bounds to be proven and choose p so that $B_\infty \triangleq \sqrt{p/m} > \bar{B}$. For simplicity, we assume that $B^2 + 1$ is a power of two but this restriction can be lifted using the observations in Section D.4. In the setup phase, we also need parameters allowing commitments to vectors of dimension $n = \max(m \cdot (1 + \lceil \log B_\infty \rceil), \lceil \log(\bar{B}^2 + 1) \rceil)$ in order to use the scheme of Section D.1.

The prover has a commitment $\hat{V} = \hat{g}^{\hat{r}} \cdot \prod_{k=1}^{m} \hat{g}_k^{x_k} \in \hat{\mathbb{G}}$ to an integer vector $\boldsymbol{x} = (x_1, \ldots, x_m) \in \mathbb{Z}^m$ and wishes to convince the verifier that $\|\boldsymbol{x}\| \leq B$ without revealing anything else. To this end, it chooses $r \xleftarrow{R} \mathbb{Z}_p$ and computes $V = g^r \cdot \prod_{k=1}^{m} g_{n+1-k}^{x_k}$, which commits to $\bar{\boldsymbol{x}} = (\mathbf{0}^{n-m} \mid (x_m, \ldots, x_1))$ in the first source group $\mathbb{G}$. Then, the prover generates a proof showing that $V \in \mathbb{G}$ commits to the same vector as $\hat{V}$, but in the reversed order. This is done by computing $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_m) = H_\theta(\hat{V}, V) \in \mathbb{Z}_p^m$ using a random oracle H_θ and computing a proof

$$\pi_\theta = \frac{\prod_{j=1}^{m} \left(g_{n+1-j}^{\hat{r}} \cdot \prod_{k \in [m] \setminus \{j\}} g_{n+1+k-j}^{x_k} \right)^{\theta_j}}{\prod_{j=1}^{m} \left(g_j^{r} \cdot \prod_{k \in [m] \setminus \{j\}} g_{n+1-k+j}^{x_k} \right)^{\theta_j}},$$

satisfying

$$\frac{e\big(\prod_{j=1}^{m} g_{n+1-j}^{\theta_j}, \hat{V}\big)}{e\big(V, \prod_{j=1}^{m} \hat{g}_j^{\theta_j}\big)} = e(\pi_\theta, \hat{g}). \tag{47}$$

By itself, (47) only argues that the first m entries of $\boldsymbol{x} \in \mathbb{Z}_p^n$ coincide with the last $n - m$ entries of $\bar{\boldsymbol{x}}$ in the reversed order. To ensure knowledge soundness, we also need to prove that the last $n - m$ positions of $\boldsymbol{x}$ are zeroes, but this will be addressed at a later step.

Next, assuming that $\hat{V}$ and V were indeed computed by the prover as commitments to $((x_1, \ldots, x_m) \mid \mathbf{0}^{n-m})$ and $((*, \ldots, *) \mid (x_m, \ldots, x_1))$, respectively, we observe that the pairing $e(V, \hat{V})$ computes a product of polynomials in the exponent, where the coefficient of α^{n+1} is $\|\boldsymbol{x}\|^2 = \langle \boldsymbol{x}, \boldsymbol{x} \rangle$. This allows the prover the compute $\pi_B = \prod_{k=1}^{m} (g_k^{r_v} \cdot \prod_{\kappa=1, \kappa \neq k}^{m} g_{n+1-\kappa+k}^{x_\kappa})^{x_k}$ such that

$$e(V, \hat{V}) = e(g_1, \hat{g}_n)^{\langle \boldsymbol{x}, \boldsymbol{x} \rangle} \cdot e(\pi_B, \hat{g}). \tag{48}$$

However, π_B is not disclosed. Instead, the prover computes the ℓ_B-bit representation of $\|\boldsymbol{x}\|^2 = \sum_{k=1}^{m} x_k^2$ (where $\ell_B = \log(B^2 + 1)$) and commits to the vector $\boldsymbol{w} = (w_1, \ldots, w_{\ell_B}, 0, \ldots, 0) \in \{0,1\}^n$ by choosing $\gamma \xleftarrow{R} \mathbb{Z}_p$, computing

$$\hat{C}_w = \hat{g}^\gamma \cdot \prod_{j=1}^{\ell_B} \hat{g}_j^{w_j} \tag{49}$$

and proving that the committed $\boldsymbol{w}$ is a binary vector. This is done by generating a proof $\boldsymbol{\pi}_w = (C_{y,w}, \pi_w) \in \mathbb{G}^2$ as in Section 3. Now, we observe that the prover can compute $\pi'_B = \prod_{i=1}^{\ell_B} \left(g_{n+1-i}^{\gamma} \cdot \prod_{j=1, j \neq i}^{\ell_B} g_{n+1-i+j}^{w_j} \right)^{2^{i-1}}$ such that

$$e\big(\prod_{i=1}^{\ell_B} g_{n+1-i}^{2^{i-1}}, \hat{C}_w\big) = e(g_1, \hat{g}_n)^{\langle \boldsymbol{x}, \boldsymbol{x} \rangle} \cdot e(\pi'_B, \hat{g}). \tag{50}$$

By dividing (50) from (48), we see that the prover is able to compute a short $\bar{\pi}_B = \pi_B / \pi'_B \in \mathbb{G}$ such that

$$\frac{e(V, \hat{V})}{e\big(\prod_{i=1}^{\ell_B} g_{n+1-i}^{2^{i-1}}, \hat{C}_w\big)} = e(\bar{\pi}_B, \hat{g}). \tag{51}$$

Together with the proof π_θ satisfying (47) and $\boldsymbol{\pi}_w$, $\bar{\pi}_B$ shows that $\hat{V}$ commits to a vector $((x_1, \ldots, x_m) \mid \mathbf{0}^{n-m}) \in \mathbb{Z}_p^n$ (we still assume that its last $n-m$ positions are proven to be 0) such that $\sum_{k=1}^m x_k^2 \bmod p$ is at most $2^{\ell_B} - 1$.

However, we also need to prove that $\sum_{k=1}^m x_k^2 \bmod p$ is actually $\sum_{k=1}^m x_k^2$ over $\mathbb{Z}$, in which case we have $\|\boldsymbol{x}\| \leq B$. To this end, the prover uses the construction of Section D.1 to prove that $\|\boldsymbol{x}\|_\infty \leq \sqrt{p/m}$, which ensures that $\sum_{k=1}^m x_k^2$ does not wrap around modulo p. We note that this additional proof component does not affect the zero-knowledge property because the proven statement $\|\boldsymbol{x}\| \leq B$ already implies $\|\boldsymbol{x}\|_\infty \leq \sqrt{p/m}$ (recall that we assumed $B \leq \sqrt{p/m}$ and we always have $\|\boldsymbol{x}\|_\infty \leq \|\boldsymbol{x}\|$). The prover thus generates a proof $\boldsymbol{\pi}_\infty = (\hat{C}, C_y, \pi_\infty) \in \hat{\mathbb{G}} \times \mathbb{G}^2$ that $\|\boldsymbol{x}\|_\infty \leq B_\infty$ by proving that $0 \leq x_k + \sqrt{p/m} < 2\sqrt{p/m}$ for each $k \in [m]$ using the construction of Section D, which also demonstrates that $\hat{V}$ commits to a vector containing zeroes in its last $n - m$ entries.

The entire proof $\boldsymbol{\pi} = \big(V, \boldsymbol{\pi}_\infty, \hat{C}_w, \boldsymbol{\pi}_w, \pi_\theta, \bar{\pi}_B\big)$ is eventually comprised of $\boldsymbol{\pi}_\infty = (\hat{C}, C_y, \pi_\infty) \in \hat{\mathbb{G}} \times \mathbb{G}^2$, the commitment $\hat{C}_w$ and its proof of binarity $\boldsymbol{\pi}_w = (C_{y,w}, \pi_w) \in \mathbb{G}^2$, the commitment V and the proof π_θ satisfying (47), and the proof $\bar{\pi}_B$ satisfying (51).

From a security standpoint, the knowledge-soundness property follows from that of underlying proof components. Simulation-extractability is also preserved as long as these components are bound together in a non-malleable way. To do this, one option is to use a short one-time signature (such as the one from [64, Section 5.4]) whose verification key is included in all random oracle inputs. However, more efficient solutions are possible by suitably combining the various sub-proofs together and including previously computed commitments in each random oracle input.

In terms of efficiency, it is also possible to exploit the linearity of verification equations and compress $(\pi_\infty, \pi_\theta, \pi_w, \bar{\pi}_B) \in \mathbb{G}^4$ into a single group element $\pi = \pi_\infty^{\delta_\infty} \cdot \pi_\theta^{\delta_\theta} \cdot \pi_w^{\delta_w} \cdot \bar{\pi}_B^{\delta_B}$ using aggregation coefficients $(\delta_\infty, \delta_\theta, \delta_w, \delta_B)$ derived from a random oracle. This shrinks the proof to 2 elements of $\hat{\mathbb{G}}$ and 4 elements of $\mathbb{G}$ while verification boils down to a product of 8 pairings.

F Proving Small Hamming Weights

In this section, we show that our argument of Section 3 can be extended to prove that committed vectors have small Hamming weights.

Compact proofs of small Hamming weights were previously considered by Damgård *et al.* [38] in the context of perfectly binding commitments. To our knowledge, no efficient solution to this problem has been reported in the case of perfectly hiding commitments if we aim at constant-size proofs. The only solution we are aware of is to rely on SNARKs for general NP relations via an expensive Karp reduction.

F.1 Proving Exact Hamming Weights for Binary Vectors

For a commitment $\hat{C} = \hat{g}^{\gamma} \cdot \prod_{j=1}^{n} \hat{g}_j^{x_j}$ to a binary $\boldsymbol{x} \in \{0,1\}^n$, we can also prove that $\boldsymbol{x}$ has a fixed Hamming weight. This is useful in the context of FHE, where secret keys are sometimes chosen with a special structure for efficiency reasons. To prove that a committed $\boldsymbol{x}$ is a binary vector of Hamming weight k, we can prove that: (i) $\boldsymbol{x}$ is binary; (ii) Its inner product with the all-one vector $(1, 1, \ldots, 1)$ is exactly k. Our technique from Section 3 allows handling (i). In order to prove (ii), the prover can generate a short $\pi_k \in \mathbb{G}$ such that

$$e(\prod_{i=1}^{n} g_{n+1-i}, \hat{C}) = e(g_1, \hat{g}_n)^k \cdot e(\pi_k, \hat{g}), \tag{52}$$

which is possible as in (1). Again, we can aggregate π_k with other proof components to obtain a proof comprised of one element of $\hat{\mathbb{G}}$ and two elements of $\mathbb{G}$.

If we combine the above idea with the range proof construction, it is also possible to prove that the Hamming weight $HW(\boldsymbol{x})$ of the committed $\boldsymbol{x}$ is such that $HW(\boldsymbol{x}) \leq B$, for some bound B, without revealing the exact weight. In Appendix F.3, we provide a more efficient way to prove the inequality $HW(\boldsymbol{x}) \leq B$ for arbitrary (i.e., not necessarily binary) vectors.

F.2 Proving Exact Hamming Weights for Ternary Vectors

If we want to prove the *exact* Hamming weight a committed $\boldsymbol{x} \in \{-1, 0, 1\}^{\ell}$, we can first generate a commitment to $\bar{\boldsymbol{x}} = \boldsymbol{x} \circ \boldsymbol{x}$, which is the Hadamard product of $\boldsymbol{x}$ with itself. We can then prove that: (i) $\bar{\boldsymbol{x}}$ is indeed the product $\boldsymbol{x} \circ \boldsymbol{x} = (x_1^2, \ldots, x_n^2)$; (ii) $\bar{\boldsymbol{x}}$ is binary and has Hamming weight B.

In order to prove (i), we need to compute $C_{sq} = g^{\gamma_{sq}} \cdot \prod_{j=1}^{n} g_j^{x_j^2}$ as an auxiliary commitment to $\bar{\boldsymbol{x}} = \boldsymbol{x} \circ \boldsymbol{x}$, for a random $\gamma_{sq} \xleftarrow{R} \mathbb{Z}_p$, and prove that C_{sq} commits to $\boldsymbol{x} \circ \boldsymbol{x}$. This can be done by adapting the technique of Section 3. The prover generates yet another commitment $C_y = g^{\gamma_y} \cdot \prod_{i=1}^{n} g_{n+1-i}^{y_i \cdot x_i}$ and proves that it commits to $(y_n \cdot x_n, \ldots, y_1 \cdot x_1)$ by proceeding exactly as in Section 3. Specifically, by computing $(y_1, \ldots, y_n) = H(\hat{C}, C_{sq})$ and adapting (7), the prover can generate a short $\pi_y \in \mathbb{G}$ such that

$$e\big(C_y, \hat{C}\big) = e(\pi_y, \hat{g}) \cdot e(g_1, \hat{g}_n)^{\sum_{i=1}^{n} y_i \cdot x_i^2} \tag{53}$$

Since the prover can also compute a short $\pi'_y \in \mathbb{G}$ such that C_{sq} satisfies

$$e\big(C_{sq}, \prod_{i=1}^{n} g_{n+1-i}^{y_i}\big) = e(\pi'_y, \hat{g}) \cdot e(g_1, \hat{g}_n)^{\sum_{i=1}^{n} y_i \cdot x_i^2}, \tag{54}$$

it can compute $\pi_{sq} = \pi_y / \pi'_y \in \mathbb{G}$ such that

$$\frac{e\big(C_y, \hat{C}\big)}{e\big(C_{sq}, \prod_{i=1}^{n} g_{n+1-i}^{y_i}\big)} = e(\pi_{sq}, \hat{g}), \tag{55}$$

which follows by dividing (54) from (53). From $\hat{C} \in \hat{\mathbb{G}}$, $C_y \in \mathbb{G}$, $C_{sq} \in \mathbb{G}$, π_{sq} and π_{eq} (which shows that C_y commits to $(y_n \cdot x_n, \ldots, y_1 \cdot x_1)$), the verifier is convinced that C_{sq} is a commitment to $\boldsymbol{x} \circ \boldsymbol{x}$.

F.3 Proving Bounded Hamming Weights for Arbitrary Vectors

We now consider the problem of proving small Hamming weights for an arbitrary vector $\boldsymbol{x} = (x_1, \ldots, x_n) \in \mathbb{Z}_q^n$ committed as $\hat{C} = \hat{g}^{\gamma} \cdot \prod_{j=1}^{n} \hat{g}_j^{x_j}$. Using the additive homomorphic property of the commitment scheme, this also allows proving that two committed vectors are close in terms of Hamming distance.

In order to prove that $\boldsymbol{x}$ has at most B non-zero positions, we first generate a commitment C_w to a random vector $\boldsymbol{w} = (w_1, \ldots, w_n) \in \{0,1\}^n$ of Hamming weight $HW(\boldsymbol{w}) = B$ for which $w_i = 1$ for all $i \in [n]$ such that $x_i \neq 0$. We can then prove that: (i) $\boldsymbol{w}$ is binary and has Hamming weight B; (ii) For each $i \in [n]$, $w_i = 1$ whenever $x_i \neq 0$, which implies $HW(\boldsymbol{x}) \leq HW(\boldsymbol{w})$.

We can prove (i) as explained in Section F.1. In order to prove (ii), we will prove that $\sum_{i=1}^{n} y_i \cdot (1 - w_i) \cdot x_i = 0$ for a random vector $\boldsymbol{y} = (y_1, \ldots, y_n)$, which ensures that $\forall i \in [n] : (x_i \neq 0) \Rightarrow (w_i = 1)$ with probability $1 - 1/p$. Indeed, if there exists $i \in [n]$ such that $x_i \neq 0$ and $w_i = 0$, we have $\sum_{i=1}^{n} y_i \cdot (1 - w_i) \cdot x_i = 0$ with probability $1/p$ since $\boldsymbol{y}$ is computed after $\boldsymbol{w}$ and $\boldsymbol{x}$.

In more details, the prover computes a commitment $\hat{C}_w = \hat{g}^{\gamma_w} \cdot \prod_{j=1}^{n} \hat{g}_j^{w_i}$ to $\boldsymbol{w} \in \{0,1\}^n$, for some random $\gamma_w \xleftarrow{R} \mathbb{Z}_p$, and proves that $\hat{C}_w$ is a commitment to a binary vector using a short proof $\boldsymbol{\pi}_w = (C_{y,w}, \pi_w) \in \mathbb{G}^2$. Then, the prover generates another commitment $C_y = g^{\gamma_y} \cdot \prod_{i=1}^{n} g_{n+1-i}^{y_i \cdot w_i}$ and proves that it commits to $(y_n \cdot w_n, \ldots, y_1 \cdot w_1)$, where $(y_1, \ldots, y_n) = H(\hat{C}, \hat{C}_w)$, by proceeding exactly as in Section 3. Next, the prover can generate a short $\pi_y \in \mathbb{G}$ such that

$$e\Big(\prod_{i=1}^{n} g_{n+1-i}^{y_i} \cdot C_y^{-1}, \hat{C}\Big) = e(\pi_y, \hat{g}) \cdot e(g_1, \hat{g}_n)^{\sum_{i=1}^{n} y_i \cdot (1 - w_i) \cdot x_i} = e(\pi_y, \hat{g}), \qquad (56)$$

which is possible since $\prod_{i=1}^{n} g_{n+1-i}^{y_i} \cdot C_y^{-1} = g^{\gamma_y} \cdot \prod_{i=1}^{n} g_{n+1-i}^{y_i \cdot (1 - w_i)}$, so that the sum $\sum_{i=1}^{n} y_i \cdot (1 - w_i) \cdot x_i$ is the coefficient of α^{n+1} when we see the left-hand-side member of (56) as a product of polynomials in the exponent.

The final proof then consists of $\hat{C}_w \in \hat{\mathbb{G}}$, $C_{y,w}, C_y \in \mathbb{G}$, and a short $\pi \in \mathbb{G}$ obtained by aggregating the various proof components (including π_w, π_y, the proof π_B that $\hat{C}_w$ satisfies (52), and the proof that C_y is correctly formed).

G Shorter Proofs for Ring LWE Ciphertexts

In this section, we show that the techniques of previous sections can be used to obtain very short proofs for natural statements that arise in lattice-based cryptography. For example, they can be used for all the applications described in [45]. It includes proving the validity of an LPR ciphertext [85], a BGV ciphertext

[17], a ring GSW ciphertext [56], a TFHE ciphertext [32], or a ring LWE public key. They can also be used to prove that a committed vector contains a GPV signature [55].

We adapt the approach of del Pino *et al.* [41] with the difference that we replace the BulletProofs range proof by our more compact proofs of smallness. We also exploit the fact that the underlying vector commitment [81] allows proving inner-product relations as in [80].

Let the polynomial rings $R = \mathbb{Z}[X]/(X^d + 1)$ and $R_q = R/(qR)$, where d is a power of two. As in [41], we aim at proving the existence of a witness $\boldsymbol{s} = (s_1, \ldots, s_M) \in R^M$ comprised of small-norm ring elements such that

$$\sum_{i=1}^{M} \boldsymbol{a}_i \cdot s_i = \boldsymbol{c} \bmod (q, X^d + 1) \tag{57}$$

for public $\boldsymbol{c}, \boldsymbol{a}_1, \ldots, \boldsymbol{a}_M \in R_q^N$. The relation is defined as the set of pairs

$$(\mathbf{x}, \mathbf{w}) = \Big((\boldsymbol{c}, \boldsymbol{a}_1, \ldots, \boldsymbol{a}_M) \in (R_q^N)^{M+1}, (\boldsymbol{s}_1, \ldots, \boldsymbol{s}_M) \in R^M \Big)$$

satisfying (57). To prove this relation, we proceed as in [41] and re-write (57) as the following equality over $\mathbb{Z}[X]/(X^d + 1)$

$$\sum_{i=1}^{M} \boldsymbol{a}_i \cdot s_i = \boldsymbol{c} + \boldsymbol{r} \cdot q \bmod (X^d + 1), \tag{58}$$

where $\boldsymbol{r} = (r_1, \ldots, r_N)^\top \in R^N$ is a vector of polynomials of degree $\leq d - 1$ and the components of $\{\boldsymbol{a}_i\}_{i=1}^M$ and $\boldsymbol{c}$ are interpreted as polynomials with coefficients in $\{-\lfloor q/2 \rfloor, \ldots, \lfloor q/2 \rfloor\}$. If $\|s_i\|_\infty \leq B_i$ for each $i \in [M]$, $\boldsymbol{r}$ contains polynomials with coefficients of magnitude smaller than $\|\boldsymbol{r}\|_\infty \leq B_r \triangleq dM \cdot \max_{i \in [M]}(B_i)/2$.

Let us parse $\boldsymbol{a}_i = (a_{i,1}, \ldots, a_{i,N})^\top \in R_q^N$. Let the coefficient embedding $\phi : R \to \mathbb{Z}^d$ that maps s_i to its coefficient vector $\phi(s_i) \in \mathbb{Z}^d$. Let $\mathsf{rot}(a_{i,j}) \in \mathbb{Z}^{d \times d}$ the anti-circulant matrix such that $\phi\big(a_{i,j} \cdot s_i \bmod (X^d + 1)\big) = \mathsf{rot}(a_{i,j}) \cdot \phi(s_i) \in \mathbb{Z}^d$. If we re-write (58) as a matrix-vector product over $\mathbb{Z}$, we obtain the relation

$$[\mathbf{A}_1 \mid \ldots \mid \mathbf{A}_M] \cdot \big[\phi(s_1) \mid \ldots \mid \phi(s_M)\big]^\top = \sum_{i=1}^{M} \mathbf{A}_i \cdot \phi(s_i) = \phi(\boldsymbol{c}) + \phi(\boldsymbol{r}) \cdot q$$

where $\mathbf{A}_i = [\mathsf{rot}(a_{i,1})^\top \mid \ldots \mid \mathsf{rot}(a_{i,N})^\top]^\top \in \mathbb{Z}_q^{Nd \times d}$ for all $i \in [M]$, $\phi(\boldsymbol{c}) \in \mathbb{Z}_q^{Nd}$, and $\phi(\boldsymbol{r}) \in \mathbb{Z}^{Nd}$. Equivalently, this can be written

$$[\mathbf{A}_1 \mid \ldots \mid \mathbf{A}_M \mid \ -q \cdot \mathbf{I}_{Nd}\ ] \cdot \underbrace{\big[\phi(s_1) \mid \ldots \mid \phi(s_M) \mid \phi(\boldsymbol{r})\big]^\top}_{\triangleq\ \bar{\boldsymbol{w}}} = \phi(\boldsymbol{c}) \tag{59}$$

In order to prove (59), a natural idea is to have the prover commit to the vector $\bar{\boldsymbol{w}} \in \mathbb{Z}^{Md+Nd}$ using a vector commitment over $\hat{\mathbb{G}}$. Then, using the batched range proof of Section D.1, it can generate short range proof that $\|\phi(s_i)\|_\infty \leq B_i$

for each $i \in [M]$ and $\|\phi(\boldsymbol{r})\|_\infty \leq dM \cdot \max_{i\in[M]}(B_i)/2$. Using the approach of [41], it can then prove that (59) holds over $\mathbb{Z}_p$,[10] where p is the order of $\hat{\mathbb{G}}$. If $p > 2Mqd\max_i(B_i)$, this ensures that (59) also holds over the integers. Instead of using the batched range proof of Section D.1, we can make the proof shorter (and spare one commitment in $\hat{\mathbb{G}}$) by directly committing to the bits of $\bar{\boldsymbol{w}}$.

For any integer $z \in \mathbb{Z}$, we define $\boldsymbol{g}_z = (1, 2, 4, \ldots, 2^{z-2}, -2^{z-1})^\top \in \mathbb{Z}^{1\times z}$ and $\boldsymbol{G}_z = \mathbf{I}_d \otimes \boldsymbol{g}_z^\top \in \mathbb{Z}^{d\times dz}$. We also define $\mathbf{G}_z^{-1}(\boldsymbol{v})$ as the decomposition function that inputs an integer vector $\boldsymbol{v} \in [-2^{z-1}, 2^{z-1}-1]^d$ and outputs a decomposition $\boldsymbol{G}_z^{-1}(\boldsymbol{v}) \in \{0,1\}^{d\cdot z}$ such that $\boldsymbol{G}_z \cdot \boldsymbol{G}_z^{-1}(\boldsymbol{v}) = \boldsymbol{v}$. Then, for each $i \in [M]$, we define

$$\tilde{\mathbf{A}}_i \triangleq \left[\mathbf{G}_{1+\log B_i}^\top \cdot \mathsf{rot}(a_{i,1})^\top \mid \ldots \mid \mathbf{G}_{1+\log B_i}^\top \cdot \mathsf{rot}(a_{i,N})^\top\right]^\top \in \mathbb{Z}_q^{Nd\times d(1+\log B_i)}$$

and we prove that

$$\underbrace{\left[\begin{array}{ccc|c} \tilde{\mathbf{A}}_1 & \ldots & \tilde{\mathbf{A}}_M & -q\cdot\left(\mathbf{I}_N \otimes \boldsymbol{G}_{1+\log B_r}\right)\end{array}\right]}_{\triangleq\ \tilde{\mathbf{A}}} \cdot \underbrace{\left[\begin{array}{c} \boldsymbol{s}_1 \\ \vdots \\ \boldsymbol{s}_M \\ \hline \boldsymbol{r}_1 \\ \vdots \\ \boldsymbol{r}_N \end{array}\right]}_{\triangleq\ \tilde{\boldsymbol{w}}} = \phi(\boldsymbol{c}), \tag{60}$$

where we set $\boldsymbol{s}_i = \boldsymbol{G}_{1+\log B_i}^{-1}(\phi(s_i)) \in \{0,1\}^{d\cdot(1+\log B_i)}$ for each $i \in [M]$, and $\boldsymbol{r}_i = \boldsymbol{G}_{1+\log B_r}^{-1}(\phi(r_i)) \in \{0,1\}^{d\cdot(1+\log B_r)}$ for each $i \in [N]$.

The prover will thus commit to the bit-decomposition $\tilde{\boldsymbol{w}} \in \{0,1\}^D$ of the witness, where $D = d\cdot(\sum_{i=1}^N(1+\log B_i) + N(1+\log B_r))$. In order to prove that relation (60) holds modulo p (and thus also over $\mathbb{Z}$ since both members have infinity norm smaller than $p/2$), the prover will use a random $\boldsymbol{\theta} \in \mathbb{Z}_p^{Nd}$ (derived from a random oracle) and prove that the committed $\tilde{\boldsymbol{w}} \in \{0,1\}^D$ satisfies $\boldsymbol{\theta}^\top \cdot \tilde{\mathbf{A}} \cdot \tilde{\boldsymbol{w}} = \boldsymbol{\theta}^\top \cdot \phi(\boldsymbol{c}) \bmod p$. If $\tilde{\mathbf{A}} \cdot \tilde{\boldsymbol{w}} \neq \phi(\boldsymbol{c}) \bmod p$, then we have $\boldsymbol{\theta}^\top \cdot (\tilde{\mathbf{A}} \cdot \tilde{\boldsymbol{w}} - \phi(\boldsymbol{c})) = 0 \bmod p$ with probability $1/p$. Proving $\boldsymbol{\theta}^\top \cdot \tilde{\mathbf{A}} \cdot \tilde{\boldsymbol{w}} = \boldsymbol{\theta}^\top \cdot \phi(\boldsymbol{c})$ is doable using one element of $\mathbb{G}$ as explained in the introduction.[11]

G.1 Description

In the description below, the CRS does not depend on a specific public key, but we allow it to depend on upper bounds on the RLWE dimension d, the modulus q of (57) and the infinity norms $\{B_i\}_{i=1}^M$. The reason is that they impact the

[10] Here, $x \bmod p$ is defined as the value $y \in (-p/2, p/2)$ such that $y \equiv x \pmod p$

[11] We actually prove (60) using the linear map commitment of Lai and Malavolta [78, Appendix D.2]. While their scheme is only proven weakly function-binding (as defined in [78]) in the random oracle model, it can be proven strongly function-binding in the AGM+ROM and it still allows us to prove simulation-extractability in the AGM+ROM.

dimension n of committed vectors and/or the order of the pairing-friendly group. Therefore the $\mathsf{CRS\text{-}Gen}$ algorithm inputs an upper bound $\bar{d}$ for the dimension, an upper bound $\bar{q}$ for the modulus and maximal values $\bar{B}_i$ for the infinity norm bounds B_i to be proven. The prover is allowed to choose a different dimension $d \leq \bar{d}$, a different noise bound $B_i \leq \bar{B}_i$, and a different modulus $q \leq \bar{q}$ in each proof. For simplicity, we assume that each norm bound B_i is a power of two.

$\mathsf{CRS\text{-}Gen}(1^\lambda, 1^{\bar{d}}, 1^{\bar{q}}, 1^{\bar{M}}, 1^{\bar{N}}, \{1^{\bar{B}_i}\}_{i=1}^{\bar{N}})$: Given a security parameter λ, a maximal dimension $\bar{d} \in \mathsf{poly}(\lambda)$, integers $\bar{N}, \bar{M} \in \mathsf{poly}(\lambda)$, $\bar{q} \in \mathsf{poly}(\lambda)$, $\bar{B}_i \in \mathsf{poly}(\lambda)$, set $B_r \triangleq \bar{d}\bar{M} \cdot \max_{i\in[M]}(\bar{B}_i)/2$ and do the following.

1. Generate asymmetric pairing-friendly groups $(\mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T)$ of prime order $p > \max(2^{l(\lambda)}, 2\bar{M}\bar{q}\bar{d}\max_i(\bar{B}_i))$, for some polynomial $l : \mathbb{N} \to \mathbb{N}$. Let $n > \bar{d} \cdot (\sum_{i=1}^{\bar{N}}(1 + \log \bar{B}_i) + \bar{N}(1 + \log B_r))$.
2. Pick a random $\alpha \xleftarrow{R} \mathbb{Z}_p$ and compute $g_1, \ldots, g_n, g_{n+2}, \ldots, g_{2n} \in \mathbb{G}$ as well as $\hat{g}_1, \ldots, \hat{g}_n \in \hat{\mathbb{G}}$, where $g_i = g^{(\alpha^i)}$ for each $i \in [2n] \setminus \{n+1\}$ and $\hat{g}_i = \hat{g}^{(\alpha^i)}$ for each $i \in [n]$.
3. Choose hash functions $H, H_t : \{0,1\}^* \to \mathbb{Z}_p^n$, $H_{\mathrm{agg}} : \{0,1\}^* \to \mathbb{Z}_p^3$ and $H_{\mathsf{lmap}} : \{0,1\}^* \to \mathbb{Z}_p^{\bar{N}\bar{d}+1}$ that will be modeled as random oracles.

Output the CRS

$$\mathsf{pp} = \Big((\mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T), g, \hat{g}, \{g_i\}_{i\in[2n]\setminus\{n+1\}}, \{\hat{g}_i\}_{i\in[n]}, \mathbf{H} = \{H, H_t, H_{\mathrm{agg}}, H_{\mathsf{lmap}}\}\Big).$$

$\mathsf{Prove}_{\mathsf{pp}}(\mathbf{x}, \mathbf{w})$: Given a statement $\mathbf{x} = (q, d, M, N, \{B_i\}_{i=1}^N, \{\boldsymbol{a}_i\}_{i=1}^M, \boldsymbol{c})$ consisting of dimensions $d \leq \bar{d}$, $M \leq \bar{M}$, $N \leq \bar{N}$, a modulus $q \leq \bar{q}$, vectors of ring elements $\{\boldsymbol{a}_i \in R_q^N\}_{i=1}^M$, $\boldsymbol{c} \in R_q^N$, and norm bounds $B_i \leq \bar{B}_i$, as well as a witness $\mathbf{w} = (s_1, \ldots, s_M) \in R^M$ such that (57) holds with $\|s_i\|_\infty \leq B_i$ for each $i \in [M]$, do the following.

1. Compute polynomials $(r_1, \ldots, r_N) \in R^N$ such that $\|r_i\|_\infty \leq B_r$ for each $i \in [N]$ and satisfying (58). Encode $(s_1, \ldots, s_M)$ and $(r_1, \ldots, r_N)$ as

$$\tilde{\boldsymbol{w}} = \left[\ \boldsymbol{s}_1^\top \mid \ldots \mid \boldsymbol{s}_M^\top \mid\ \boldsymbol{r}_1^\top \ \mid\ \ldots \ \mid\ \boldsymbol{r}_N^\top\ \right]^\top \in \{0,1\}^D,$$

   using bit decompositions $\boldsymbol{s}_i = \boldsymbol{G}_{1+\log B}^{-1}(\phi(s_i)) \in \{0,1\}^{d(1+\log B_i)}$ for each $i \in [M]$ and $\boldsymbol{r}_i = \boldsymbol{G}_{\log d}^{-1}(\phi(r_i)) \in \{0,1\}^{d\cdot(1+\log B_r)}$ for each $i \in [N]$, where $D = d \cdot (\sum_{i=1}^N (1 + \log B_i) + N(1 + \log B_r))$.
2. Commit to $\boldsymbol{w} \triangleq (\tilde{\boldsymbol{w}} \mid \boldsymbol{0}^{n-D}) = (w_1, \ldots, w_D, 0, \ldots, 0) \in \{0,1\}^n$ and prove that $\boldsymbol{w}$ is binary. Namely,
   a. Choose $\gamma \xleftarrow{R} \mathbb{Z}_p$ and compute $\hat{C} = \hat{g}^\gamma \cdot \prod_{j=1}^D \hat{g}_j^{w_j}$.

b. Compute $\boldsymbol{y} = (y_1, \ldots, y_n) = H\big(\mathbf{x}, \hat{C}\big) \in \mathbb{Z}_p^n$. Next, choose $\gamma_y \xleftarrow{R} \mathbb{Z}_p$ and compute

$$C_y = g^{\gamma_y} \cdot \prod_{j=1}^{D} g_{n+1-j}^{y_j \cdot w_j}$$

Then, compute $\boldsymbol{t} = (t_1, \ldots, t_n) = H_t(\boldsymbol{y}, \mathbf{x}, \hat{C}, C_y) \in \mathbb{Z}_p^n$.

c. Using (12), compute $\pi_{eq} \in \mathbb{G}$ such that

$$\frac{e(\prod_{i=1}^{n} g_{n+1-i}^{t_i \cdot y_i}, \hat{C})}{e(C_y, \prod_{i=1}^{n} \hat{g}_i^{t_i})} = e(\pi_{eq}, \hat{g}). \tag{61}$$

which shows that C_y commits to the (reversed) product $\boldsymbol{y} \circ \boldsymbol{w} \in \mathbb{Z}_p^n$.

d. Compute $\pi_y = C_y^{\gamma} \cdot \prod_{i=1}^{n} \left(g_i^{\gamma_y} \cdot \prod_{j \in [n] \setminus \{i\}} g_{n+1-j+i}^{y_j \cdot (w_j - 1)} \right)^{w_i}$ such that

$$e\big(C_y \cdot \prod_{j=1}^{n} g_{n+1-j}^{-y_j}, \hat{C}\big) = e(\pi_y, \hat{g}) \tag{62}$$

which shows that $\sum_{i=1}^{n} y_i \cdot w_i \cdot (w_i - 1) = 0$.

3. Compute $\bar{\boldsymbol{\theta}} = H_{\mathsf{lmap}}\big(\mathbf{x}, \hat{C}, C_y\big) \in \mathbb{Z}_p^{\bar{N}\bar{d}+1}$ and define $\tilde{\mathbf{A}} \in \mathbb{Z}^{Nd \times D}$ and $\phi(\boldsymbol{c}) \in \mathbb{Z}^{Nd}$ as in (60). Let $\boldsymbol{\theta} \in \mathbb{Z}_p^{Nd+1}$ the first $Nd+1$ entries of $\bar{\boldsymbol{\theta}}$.
4. Parse $\boldsymbol{\theta}$ as $\boldsymbol{\theta} = \big(\boldsymbol{\theta}_0^{\top} \mid \delta_\theta\big)^{\top}$, with $\boldsymbol{\theta}_0 \in \mathbb{Z}_p^{Nd}$. Let $t_\theta = \boldsymbol{\theta}_0^{\top} \cdot \phi(\boldsymbol{c}) \bmod p$ and $\boldsymbol{a}_\theta^{\top} = \boldsymbol{\theta}_0^{\top} \cdot \tilde{\mathbf{A}} \bmod p$. Generate a proof $\pi_\theta \in \mathbb{G}$ satisfying

$$e\left(\prod_{k=1}^{D} g_{n+1-k}^{\boldsymbol{a}_\theta[k]},\ \hat{C} \right) \cdot e(g_1, \hat{g}_n)^{-t_\theta} = e(\pi_\theta, \hat{g}) \tag{63}$$

by computing $\pi_\theta = \prod_{k=1}^{D} (g_{n+1-k}^{\gamma} \cdot \prod_{j \in [D] \setminus [k]} g_{n+1-k+j}^{w_j})^{\boldsymbol{a}_\theta[k]}$.

5. Compute $(\delta_{eq}, \delta_y, \delta_\theta) = H_{\mathrm{agg}}(\mathbf{x}, \hat{C}, C_y) \in \mathbb{Z}_p^3$ and an aggregated proof

$$\pi = \pi_y^{\delta_y} \cdot \pi_{eq}^{\delta_{eq}} \cdot \pi_\theta^{\delta_\theta}.$$

Output the final proof $\boldsymbol{\pi} = \big(\hat{C}, C_y, \pi\big)$.

$\mathsf{Verify}_{\mathsf{pp}}\big(\mathbf{x}, \boldsymbol{\pi}\big)$: Given a statement $\mathbf{x} = (q, d, M, N, \{B_i\}_{i=1}^{N}, \{\boldsymbol{a}_i\}_{i=1}^{M}, \boldsymbol{c})$ and a candidate $\boldsymbol{\pi}$, return 0 if $\boldsymbol{\pi}$ does not parse properly. Otherwise,

1. Compute $(\delta_{eq}, \delta_y, \delta_\theta) = H_{\mathrm{agg}}(\mathbf{x}, \hat{C}, C_y) \in \mathbb{Z}_p^3$, $\boldsymbol{y} = H(\mathbf{x}, \hat{C}) \in \mathbb{Z}_p^n$, $\boldsymbol{t} = H_t(\boldsymbol{y}, \mathbf{x}, \hat{C}, C_y) \in \mathbb{Z}_p^n$.
2. Compute $\bar{\boldsymbol{\theta}} = H_{\mathsf{lmap}}\big(\mathbf{x}, \hat{C}, C_y\big) \in \mathbb{Z}_p^{\bar{N}\bar{d}+1}$ and let $\boldsymbol{\theta} = \big(\boldsymbol{\theta}_0^{\top} \mid \delta_\theta\big)^{\top} \in \mathbb{Z}_p^{Nd+1}$ the first $Nd+1$ coordinates of $\bar{\boldsymbol{\theta}}$. Compute $t_\theta = \boldsymbol{\theta}_0^{\top} \cdot \phi(\boldsymbol{c}) \in \mathbb{Z}_p$ and $\tilde{\boldsymbol{a}}_\theta^{\top} = \boldsymbol{\theta}_0^{\top} \cdot \tilde{\mathbf{A}} \in \mathbb{Z}_p^D$. Define $\boldsymbol{a}_\theta^{\top} = (\tilde{\boldsymbol{a}}_\theta^{\top} \mid \mathbf{0}^{n-D}) \in \mathbb{Z}_p^n$.

3. Return 1 if the following equality holds and 0 otherwise:

$$e(\pi, \hat{g}) = e\Big(C_y^{\delta_y} \cdot \prod_{i=1}^{n} g_{n+1-i}^{(\delta_{eq} \cdot t_i - \delta_y) \cdot y_i + \delta_\theta \cdot \boldsymbol{a}_\theta[i]},\ \hat{C}\Big) \cdot e\Big(C_y, \prod_{i=1}^{n} \hat{g}_i^{\delta_{eq} \cdot t_i}\Big)^{-1} \cdot e(g_1, \hat{g}_n)^{-t_\theta \cdot \delta_\theta}. \quad (64)$$

Correctness. Equation (64) is obtained by aggregating (61), (62), and (63) using randomness $(\delta_{eq}, \delta_y, \delta_\theta)$. The correctness of (61)-(62) can be shown as in Section 4 while (63) is a special case of the verification equation of the inner product functional commitment of [80] (recalled in the introduction, cf. (1)).

Efficiency. We note that a 256-bit p is more than enough to satisfy the constraint $p > 2M \cdot qd \max_i(B_i)$ since d is typically 1024 or 2048, $q \approx 2^{64}$, and M is a small constant (concrete numbers are given in Supplementary Material G.3).

The CRS is comprised of $2n$ elements of $\mathbb{G}$ and n elements of $\hat{\mathbb{G}}$. As in PointProofs [61], the verifier does not need $\{g_i\}_{i=n+2}^{2n}$, which are only used by the prover. The proof only consists of one element of $\hat{\mathbb{G}}$ and two elements of $\mathbb{G}$. Compared to the most efficient simulation-extractable variant [4] of Groth's SNARK [67], our proofs are shorter by one element of $\hat{\mathbb{G}}$. This matches the optimal proof size of the simulation-extractable SNARK of Groth and Maller [68], which is significantly more expensive than [67] in terms of prover time and CRS size (see, e.g., [4] for detailed comparisons among them).

In terms of computation, $\pi = \pi_y^{\delta_y} \cdot \pi_{eq}^{\delta_{eq}} \cdot \pi_\theta^{\delta_\theta}$ can be computed using $2n$ exponentiations. At first, computing the corresponding exponents seems to require $O(n^2)$ multiplications over $\mathbb{Z}_p$, which can be quite expensive for a very large n. Fortunately, these exponents can be obtained via two products of degree-n polynomials,[12] using only $O(n \cdot \log n)$ $\mathbb{Z}_p$-multiplications for a suitable prime p. At step 4, the prover computes a product $\boldsymbol{a}_\theta^\top = \boldsymbol{\theta}_0^\top \cdot \tilde{\mathbf{A}} \bmod p$, which takes time $O(Nd \cdot D)$ in general. When it comes to proving many natural statements in structured lattices, the matrix $\tilde{\mathbf{A}}$ has a special structure allowing to compute $\boldsymbol{\theta}_0^\top \cdot \tilde{\mathbf{A}}$ using only $O(d \cdot \log d)$ multiplications in $\mathbb{Z}_p$, as explained in Supplementary Material G.4. The prover's cost is thus dominated by $3n$ exponentiations in $\mathbb{G}$ and a product of $D = d \cdot (\sum_{i=1}^{N}(1 + \log B_i) + N(1 + \log B_r))$ elements in $\hat{\mathbb{G}}$. The verifier computes 3 pairings and n exponentiations in each source group.

The scheme is not fully succinct since the number of exponentiations at the verifier grows with the length of the witness. On the prover side, however, it enables significant savings compared to R1CS-based SNARKs as the number of exponentiations only grows with the size of the witness, rather than the size of the arithmetic circuit that computes the encryption function. Indeed, the number of ring operations in the encryption algorithm does not affect the number of

[12] These polynomial products are implicitly computed in the exponent by the pairings in the right-hand-side member of (64). One of these two products is much faster to compute as it involves a polynomial of which almost all coefficients are binary.

exponentiations in the argument system.

In Supplementary Material G.3, we provide concrete proof/CRS sizes together with an estimation of the prover's complexity when it comes to proving the validity of a ciphertext in the LPR cryptosystem [85]. For such a statement, we provide a detailed comparison with SNARKs [67] providing similarly short proofs. Our construction is shown advantageous in applications (e.g., [93]) that seek to decrease the prover's computational effort, even at the cost of increasing the verifier's. We also provide a comparison with [41].

In Supplementary Material G.6, we describe a variant where the verifier computes a constant number of exponentiations and the task computing the expensive multi-exponentiations is shifted from the verifier to the prover.

G.2 Security

We first describe a simple zero-knowledge simulator.

Theorem 5. *The above non-interactive argument is perfectly zero-knowledge.*

Proof. To simulate a proof for a statement $\mathbf{x} = (q, d, M, N, \{B_i\}_{i=1}^{N}, \{\boldsymbol{a}_i\}_{i=1}^{M}, \boldsymbol{c})$, we can use the trapdoor $\alpha \in \mathbb{Z}_p$ of the CRS as follows. First, the simulator samples $\gamma, r, \gamma_y \xleftarrow{R} \mathbb{Z}_p$ and computes $\hat{C} = \hat{g}^{\gamma}$ and $C_y = g^{\gamma_y}$ as commitments to the all-zeroes vector. Next, it computes the polynomial

$$P[X] = \gamma \cdot \gamma_y \cdot \delta_y + \gamma \cdot \sum_{i=1}^{n} \Big((\delta_{eq} \cdot t_i - \delta_y) \cdot y_i + \delta_\theta \cdot \boldsymbol{a}_\theta[i] \Big) \cdot X^{n+1-i} - \gamma_y \cdot \delta_{eq} \cdot \sum_{i=1}^{n} t_i \cdot X^i - t_\theta \cdot \delta_\theta \cdot X^{n+1}$$

for which the right-hand-side member of (64) can be written $e(g, \hat{g})^{P(\alpha)}$. Using the secret exponent $\alpha \in \mathbb{Z}_p$, the simulator can then simulate a proof by computing $\pi = g^{P(\alpha)}$. It is easy to see that the resulting tuple $\boldsymbol{\pi} = (\hat{C}, C_y, \pi)$ is distributed as a real proof since the commitments $\hat{C}$ and C_y are uniformly distributed in their group and $\pi \in \mathbb{G}$ is uniquely determined by $(\hat{C}, C_y)$ and the aggregation coefficients. □

We note that the zero-knowledge simulator of Theorem 5 is not trapdoor-less [51] as it relies on the trapdoor of the CRS to simulate proofs. On the other hand, it works in the standard model, without relying on random oracles. In the proof of Theorem 6, we describe a trapdoor-less simulator that does not use the trapdoor of the CRS, but rather proceeds by programming the random oracles.

Theorem 6. *If the $(2n, n)$-DLOG assumption holds, the above non-interactive argument provides simulation-extractability in the algebraic group model and in the random oracle model.*

Proof. In the AGM+ROM model, we show that, under the $(2n, n)$-DLOG assumption, there is an extractor that can extract a witness from any adversarially-generated proof $\boldsymbol{\pi}$ and statement $\mathbf{x} = (q, d, M, N, \{B_i\}_{i=1}^N, \{\boldsymbol{a}_i\}_{i=1}^M, \boldsymbol{c})$. Concretely, we show an algorithm $\mathcal{B}$ that either extracts a witness or solves an $(2n, n)$-DLOG instance by computing $\alpha \in \mathbb{Z}_p$ from $\{(g, g_1, \ldots, g_{2n}), (\hat{g}_1, \ldots, \hat{g}_n)\}$, where $g_i = g^{(\alpha^i)}$ and $\hat{g}_i = \hat{g}^{(\alpha^i)}$ for all i.

The problem instance $\{(g, g_1, \ldots, , g_{2n}), (\hat{g}_1, \ldots, \hat{g}_n)\}$ is used to define pp. Note that $g_{n+1} = g^{(\alpha^{n+1})}$ is not included in pp although it is part of $\mathcal{B}$'s input. Our reduction/extractor $\mathcal{B}$ interacts with $\mathcal{A}$ as follows.

Queries: At any time, $\mathcal{A}$ can provide $\mathbf{x} = (q, d, M, N, \{B_i\}_{i=1}^N, \{\boldsymbol{a}_i\}_{i=1}^M, \boldsymbol{c})$ and ask for a simulated proof that $\boldsymbol{c} \in R_q^N$ is a valid ciphertext for the public key $(\boldsymbol{a}_1, \ldots, \boldsymbol{a}_M)$. To generate such a proof, the reduction $\mathcal{B}$ defines the public-key-dependent matrix $\tilde{\mathbf{A}} \in \mathbb{Z}^{Nd \times D}$ and the ciphertext-dependent vector $\phi(\boldsymbol{c}) \in \mathbb{Z}^{Nd}$ as in (60). It chooses $\boldsymbol{\theta}_0 \xleftarrow{R} \mathbb{Z}_p^{Nd}$ and $\delta_\theta \xleftarrow{R} \mathbb{Z}_p$ and computes $\tilde{\boldsymbol{a}}_\theta^\top = \boldsymbol{\theta}_0^\top \cdot \tilde{\mathbf{A}} \bmod p$ and $t_\theta = \boldsymbol{\theta}_0^\top \cdot \phi(\boldsymbol{c}) \bmod p$. We note that the first component $\tilde{\boldsymbol{a}}_\theta[1] \in \mathbb{Z}_p$ of $\tilde{\boldsymbol{a}}_\theta \in \mathbb{Z}_p^D$ is non-zero with overwhelming probability (as we may assume that the first column $\tilde{\mathbf{A}}[1]$ of $\tilde{\mathbf{A}} \in \mathbb{Z}^{Nd \times D}$ is non-zero). If $\boldsymbol{\theta}_0^\top \cdot \tilde{\mathbf{A}}[1] = \boldsymbol{\theta}_0^\top \cdot \phi(\boldsymbol{c})$, $\mathcal{B}$ can generate a real proof using the witness $\tilde{\boldsymbol{w}} = (1, 0, \ldots, 0)$. We thus assume $\boldsymbol{\theta}_0^\top \cdot \tilde{\mathbf{A}}[1] \neq \boldsymbol{\theta}_0^\top \cdot \phi(\boldsymbol{c})$, so that $\mathcal{B}$ can compute a non-binary $\boldsymbol{w} = (w_1 \mid \mathbf{0}^{n-1}) \in \mathbb{Z}_p^n$ satisfying the equation

$$\boldsymbol{a}_\theta^\top \cdot \boldsymbol{w} = t_\theta \bmod p,$$

where $\boldsymbol{a}_\theta^\top = (\tilde{\boldsymbol{a}}_\theta^\top \mid \mathbf{0}^{n-D})$. It commits to $\boldsymbol{w}$ by computing $\hat{C} = \hat{g}^\gamma \cdot \hat{g}_1^{w_1}$ for a random $\gamma \xleftarrow{R} \mathbb{Z}_p$. Next, $\mathcal{B}$ simulates other proof elements as follows.

1. Choose random vectors $\boldsymbol{\delta} = (\delta_{eq}, \delta_y, \delta_\theta) \xleftarrow{R} \mathbb{Z}_p^3$, $\boldsymbol{y} = (y_1, \ldots, y_n) \xleftarrow{R} \mathbb{Z}_p^n$, $\boldsymbol{t} = (t_1, \ldots, t_n) \xleftarrow{R} \mathbb{Z}_p^n$.
2. Set $z_n = y_1$ and find an arbitrary $(z_1, \ldots, z_{n-1}) \in \mathbb{Z}_p^{n-1}$ such that

$$\sum_{i=2}^{n} t_i \cdot z_{n+1-i} = t_1 \cdot y_1 \cdot (w_1 - 1).$$

3. Choose a random $z_0 \xleftarrow{R} \mathbb{Z}_p$ and compute a simulated commitment

$$C_y = g^{z_0} \cdot \prod_{i=1}^{n} g_i^{z_i}.$$

4. If one of the random oracle values $H_{\mathrm{agg}}(\mathbf{x}, \hat{C}, C_y)$, $H(\mathbf{x}, \hat{C})$, $H_{\mathsf{lmap}}\big(\mathbf{x}, \hat{C}, C_y\big)$ or $H_t(\boldsymbol{y}, \mathbf{x}, \hat{C}, C_y)$ was already defined, then abort and report failure. Otherwise, set $\boldsymbol{y} = H(\mathbf{x}, \hat{C})$, $\boldsymbol{t} = H_t(\boldsymbol{y}, \mathbf{x}, \hat{C}, C_y)$, $\boldsymbol{\delta} = H_{\mathrm{agg}}(\mathbf{x}, \hat{C}, C_y)$ and $\bar{\boldsymbol{\theta}} = H_{\mathsf{lmap}}\big(\mathbf{x}, \hat{C}, C_y\big) \in \mathbb{Z}_p^{\bar{N}\bar{d}+1}$ for a random vector $\bar{\boldsymbol{\theta}} \in \mathbb{Z}_p^{\bar{N}\bar{d}+1}$ whose first $Nd + 1$ components are $(\boldsymbol{\theta}_0 \mid \delta_\theta)$.

5. Define the polynomials

$$\begin{aligned}
Q_y[X] &= \Big(\sum_{i=0}^{n} z_i \cdot X^i - \sum_{i=1}^{n} y_i \cdot X^{n+1-i}\Big) \cdot \Big(\gamma + w_1 \cdot X\Big) \\
&= \Big(z_0 + \sum_{i=1}^{n} \big(z_{n+1-i} - y_i\big) \cdot X^{n+1-i}\Big) \cdot \Big(\gamma + w_1 \cdot X\Big) = \sum_{i=0}^{n+1} \sigma_i \cdot X^i \\
Q_{eq}[X] &= \Big(\gamma + w_1 \cdot X\Big) \cdot \Big(\sum_{i=1}^{n} t_i \cdot y_i \cdot X^{n+1-i}\Big) \\
&\qquad -\Big(\sum_{i=0}^{n} z_i \cdot X^i\Big) \cdot \Big(\sum_{i=1}^{n} t_i \cdot X^i\Big) = \sum_{j=0}^{2n} e_j \cdot X^j, \\
Q_\theta[X] &= \Big(\sum_{k=1}^{D} \boldsymbol{a}_\theta[k] \cdot X^{n+1-k}\Big) \cdot \Big(\gamma + w_1 \cdot X\Big) - t_\theta \cdot X^{n+1} = \sum_{i=0}^{n+1} \zeta_i \cdot X_i
\end{aligned}$$

Their degree-$(n+1)$ coefficients are

$$\begin{aligned}
\sigma_{n+1} &= w_1 \cdot (z_n - y_1) = 0 \\
e_{n+1} &= w_1 t_1 y_1 - \sum_{i=1}^{n} t_i \cdot z_{n+1-i} = t_1 \cdot y_1 \cdot (w_1 - 1) - \sum_{i=2}^{n} t_i \cdot z_{n+1-i} = 0 \\
\zeta_{n+1} &= \boldsymbol{a}_\theta[1] \cdot w_1 - t_\theta = 0
\end{aligned}$$

due to the definition of committed $\boldsymbol{w} = (w_1, \ldots, w_n)$ and $\boldsymbol{z} = (z_1, \ldots, z_n)$.

6. Define the polynomial

$$Q_{\text{agg}}[X] = \delta_{eq} \cdot Q_{eq}[X] + \delta_y \cdot Q_y[X] + \delta_\theta \cdot Q_\theta[X] = \sum_{i=0}^{2n} \eta_i \cdot X^i$$

for which $\eta_{n+1} = 0$ by construction. Compute

$$\pi = \prod_{i=1, i \neq n+1}^{2n} g_i^{\eta_i} \tag{65}$$

using $(g, \{g_i\}_{i \in [2n] \setminus \{n+1\}})$ and return the simulated proof $\boldsymbol{\pi} = (\hat{C}, C_y, \pi)$.

The proof $\boldsymbol{\pi}$ has the same distribution as an output of the simulator in the proof of Theorem 5. Indeed, π is uniquely determined by $\mathbf{x}$, $(\hat{C}, C_y)$ and the $\mathbb{Z}_p$-elements $\boldsymbol{y}, \boldsymbol{t}$, and $\boldsymbol{\delta}$ in the right-hand-side member (64). Moreover, while the committed $\boldsymbol{w}, \boldsymbol{z} \in \mathbb{Z}_p^n$ are programmed in a special way, they are completely independent of $\mathcal{A}$'s view due to the randomness γ and z_0 in $(\hat{C}, C_y)$.

Consequently, the simulation is perfect, unless one of the random oracles has to be programmed on an input where it was previously defined. If Q_S (reps. Q_H) is the number of queries made by $\mathcal{A}$ to the simulator (resp. to random oracles), this happens with probability $\leq (Q_S + Q_H) \cdot Q_H / p$.

Output: When $\mathcal{A}$ halts, it outputs $\mathbf{x} = (q, d, M, N, \{B_i\}_{i=1}^N, \{\boldsymbol{a}_i\}_{i=1}^M, \boldsymbol{c})$ and a valid proof $\boldsymbol{\pi} = \big(\hat{C}, C_y, \pi\big)$. Let $\tilde{\mathbf{A}} \in \mathbb{Z}^{Nd \times D}$ the matrix obtained by encoding $\{\boldsymbol{a}_i \in R_q^N\}_{i=1}^M$ in (60).

Since we are in the AGM, $\mathcal{A}$ must provide representations of $\hat{C}$ w.r.t to the set of all $\hat{\mathbb{G}}$-elements that it could observe during the game. It also has to provide representations of C_y and π w.r.t to all $\mathbb{G}$-elements that it was allowed to see. Since the simulator used by $\mathcal{B}$ is algebraic, it also knows a representation of each simulated $C_y^{(i)}$ and $\pi^{(i)}$ w.r.t $(g, \{g_i\}_{i \in [2n] \setminus \{n+1\}})$. It also knows a representation of each simulated $\hat{C}^{(i)}$ w.r.t $(\hat{g}, \{\hat{g}_i\}_{i \in [n]})$. From $\mathcal{A}$'s output and the randomness of the simulation, $\mathcal{B}$ can therefore compute scalars $\{(\psi_i, z_i) \in \mathbb{Z}_p^2\}_{i \in [0,2n] \setminus \{n+1\}}$ and $\{w_i \in \mathbb{Z}_p\}_{i \in [0,n]}$ such that

$$\hat{C} = \prod_{i=0}^{n} \hat{g}_i^{w_i}, \qquad C_y = \prod_{i=0, i \neq n+1}^{2n} g_i^{z_i}, \qquad \pi = \prod_{i=0, i \neq n+1}^{2n} g_i^{\psi_i},$$

where $g_0 = g$ and $\hat{g}_0 = \hat{g}$.

If the representation $\boldsymbol{w} = (w_0, w_1, \ldots, w_n) \in \mathbb{Z}_p^n$ of the commitment $\hat{C}$ satisfies the conditions

(i) $w_k \in \{0, 1\}$ for all $k \in [1, D]$;

(ii) $\tilde{\mathbf{A}} \cdot \tilde{\boldsymbol{w}} = \phi(\boldsymbol{c}) \bmod p$, where $\tilde{w} = (w_1, \ldots, w_D) \in \mathbb{Z}_p^D$;

then $\mathcal{B}$ can use the bits $(w_0, w_1, \ldots, w_D) \in \{0,1\}^D$ to reconstruct witnesses $s_1, \ldots, s_M \in \mathbb{Z}[X]/(X^d + 1)$ such that $\|s_i\|_\infty \leq B_i$ for all $i \in [M]$ and (57) holds, meaning that $\{s_i\}_{i=1}^M$ are valid outputs for the knowledge extractor. We now assume that at least one of the conditions (i)-(ii) does not hold.

Solving $(2n, n)$-DLOG: We remark that a non-trivial valid proof $\boldsymbol{\pi}$ cannot recycle $(\mathbf{x}, \hat{C}, C_y)$ from a a simulated proof: That is, for all $i \in [Q_S]$, we must have $(\mathbf{x}, \hat{C}, C_y) \neq (\mathbf{x}^{(i)}, \hat{C}^{(i)}, C_y^{(i)})$ since the right-hand-side member of (64) is uniquely determined by $(\mathbf{x}^{(i)}, \hat{C}^{(i)}, C_y^{(i)})$ and it in turn determines a unique valid proof element $\pi^{(i)} \in \mathbb{G}$ in the left-hand-side member. This implies that $(\delta_y, \delta_{eq}, \delta_\theta) = H_{\text{agg}}(\mathbf{x}, \hat{C}, C_y)$ is not one of the hashes programmed by the simulator and neither are $\boldsymbol{t} = H_t(\boldsymbol{y}, \mathbf{x}, \hat{C}, C_y)$, and $\bar{\boldsymbol{\theta}} = H_{\mathsf{Imap}}\big(\mathbf{x}, \hat{C}, C_y\big)$.

Let the vector $\boldsymbol{a}_\theta^\top = (\tilde{\boldsymbol{a}}_\theta^\top \mid \mathbf{0}^{n-D}) \in \mathbb{Z}_p^n$ defined in the Verify algorithm. From the algebraic representations of $\mathcal{A}$'s commitments and proof π, $\mathcal{B}$ can compute

$$P_\theta[X] = \Big(\sum_{k=1}^{D} \boldsymbol{a}_\theta[k] \cdot X^{n+1-k}\Big) \cdot \Big(\sum_{i=0}^{n} w_i \cdot X^i\Big) - t_\theta \cdot X^{n+1} = \sum_{i=0}^{n+D} \omega_i \cdot X^i$$

as well as the polynomials

$$\begin{aligned}
P_y[X] &= \Big(\sum_{i=0, i\neq n+1}^{2n} z_i \cdot X^i - \sum_{i=1}^{n} y_i \cdot X^{n+1-i} \Big) \cdot \Big(\sum_{i=0}^{n} w_i \cdot X^i \Big) \\
&= \Big(z_0 + \sum_{i=1}^{n} \big(z_{n+1-i} - y_i \big) \cdot X^{n+1-i} + \sum_{i=n+2}^{2n} z_i \cdot X^i \Big) \cdot \Big(\sum_{i=0}^{n} w_i \cdot X^i \Big) \\
&= \sum_{i=0}^{3n} \gamma_i \cdot X^i \\
P_{eq}[X] &= \Big(\sum_{i=0}^{n} w_i \cdot X^i \Big) \cdot \Big(\sum_{i=1}^{n} t_i \cdot y_i \cdot X^{n+1-i} \Big) \\
&\qquad - \Big(\sum_{i=0, i\neq n+1}^{2n} z_i \cdot X^i \Big) \cdot \Big(\sum_{i=1}^{n} t_i \cdot X^i \Big) = \sum_{j=0}^{3n} \beta_j \cdot X^j,
\end{aligned}$$

for which the left-hand-side members of (61)-(63) can be written $e(g, \hat{g})^{P_{eq}(\alpha)}$, $e(g, \hat{g})^{P_y(\alpha)}$, and $e(g, \hat{g})^{P_\theta(\alpha)}$, respectively.

The right-hand-side member of (64) can be written $e(g, \hat{g})^{P_{\text{agg}}(\alpha)}$, where $P_{\text{agg}}[X]$ is the polynomial

$$P_{\text{agg}}[X] = \delta_y \cdot P_y[X] + \delta_{eq} \cdot P_{eq}[X] + \delta_\theta \cdot P_\theta[X] = \sum_{i=0}^{3n} \nu_i \cdot X^i$$

In $P_{\text{agg}}[X]$, the coefficient ν_{n+1} of the degree-$(n+1)$ term can be written

$$\begin{aligned}
\nu_{n+1} = \delta_y \cdot \underbrace{\sum_{i=1}^{n} \big(z_{n+1-i} - y_i \big) \cdot w_i}_{\triangleq\ \gamma_{n+1}} \ + \ \delta_{eq} \cdot \underbrace{\sum_{i=1}^{n} t_i \cdot (w_i \cdot y_i - z_{n+1-i})}_{\triangleq\ \beta_{n+1}} \\
\delta_\theta \cdot \underbrace{\Big(\sum_{k=1}^{D} \boldsymbol{a}[k] \cdot w_k - t_\theta \Big)}_{\triangleq\ \omega_{n+1}},
\end{aligned}$$

where $\boldsymbol{\rho} \triangleq (\gamma_{n+1}, \beta_{n+1}, \omega_{n+1})$ is the vector containing the coefficients of the degree-$(n+1)$ terms of $(P_y[X], P_{eq}[X], P_\theta[X])$.

We now argue that, if one of the conditions (i)-(ii) does not hold, the probability to have $\nu_{n+1} = 0$ is negligible. This follows from the following observations:

- The probability to have $\boldsymbol{\rho} = \mathbf{0}$ is negligible. First, if $z_{n+1-i} \neq w_i \cdot y_i$ for some $i \in [n]$, we have $\beta_{n+1} = 0$ with probability $1/p$ since $\boldsymbol{t} = H_t(\boldsymbol{y}, \mathbf{x}, \hat{C}, C_y)$ is derived *after* the choice of $\boldsymbol{y}$, $\{w_i\}_{i=0}^n$ and $\{z_i\}_{i \in [0,2n] \setminus \{n+1\}}$. Now, if we assume that $z_{n+1-i} = w_i \cdot y_i$ for all $i \in [n]$, then we have

$$\gamma_{n+1} = \sum_{i=1}^{n} y_i \cdot (w_i - 1) \cdot w_i,$$

which vanishes with probability $1/p$ if there exists $i \in [n]$ such that $w_i \notin \{0,1\}$. This can be seen by distinguishing two cases:

a. If $\boldsymbol{y} = H(\mathbf{x}, \hat{C})$ was programmed when answering a simulation query, we can only have $\gamma_{n+1} = 0$ with probability $1/p$ since the simulator programmed $(w_1, \ldots, w_n)$ so has to have

$$\gamma_{n+1} = \sum_{i=1}^{n} y_i \cdot w_i \cdot (w_i - 1) = y_1 \cdot w_1 \cdot (w_1 - 1)$$

where $w_1 \notin \{0,1\}$ and $y_1 \in_R \mathbb{Z}_p$. This captures the case of an adversary attempting to re-use the components $(\mathbf{x}, \hat{C}) = (\mathbf{x}^{(i)}, \hat{C}^{(i)})$ of a simulated proof $\boldsymbol{\pi}^{(i)} = (\hat{C}^{(i)}, C_y^{(i)}, \pi^{(i)})$ with a modified $C_y \neq C_y^{(i)}$.

b. If $H(\mathbf{x}, \hat{C})$ was not programmed by the simulator, then $\boldsymbol{y} = H(\mathbf{x}, \hat{C})$ was defined after $\mathcal{B}$ obtained the scalars $\{w_i\}_{i=0}^n$ underlying $\hat{C}$. We then have the equality $\sum_{i=1}^n y_i \cdot (w_i - 1) \cdot w_i = 0$ with probability $1/p$ over the random choice of $\{y_i\}_{i=1}^n$.

If none of the previous events occurs, we have $w_i \in \{0,1\}$ for all $i \in [D]$. Then, we are left with bounding the probability that $\omega_{n+1} = 0$ when condition (ii) does not hold. In this case, we have $\boldsymbol{\theta}_0^\top \cdot (\tilde{\mathbf{A}} \cdot \tilde{\boldsymbol{w}} - \phi(\boldsymbol{c})) = 0 \bmod p$ with probability $1/p$ since $\bar{\boldsymbol{\theta}} = H_{\mathsf{lmap}}\big(\mathbf{x}, \hat{C}, C_y\big)$ is defined after $\tilde{\mathbf{A}}$ and $\phi(\boldsymbol{c})$.

- If $\boldsymbol{\rho} \neq \mathbf{0}$, then we have $\nu_{n+1} \neq 0$ with overwhelming probability $1 - 1/p$ since the aggregation coefficients $\boldsymbol{\delta} \triangleq (\delta_{eq}, \delta_y, \delta_\theta) = H_{\mathrm{agg}}(\mathbf{x}, \hat{C}, C_y)$ and $(\boldsymbol{\theta}_0 \mid \delta_\theta) = H_{\mathsf{lmap}}\big(\mathbf{x}, \hat{C}, C_y\big)$ are chosen uniformly *after* the choice of $\{w_i\}_{i=0}^n$, $\{z_i\}_{i \in [0,2n] \setminus \{n+1\}}$ and $\mathbf{x}$, which determine $\boldsymbol{\rho}$. Therefore, the probability to have $\langle \boldsymbol{\delta}, \boldsymbol{\rho} \rangle = 0 \bmod p$ is $1/p$.

If $\nu_{n+1} \neq 0$, $\mathcal{B}$ can compute $\alpha \in \mathbb{Z}_p$ using the algebraic representation of π as in the proof of Theorem 2. □

G.3 Efficiency Comparisons for Proving the Validity of Ring LWE Ciphertexts

We consider a special case of the statement in (57) which corresponds to a proof of validity of an LPR ciphertext [85]. For this specific concrete statement, we compare our approach with a generic use of SNARKs for arithmetic circuits.

Let a statement consisting of a public key $(a, b) \in R_q^2$ and an LPR ciphertext $(c_1, c_2) = (a \cdot r + e_1, b \cdot r + e_2 + \Delta \cdot m) \in R_q^2$, where $\Delta = \lfloor q/2 \rfloor$ and $m \in R/(2R)$ is the plaintext.[13] We consider a prover willing to convince a verifier that there exist

[13] We consider a parameter setting where the secret r is chosen so that $\|r\|_\infty = 1$ an the noise is sampled from a Gaussian with larger standard deviation.

$m \in R/(2R)$, $r \in R/(2R)$, and noise terms $e_1, e_2 \in R$ of norm $\|e_1\|_\infty, \|e_2\|_\infty \leq B$ such that

$$\begin{bmatrix} a & & 1 & \\ b & \Delta & & 1 \end{bmatrix} \cdot \begin{bmatrix} r \\ m \\ e_1 \\ e_2 \end{bmatrix} = \begin{bmatrix} c_1 \\ c_2 \end{bmatrix} \bmod q \tag{66}$$

Following [41], we will prove the above statement by showing the existence of small polynomials $r, m \in R/(2R)$, $e_1, e_2 \in R$, and $r_1, r_2 \in R$ such that

$$\begin{bmatrix} a(X) & & 1 & & -q & \\ b(X) & \Delta & & 1 & & -q \end{bmatrix} \cdot \begin{bmatrix} r(X) \\ m(X) \\ e_1(X) \\ e_2(X) \\ r_1(X) \\ r_2(X) \end{bmatrix} = \begin{bmatrix} c_1(X) \\ c_2(X) \end{bmatrix} \bmod (X^d + 1)$$

with $\|e_1\|_\infty, \|e_2\|_\infty \leq B$, and $\|r_1\|_\infty, \|r_2\|_\infty \leq (d+1)/2$. Over $\mathbb{Z}$, this can be written

$$\underbrace{\begin{bmatrix} \mathsf{rot}(a) & & \mathbf{I}_d & & -q \cdot \mathbf{I}_d & \\ \mathsf{rot}(b) & \Delta \cdot \mathbf{I}_d & & \mathbf{I}_d & & -q \cdot \mathbf{I}_d \end{bmatrix}}_{\triangleq\ \bar{\mathbf{A}}} \cdot \underbrace{\begin{bmatrix} \phi(r) \\ \phi(m) \\ \phi(e_1) \\ \phi(e_2) \\ \phi(r_1) \\ \phi(r_2) \end{bmatrix}}_{\triangleq\ \bar{\boldsymbol{w}}} = \underbrace{\begin{bmatrix} \phi(c_1) \\ \phi(c_2) \end{bmatrix}}_{\triangleq\ \phi(\boldsymbol{c})}, \tag{67}$$

where $\bar{\mathbf{A}}$ is interpreted as a $2d \times 6d$ matrix with coefficients in $\{-\lfloor q/2 \rfloor, \ldots, \lfloor q/2 \rfloor\}$.

The prover commits to the bits of $\bar{\boldsymbol{w}}$ and proves that

$$\underbrace{\begin{bmatrix} \mathsf{rot}(a) & & \boldsymbol{G}_{1+\log B} & & -q \cdot \boldsymbol{G}_{\log d} & \\ \mathsf{rot}(b) & \Delta \cdot \mathbf{I}_d & & \boldsymbol{G}_{1+\log B} & & -q \cdot \boldsymbol{G}_{\log d} \end{bmatrix}}_{\triangleq\ \tilde{\mathbf{A}}\ \in\ \mathbb{Z}^{2d \times D}} \cdot \underbrace{\begin{bmatrix} \phi(r) \\ \phi(m) \\ \boldsymbol{e}_1 \\ \boldsymbol{e}_2 \\ \boldsymbol{r}_1 \\ \boldsymbol{r}_2 \end{bmatrix}}_{\triangleq\ \tilde{\boldsymbol{w}}} = \underbrace{\begin{bmatrix} \phi(c_1) \\ \phi(c_2) \end{bmatrix}}_{\triangleq\ \phi(\boldsymbol{c})}, \tag{68}$$

where $\boldsymbol{e}_1 = \boldsymbol{G}^{-1}_{1+\log B}(\phi(e_1))$, $\boldsymbol{e}_2 = \boldsymbol{G}^{-1}_{1+\log B}(\phi(e_2))$, $\boldsymbol{r}_1 = \boldsymbol{G}^{-1}_{\log d}(\phi(r_1))$ and $\boldsymbol{r}_2 = \boldsymbol{G}^{-1}_{\log d}(\phi(r_2))$. We note that there is no need to decompose $m, r \in R/(2R)$ since they are natively binary. The prover will thus commit to the decomposition of the witness $\tilde{\boldsymbol{w}} \in \{0,1\}^D$, where $D = 2d(2 + \log B + \log d)$.

It is also interesting to consider proofs of validity in an encryption scheme proposed by Joye [72], which was designed to be used as a component of the

TFHE [32] homomorphic encryption scheme. The scheme of [72] can be seen as a variant of the LPR cryptosystem where the second ciphertext component computes an inner product over $\mathbb{Z}_q$ instead of a multiplication over R_q. For a plaintext $m \in \mathbb{Z}_p$ and a noise $e_2 \in \mathbb{Z}$, ciphertexts are of the form

$$(c_1, c_2) = \big(a \cdot r + e_1, \langle \phi(\bar{b}),\ \phi(r) \rangle + e_2 + \Delta \cdot m\big) \in R_q \times \mathbb{Z}_q$$

where $\Delta = \lfloor q/t \rfloor$ (for a plaintext modulus t) and $\phi(\bar{b}) = (b_{n-1}, \ldots, b_0) \in \mathbb{Z}_q^n$ contains the coefficients of the polynomial $b(x) = b_0 + b_1 X + \cdots + b_{n-1} X^{n-1} \in R_q$ in reversed order. Note that $c_2 \in \mathbb{Z}_q$ can be seen as the extraction of the last slot from the second component $b \cdot r + \Delta \cdot m + \mathsf{noise}$ of an LPR ciphertext since the degree-$(n-1)$ coefficient of the polynomial product $b \cdot r \in R_q$ is $\langle \phi(\bar{b}),\ \phi(r) \rangle$. In this case, relation (67) simplifies as

$$\underbrace{\begin{bmatrix} \mathsf{rot}(a) & \mathbf{I}_d & -q \cdot \mathbf{I}_d & \\ \phi(\bar{b}) & \Delta & 1 & -q \end{bmatrix}}_{\triangleq\ \bar{\mathbf{A}}} \cdot \underbrace{\begin{bmatrix} \phi(r) \\ m \\ \phi(e_1) \\ e_2 \\ \phi(r_1) \\ r_2 \end{bmatrix}}_{\triangleq\ \bar{\boldsymbol{w}}} = \underbrace{\begin{bmatrix} \phi(c_1) \\ c_2 \end{bmatrix}}_{\triangleq\ \phi(\boldsymbol{c})}. \tag{69}$$

while (68) becomes

$$\underbrace{\begin{bmatrix} \mathsf{rot}(a) & \boldsymbol{G}_{1+\log B} & -q \cdot \boldsymbol{G}_{\log d} & \\ \phi(\bar{b}) & \Delta \cdot \boldsymbol{g}_{\log t} & \boldsymbol{g}_{1+\log B} & -q \cdot \boldsymbol{g}_{\log d} \end{bmatrix}}_{\triangleq\ \tilde{\mathbf{A}} \in \mathbb{Z}^{(d+1) \times D}} \cdot \underbrace{\begin{bmatrix} \phi(r) \\ m \\ \boldsymbol{e}_1 \\ \boldsymbol{e}_2 \\ \boldsymbol{r}_1 \\ \boldsymbol{r}_2 \end{bmatrix}}_{\triangleq\ \tilde{\boldsymbol{w}}} = \underbrace{\begin{bmatrix} \phi(c_1) \\ c_2 \end{bmatrix}}_{\triangleq\ \phi(\boldsymbol{c})}, \tag{70}$$

with $\boldsymbol{e}_2 = \boldsymbol{g}_{1+\log B}^{-1}(e_2) \in \{0,1\}^{1+\log B}$ and $\boldsymbol{r}_2 = \boldsymbol{g}_{\log d}^{-1}(r_2) \in \{0,1\}^{\log d}$. The prover then commits to the decomposition of the witness $\tilde{\boldsymbol{w}} \in \{0,1\}^D$, where $D = d + \log t + (d+1)(1 + \log B + \log d)$, where t is the plaintext modulus.

G.4 Efficiency Estimations

Application to the LPR cryptosystem. In an instantiation of LPR for $\lambda = 128$, a common choice of parameters is $d = 1024$, $q \approx 2^{64}$, with binary uniform $r \in R/(2R)$ while e_1, e_2 are sampled from a discrete Gaussian distribution with standard deviation $\alpha q \approx 2^{39}$. In this case, a given noise vector $e_i \sim D_{\mathbb{Z}^d, \alpha q}$

has infinity norm $\|e_i\|_\infty \leq B = \alpha q\sqrt{\lambda} < 2^{43}$ with overwhelming probability by [84, Lemma 4.4]. The computational complexity and the CRS size are then determined by $n = D = 112640$.

In order to obtain the coefficients allowing to compute π from the generators $\{g_i\}_{i\in[2n]\setminus\{n+1\}}$, the prover has to evaluate two products of degree-n polynomials,[14] which can be done using $O(n\cdot\log n)$ multiplications. The prover also has to compute the product $\boldsymbol{a}_\theta^\top = \boldsymbol{\theta}_0^\top \cdot \tilde{\mathbf{A}} \bmod p$. In (68), each block of $\tilde{\mathbf{A}}$ has a special structure allowing to compute the matrix-vector product using $O(d\cdot\log d)$ multiplications over $\mathbb{Z}_p$. Indeed, computing $\boldsymbol{\theta}_0^\top \cdot [\mathsf{rot}(a)^\top \mid \mathsf{rot}(b)^\top]^\top$ takes $O(d\cdot\log d)$ multiplications while computing $\boldsymbol{\theta}_0^\top \cdot (\mathbf{I}_2 \otimes \boldsymbol{G}_z)$ can be done using $2dz$ *additions* over $\mathbb{Z}_p$ since $\boldsymbol{G}_z$ is of the form $\mathbf{I}_d \otimes (1, 2, 4, \ldots, 2^{z-2}, -2^{z-1})$.

Eventually, the prover's cost is dominated by 337920 exponentiations in $\mathbb{G}$ and $D + 1$ multiplications, which are used to compute $\hat{C}$. If we assume that exponentiations in $\hat{\mathbb{G}}$ are three times as expensive as in $\mathbb{G}$, the overall workload of the prover is roughly equivalent to 339150 exponentiations in $\mathbb{G}$.

Given the relatively large value of $n \approx 2^{17}$, we need to increase the group order p by about 20 bits in order to obtain a sufficient security margin against Cheon's algorithm [31]. If we use a 275-bit group order, elements of $\mathbb{G}$ (resp. $\hat{\mathbb{G}}$) can have a 374-bit (resp. 1122-bit) representation using KSS18 curves. With these parameters, the CRS size amounts to 25712 KB and proofs fit in 1870 bits.

Application to Joye's scheme. We now consider an instantiation of the scheme in [72] for parameters of interest where $d = 1024$, $q \approx 2^{64}$ and when the noise has magnitude $B \leq 2^{42}$. We assume that the plaintext modulus t has 8 bits due to compatibility constraints with the bootstrapping of TFHE. In order to encrypt 256-bit messages, we consider a packed version of the scheme allowing to encrypt $k = 32$ slots of 8-bit messages each and where all slots $\{c_{2,i} = \langle\phi(\bar{b}_i), \phi(r)\rangle + e_{2,i} + \Delta\cdot m_i\}_{i=1}^k$ share the same secret $r \in R/(2R)$ but use independent noise terms $e_{2,i}$ such that $\|e_{2,i}\|_\infty \leq B$.[15] To prove a packed version of relation (70), the prover commits to a vector $\tilde{\boldsymbol{w}}$ of dimension

$$n > D = (d + k\log t) + (d + k)(1 + \log B + \log d) = 57248.$$

Using KSS18 or BLS24 curves, this requires a CRS of 13068 KB or 14340 KB, respectively. The prover computes at most 171744 exponentiations in $\mathbb{G}$ (143000 on average if the noise is sampled from a uniform distribution over $[-B, B]$) besides 57248 multiplications in $\hat{\mathbb{G}}$.

The prover also computes 2 multiplications of polynomials with degree 57248 over a 275-bit field $\mathbb{Z}_p$ and a matrix-vector product over $\mathbb{Z}_p$ (which can be fast since the matrix is structured). For a prime p such that $p-1$ is divisible by the smallest power of 2 above n, all $\mathbb{Z}_p$-operations can be optimized using the FFT.

[14] The first product is cheaper since one of the factors is of the form $\gamma + \sum_{i=1}^D w_i \cdot X^i$ for binary $w_i \in \{0, 1\}$

[15] In this case, the matrix $\tilde{\mathbf{A}}$ in (70) is modified to have $d + k$ rows, where the last k rows encode public keys components $\{\phi(\bar{b}_i)\}_{i=1}^k$ in the lower block.

Verification requires 57248 exponentiations in each source group of the pairing.[16] Assuming multiple threads at the verifier, we can speed up its computation by splitting exponentiations in smaller batches to be processed in parallel. Also, we can reduce the cost of $\hat{\mathbb{G}}$-exponentiations by observing that the exponents $\boldsymbol{t} = (t_1, \ldots, t_n)$ do not need to be uniformly distributed over $\mathbb{Z}_p$ since they are only used to perform a batch verification in the proof of Theorem 6 (i.e., to guarantee that $\beta_{n+1} \neq 0$ w.h.p. in the expression of ν_{n+1}). By [46, Theorem 3.2], we can choose each t_i uniformly in a 128-bit interval (instead of a 275-bit one) and change the verification equation (64) into

$$e(\pi, \hat{g}) = e\Big(C_y^{\delta_y} \cdot \prod_{i=1}^{n} g_{n+1-i}^{(\delta_{eq} \cdot t_i - \delta_y) \cdot y_i + \delta_\theta \cdot \boldsymbol{a}_\theta[i]},\ \hat{C}\Big) \cdot e\Big(C_y^{\delta_{eq}}, \prod_{i=1}^{n} \hat{g}_i^{t_i}\Big)^{-1} \cdot e(g_1, \hat{g}_n)^{-t_\theta \cdot \delta_\theta},$$

which makes it faster to compute $\prod_{i=1}^{D} \hat{g}_i^{t_i}$ for 128-bit exponents $\{t_i\}_i$.

We note that, in both schemes [85,72], the NIZK argument of ciphertext validity does not impose any constraint on the modulus q of the encryption scheme (so that any NTT-friendly ring can be used). We only need to choose a sufficiently large group order p to make sure that no implicit modular reduction occurs when we want to prove relations (67) and (69) over the integers.

Possible choices of elliptic curve. In order to optimize the prover's cost, one may prefer using pairing-friendly curves enabling faster exponentiations in the first source group $\mathbb{G}$. One option is to choose $\mathbb{G}$ as a subgroup of a curve $E(\mathbb{F}_r)$, for which the base prime field $\mathbb{F}_r$ is as small as possible. In this case, the BLS24 curves [7] are good candidates as they offer the fastest exponentiations in $\mathbb{G}$ (but slower exponentiations in $\hat{\mathbb{G}}$). In order to obtain a sufficient security margin against Cheon's attack, we can choose $p = |\mathbb{G}| > 2^{275}$, in which case r has $1.25 \cdot 275 = 342$ bits. By keeping r small, we also have a short representation for group elements in $\mathbb{G}$ while elements of $\hat{\mathbb{G}}$ are typically 4 times as long as those of $\mathbb{G}$ (when they live in the twisted curve $E'(\mathbb{F}_{r^4})$). This yields a proof size of 2052 bits and a CRS size of 14340 KB.

One disadvantage of BLS24 curves is their slower arithmetic in $\hat{\mathbb{G}}$. In order to obtain a more balanced tradeoff between the costs of multi-exponentiations in $\mathbb{G}$ and $\hat{\mathbb{G}}$, one may prefer using BLS12 curves. Using BLS12-379 curves, a multi-exponentiation in $\hat{\mathbb{G}}$ with 57248 elements is computable in less than a second according to the timings given in [43, Figure 4.2] for a 256-bit group order. In order to obtain 128 bits of security and taking Cheon's attack, one may use the

[16] We note that, in applications to private smart contracts [93], this is acceptable since transaction validators can proceed in parallel, regardless of the number of validators. Moreover, a transaction is often considered valid when $2N/3$ out of N validators have verified the proof.

BLS12-446 curve from [70],[17] which yields a group order $p \approx 2^{299}$ (such that 2^{16} divides $p-1$) whereas elements of $\mathbb{G}$ (resp. $\hat{\mathbb{G}}$) fit within 446 (resp. 892) bits.

G.5 Comparisons

For the above choice of parameters in the LPR cryptosystem, we commit to vectors of dimension $n = 112640$, which translates into a prover computing 337920 exponentiations in $\mathbb{G}$ and 8196 exponentiations in $\hat{\mathbb{G}}$. In general, exponentiations in $\hat{\mathbb{G}}$ are at least 3 times as expensive as in $\mathbb{G}$ using KSS18 curves (see, e.g., [6, Table 12]). In our setting, the prover computes the equivalent of $\approx$ 339150 exponentiations in $\mathbb{G}$. In the example given in [41, Section 5.3] for a smaller modulus q, del Pino *et al.* need about 724986 exponentiations at the prover (and 200667 at the verifier). In general, their construction [41, Section 5.2] incurs up to $10n + 6\log n$ exponentiations at the prover (and $2n + 4\log n$ exponentiations at the verifier) in order to generate a proof for a vector of dimension n. Here, we only need $3n$ exponentiations in $\mathbb{G}$ (and n multiplications in $\hat{\mathbb{G}}$) at the prover and the equivalent of $4n$ $\mathbb{G}$-exponentiations at the verifier. Although we need a slightly larger group order than theirs (i.e., 275 bits vs 256), we expect our prover to be faster and our verification algorithm to be slower. On the other hand, we lose the transparent setup property of BulletProofs and we need to rely on the algebraic group model.

If we want to prove the same statement using Groth's SNARK [67] in order to obtain a similar proof size, we have to express the statement in the language of Quadratic Arithmetic Programs (QAPs) [54] and obtain a CRS size growing with the number of arithmetic constraints. Then, we run into two issues that increase the size of the arithmetic circuit. First, proving the smallness of noise terms $e_1 = (e_{1,1}, \ldots, e_{1,d})^\top, e_2 = (e_{2,1}, \ldots, e_{2,d})^\top \in \mathbb{Z}[X]/(X^d+1)$ requires to break their components into bits. Then, for each pair $(b,j) \in \{0,1\} \times [d]$, proving that $\{e_{b,j,\tau}\}_{\tau=1}^{1+\log B}$ are all bits requires $d \cdot (1+\log B)$ constraints of the form $e_{b,j,\tau} \cdot (e_{b,j,\tau} - 1) = 0 \bmod p$, each of which contributes to the number of multiplication gates. In our example, this would amount to $2d \cdot (1+\log B) = 90112$ arithmetic constraints. We also need $2d$ constraints to prove that r and m are binary. Then, we need $O(d \cdot \log d)$ additional constraints to compute the products $a \cdot r$ and $b \cdot r$ over $\mathbb{Z}_p[X]/(X^d+1)$ and map e_1, e_2 to the FFT domain. We note that the prover can pre-compute $(\mathrm{FFT}(a), \mathrm{FFT}(b))$ and $(\mathrm{FFT}(c_1), \mathrm{FFT}(c_2))$ instead of leaving it to the circuit,[18] but the circuit still needs $O(d \cdot \log d)$ multiplication gates to compute $\mathrm{FFT}(r)$, $\mathrm{FFT}(e_1)$ and $\mathrm{FFT}(e_2)$.

The second issue is that constant-size SNARKs like [67,54] are designed to handle arithmetic circuits over a large prime field $\mathbb{F}_p$ (where $p > 2^{275}$ is the order of the pairing-friendly group), whereas we need to prove a statement over a ring R_q where $q \approx 2^{64}$. As observed in [52], using finite field arithmetic to emulate arithmetic over rings induces some overhead. For example, additions in R_q may no longer be for free since adding two $\log q$-bit integers over $\mathbb{F}_p$ may

[17] See also `https://neuromancer.sk/std/bls/BLS12-446`.

[18] This allows computing $\mathrm{FFT}(a \cdot r)$ and $\mathrm{FFT}(b \cdot r)$ using $2d$ multiplications.

result in a $(1 + \log q)$-bit sum to be reduced in R_q. In [76], short-integer arithmetic is emulated over $\mathbb{F}_p$ by reducing intermediate computation values modulo q on carefully chosen occasions. In order to prove that a modular reduction $x \bmod q$ is performed correctly (when q is not a power of 2), the prover is required to provide wires $x \div q$ and $x \bmod q$, allowing the circuit to check that $x = q \cdot (x \div q) + (x \bmod q)$ and $(x \bmod q) < q$. In turn, the latter comparison requires access to the bits of $x \bmod q$, which introduces $\log q$ arithmetic constraints. Using a greedy approach that only performs one reduction modulo q per component of $(c_1, c_2) = (a \cdot r + e_1, b \cdot r + e_2 + \Delta \cdot m)$, the remainder checking technique of [76] would require $2d \cdot \log q = 131072$ constraints, thus leading to an arithmetic circuit with more than 250000 multiplication gates. To improve this, we can instead interpret the components of $(c_1, c_2) \in R_q^2$ as remainders of the long division and prove the smallness of its quotients, which is also what our construction is doing. Since each quotient has magnitude $\approx d/2$, this decreases the number of constraints from $2d \cdot \log q$ to $2d \cdot \log d$ when it comes to proving correct reductions modulo q. Overall, we estimate that the entire process would still cost $n_m = 2d(1 + \log B) + 4d + 5d \cdot \log d = 145408$ arithmetic constraints to prove the global statement.

While the number $n_m = 145408$ of multiplication gates might seem only slightly larger than our vector dimension $n = 112640$, it has a significant impact. In the SNARK, the prover has to compute $n_m = 145408$ exponentiations in $\hat{\mathbb{G}}$ (with possibly large exponents over $\mathbb{Z}_p$) besides $3n_m + (n_w - \ell_s) \approx 806913$ $\mathbb{G}$-exponentiations, where $n_w \approx 374785$ is the number of wires[19] and $\ell_s = 4d = 4096$ is the number of field elements describing the statement. In comparison, we only need 337920 exponentiations in $\mathbb{G}$ and only 410 exponentiations in $\hat{\mathbb{G}}$. If we count each exponentiation over $\hat{\mathbb{G}}$ as 3 exponentiations in $\mathbb{G}$, the SNARK of [67] computes about 1243137 equivalent $\mathbb{G}$-exponentiations. On the other hand, our verification algorithm is more demanding and computes n exponentiations in both groups $\hat{\mathbb{G}}$ and $\mathbb{G}$ when the SNARK only needs ℓ_s exponentiations in $\mathbb{G}$.

As far as the CRS size goes, the SNARK approach would cost $n_m \approx 145408$ elements of $\hat{\mathbb{G}}$ and $2n_m + (n_w - \ell_s) \approx 661505$ elements of $\mathbb{G}$. Using KSS18 curves with a 275-bit group order, it would take about 50116 KB. On the other hand, the verifier only needs to store a small part of the CRS in Groth's SNARK.

G.6 A Variant with $O(1)$ Exponentiations at the Verifier

If we increase the number of exponentiations on the prover side, we can have a verification algorithm that only computes a constant number of exponentiations and $O(n \cdot \log n)$ field operations in $\mathbb{Z}_p$. This can be done by exploiting the fact that the CRS has a similar structure to that of KZG polynomial commitments [74] and leveraging the constant verification time of KZG. If we consider the

[19] The number of wires is $n_w = n_m + n_{in} + n_{out}$, where $n_{in} = 4d + 2d + 2d(\log B + 1) + 2d \log d$ is the number of input wires and $n_{out} = 1 + 2d + 2d(\log B + 1) + 2d \log d$ is the number of output wires (where we count one output wire per bit-proving constraint).

scheme from Section G.1 where the verification equation is

$$e(\pi, \hat{g}) = e\Big(C_y^{\delta_y} \cdot \underbrace{\prod_{i=1}^{n} g_{n+1-i}^{(\delta_{eq}\cdot t_i - \delta_y)\cdot y_i + \delta_\theta \cdot \boldsymbol{a}_\theta[i]}}_{\triangleq\ C_h},\ \hat{C}\Big) \cdot e\Big(C_y^{\delta_{eq}}, \underbrace{\prod_{i=1}^{n} \hat{g}_i^{t_i}}_{\triangleq\ \hat{C}_t}\Big)^{-1} \cdot e(g_1, \hat{g}_n)^{-t_\theta \cdot \delta_\theta}, \tag{71}$$

the idea is to have the prover compute all multi-exponentiations and convince the verifier that they were correctly computed as KZG commitments (similar ideas were used in [21]). The prover runs as in Section G.1 but it also computes $\hat{C}_t = \prod_{i=1}^{n} \hat{g}_i^{t_i}$ and another helper commitment $C_h = \prod_{i=1}^{n} g_{n+1-i}^{(\delta_{eq}\cdot t_i - \delta_y)\cdot y_i + \delta_\theta \cdot \boldsymbol{a}_\theta[i]}$. Then, $(C_h, \hat{C}_t)$ are included in the proof so that the verifier can test the equality (71) while performing only 2 exponentiations in $\mathbb{G}$, one exponentiation in $\hat{\mathbb{G}}$, and one exponentiation in $\mathbb{G}_T$.

Note that $(C_h, \hat{C}_t)$ can be seen as deterministic KZG commitments to the polynomials

$$P_h[X] = \sum_{i=1}^{n} \big((\delta_{eq} \cdot t_i - \delta_y) \cdot y_i + \delta_\theta \cdot \boldsymbol{a}_\theta[i]\big) \cdot X^{n+1-i} \qquad P_t[X] = \sum_{i=1}^{n} t_i \cdot X^i, \tag{72}$$

which are computable by the verifier. To provide evidence that $(C_h, \hat{C}_t)$ were really computed as KZG commitments to the above polynomials, the prover can evaluate them on a random input z obtained by hashing $(C_h, \hat{C}_t)$ together with all other commitments and proof components. Since both polynomials are evaluated on a common input, we can generate a single aggregated evaluation proof $\pi_{\mathsf{KZG}} \in \mathbb{G}$ for the polynomial $P_h[X] + \omega \cdot P_t[X]$, for a random scalar $\omega \in \mathbb{Z}_p$. Given the challenge point $z = H_z(\mathbf{x}, \hat{C}, C_y, \pi, C_h, \hat{C}_t, \boldsymbol{y}, \boldsymbol{t}, \boldsymbol{\delta}) \in \mathbb{Z}_p$, the prover can derive a randomizer $\omega = H_\omega(\mathbf{x}, \hat{C}, C_y, \pi, C_h, \hat{C}_t, \boldsymbol{y}, \boldsymbol{t}, \boldsymbol{\delta}, z, P_h(z), P_t(z)))$ and provide an aggregated evaluation proof consisting of a single group element $\pi_{\mathsf{KZG}} \in \mathbb{G}$ satisfying

$$e(C_h \cdot g^{-p_h}, \hat{g}) \cdot e(g, \hat{C}_t \cdot \hat{g}^{-p_t})^{\omega} = e(\pi_{\mathsf{KZG}}, \hat{g}_1 \cdot \hat{g}^{-z}), \tag{73}$$

where $(p_h, p_z) = (P_h(z), P_t(z))$. Since $X - z$ divides the linear combination $(P_h[X] + \omega \cdot P_t[X]) - (P_h(z) + \omega \cdot P_t(z))$, the prover can compute π_{KZG} from $(g_1, \ldots, g_n)$ using n exponentiations in $\mathbb{G}$.

The proof now consists of $\boldsymbol{\pi} = (\hat{C}, C_y, \pi, C_h, \hat{C}_t, \pi_{\mathsf{KZG}})$, which is twice as large as the original proof. The verifier checks both pairing product equations (71) and (73). The polynomial evaluations $(p_h, p_z) = (P_h(z), P_t(z))$ are not included in the proof since they can be re-computed by the verifier from $(P_h[X], P_t[X])$.

The security analysis proceeds identically to the proof of Theorem 6, except that it additionally considers the case of adversarially-generated proof containing malformed $(C_h, \hat{C}_t)$. Let the real polynomials $(P_h[X], P_t[X])$ defined in (72). From the algebraic representation of commitments $(C_h, \hat{C}_t)$, the reduction can compute polynomials $Q_h[X]$ and $Q_t[X]$ such that $C_h = g^{Q_h(\alpha)}$ and $\hat{C}_t = \hat{g}^{Q_t(\alpha)}$. If either C_h or $\hat{C}_t$ is malformed, we have $(Q_h[X], Q_t[X]) \neq (P_h[X], P_t[X])$. This implies $Q_h[X] + \omega \cdot Q_t[X] \neq P_h[X] + \omega \cdot P_t[X]$ with overwhelming probability since $(C_h, \hat{C}_t, \boldsymbol{y}, \boldsymbol{t}, \boldsymbol{\delta})$ are included in the inputs of H_z and completely determine $(P_h[X], P_t[X])$ in (72) and $(Q_h[X], Q_t[X])$ (via the algebraic representation of C_h and $\hat{C}_t$). The KZG verification equation (73) then implies

$$(Q_h(\alpha) - p_h) + \omega \cdot (Q_t(\alpha) - p_t) = \pi(\alpha) \cdot (\alpha - z)$$

where $\pi[X]$ is defined by the algebraic representation of $\pi_{\mathsf{KZG}} = g^{\pi(\alpha)}$. Then, the polynomial identity

$$(Q_h[X] - p_h) + \omega \cdot (Q_t[X] - p_t) = \pi[X] \cdot (X - z) \tag{74}$$

must hold unless the reduction can compute $\alpha \in \mathbb{Z}_p$ by factoring a non-zero polynomial. Since $(p_h, p_t) = (P_h(z), P_t(z))$, (74) implies

$$(Q_h(z) - P_h(z)) + \omega \cdot (Q_t(z) - P_t(z)) = 0$$

However, since we assumed $Q_h[X] + \omega \cdot Q_t[X] \neq P_h[X] + \omega \cdot P_t[X]$ at this point, this is only possible with probability $1/p$ because $z = H_z(\mathbf{x}, \hat{C}, C_y, \pi, C_h, \hat{C}_t, \boldsymbol{y}, \boldsymbol{t}, \boldsymbol{\delta})$ is defined after $(Q_h[X], Q_t[X], P_h[X], P_t[X])$. This shows that, if an algebraic cheating prover sends a malformed $(C_h, \hat{C}_t) \neq (g^{P_h(\alpha)}, \hat{g}^{P_t(\alpha)})$, it can only provide an accepting π_{KZG} (i.e., satisfying (73)) with negligible probability under the $(2n, n)$-DLOG assumption.

The total cost for the prover now amounts to $5n$ exponentiations in $\mathbb{G}$ (namely, n for each group element $(C_y, C_h, \pi_{\mathsf{KZG}})$ and another $2n$ to compute π) and n exponentiations in $\hat{\mathbb{G}}$. The proof size has now increased by a factor 2. While the prover complexity becomes roughly similar to that of a Groth16 prover for the same statement, the CRS size remains significantly smaller than in R1CS-based SNARKs. In particular, the verifier only needs to store a constant number of CRS components as in [67].