SHARP-PQ: Succinct Hash-based Arbitrary-Range Proof

***Abstract*—Succinct zero-knowledge range proof (ZKRP) asserts that a secret integer x is in a specified range, such as $[0, 2^n - 1]$ or $[A, B-1]$ for some arbitrary non-negative integers A and B, without leaking extra information about x. It is vital in various privacy-preserving systems. With the success of Bulletproofs, the deployment of succinct ZKRP is surging, especially with immutable ledgers like blockchain, where proof size directly impacts operational efficiency and storage sustainability.**

Momentously, the quest for long-term security through post-quantum resistance is still in its infancy. State-of-the-art lattice-based ZKRP proof size remains linear in n (Lyubashevsky *et al.*, CCS 20 and Couteau *et al.*, Eurocrypt 21).

Considering this unresolved impasse, we initiate the study of succinct hash-based ZKRP. We propose SHARP-PQ, which supports succinctness, arbitrary range, and optimized batch proof. Its success stems from our methodology in exploiting an under-explored homomorphism over Reed-Solomon codes. Empirically, SHARP-PQ offers at least 10× smaller proof size for multiple ranges over state-of-the-art lattice-based ZKRPs while maintaining competitive prover and verifier times.

1. Introduction

Zero-knowledge range proofs (ZKRPs) allow a prover to convince a verifier that a secretly-committed integer belongs to a specific range without leaking extra information beyond its validity. Since 80's [1], ZKRPs have played an indispensable role in various applications for secure and efficient operations, *e.g.*, confining choices of numerous votes/bids to be *aggregately counted* in e-voting/auction [2], [3], [4], [5], or the numbers of coins/tokens in a *compact* e-cash wallet/k-time anonymous credential [6], [7]. The larger the supported range, the more flexible, scalable, or efficiency-saving it is.

Recent years have seen a surge of interest in ZKRP research and deployment, boosted by the advances in succinct proofs (*i.e.*, logarithmic proof size or communication complexity and the corresponding efficient verification) and deployment of decentralized applications like smart-contracts and confidential transactions [8], [9]. These applications often allow an optimization opportunity of having a prover performing multiple proofs in a batch setting, *e.g.*, paying many (decoy) accounts and proving the validity of more than one smart-contract variable. Batching is also commonly considered in the broader setting of membership proof [10]. Some scenarios (*e.g.*, [9]) require the support of arbitrary ranges $[A, B]$ instead of $[0, 2^n - 1]$ for a fixed dimension n.

Long-term security is expected [11] by ZKRP applications with high stakes in democracy-enhancing systems or monetary values in commercial/financial applications. Most practical ZKRPs, based on discrete logarithm (DLOG) assumptions [12], [13] are not secure against quantum adversaries. A representative work is Bulletproofs [12], which has been widely used, *e.g.*, in Monero [8].

Recent lattice-based proposals [14], [15], [13] offer plausible post-quantum security in proving specialized relations about integers, covering ZKRPs. Unfortunately, they fall short of achieving succinct ZKRPs (sublinear in dimension n). The crux is the lack of lattice-based commitments with succinct opening proofs, let alone committing multiple secrets non-trivially for an efficient batch ZKRP [15]. CKLR21 [13] is the state of the art. LNS20 [15], the only existing lattice-based system with practical empirical performance reported (absent in CKLR21), takes $O(n \log n)$ FFT operations over ring $\mathbb{R}$ for both the prover ($\mathcal{P}$) and verifier ($\mathcal{V}$), which are not asymptotically better than CKLR21.

Hash-based proofs for NP-complete languages also offer post-quantum security [16], [17]. Committing to message vectors can be done by building a Merkle hash tree of their encodings. Opening of a single entry is logarithmic in the tree size; however, natural applications require opening dozens of entries in many trees, incurring a start-up cost (*i.e.*, before proving anything else) of 50-100 KB proof size [15], concretely larger than lattice-based schemes.

In short, we face the following open problem:

How to build a post-quantum succinct ZKRP with proof size concretely smaller than known latticed-based proofs and, ideally, supporting arbitrary ranges and batch proof?

1.1. Our Contribution

We initiate the study of succinct hash-based range proof. Table 1 summarizes the achievement of our proposed system SHARP-PQ over CKLR21 [13] and Bulletproofs [12]. SHARP-PQ features communication complexity poly-logarithmic in the range dimension n, while lattice-based works are linear. The verifier complexity is also poly-logarithmic, while other works are at least linear. Technically, we propose a general methodology of realizing range proofs from u-ary decomposition over Reed-Solomon (RS) codes and inner product argument (IPA), which may be of interest for range proofs based on other assumptions, proofs in general based on RS codes, or other IPA applications.

The hash-based setting poses challenges that are easy to solve in other settings. In particular, supporting arbitrary ranges instead of the typical fixed range $[0, 2^n - 1]$ becomes non-trivial due to the absence of direct transformation between vector and integer. Recall the absence of non-trivial

TABLE 1. COMPARISON OF CKLR21, BULLETPROOFS, AND OURS

	CKLR21	Bulletproofs	SHARP-PQ
Assumption	LWE/SIS	DLOG	Hash
Post-quantum	yes	no	yes
Proof size	$O(n)\,\lvert\mathbb{R}\rvert$	$O(\log n)\,\lvert\mathbb{G}\rvert$ $+O(\log n)\,\lvert\mathbb{F}\rvert$	$O(\log n)\,\lvert\mathbb{F}\rvert$ $+O(\log^2 n)\,\lvert\mathsf{H}\rvert$
$\mathcal{P}$'s time	$O(n)\,\mathbb{R}_{\mathsf{op}}$	$O(n)\,\mathbb{G}_{\mathsf{exp}}$	$O(n \log n)$ FFT
$\mathcal{V}$'s time	$O(n)\,\mathbb{R}_{\mathsf{op}}$	$O(n)\,\mathbb{G}_{\mathsf{exp}}$	$O(\log^2 n)\,\mathbb{F}_{\mathsf{op}}$ $+O(\log^2 n)\,\mathsf{H}$
Proof size for t ranges	$O(nt)\,\lvert\mathbb{R}\rvert$	$O(\log nt)\,\lvert\mathbb{G}\rvert$ $+O(\log nt)\,\lvert\mathbb{F}\rvert$	$O(t \log n)\,\lvert\mathbb{F}\rvert$ $+O(\log^2 n)\,\lvert\mathsf{H}\rvert$

TABLE 2. MAJOR PARAMETERS AND THEIR SIZE IN PRACTICE

Notation	Meaning	Bit Size or Time
$\mathbb{R}_{\mathsf{op}}$	ring operation	no experiments [13]
$\lvert\mathbb{R}\rvert$	ring size	32 [15] or $\leq$190 [13]
$\lvert\mathbb{G}\rvert$, $\lvert\mathbb{F}\rvert$	group and field size	128 (Bulletproofs) [12]
(Below are for SHARP-PQ, FFT is for a 2^{20}-length vector)		
FFT	fast Fourier transformation	0.12s
$\mathbb{F}_{\mathsf{op}}$, H	field and hash operations	36.30ns, 0.1ms
$\lvert\mathsf{F}\rvert$, $\lvert\mathsf{H}\rvert$	field size, hash length	64, 256

batch proof in LNS20 over multiple secrets due to the involved mathematical structure of lattice-based commitments; our methodology exploits a key feature of queries to multiple RS codes to support efficient batch processing. Such an explicit connection may own independent interest.

SHARP-PQ is proven secure in the random oracle model (ROM) from lightweight cryptography [16], without relying on lattice/number-theoretic assumptions. It is an interactive argument of knowledge with a transparent setup, which can be made non-interactive using the BCS transformation [18].[1]

Table 3 illustrates the practical advantages of SHARP-PQ. It has a 10-20× shorter proof size and a better verifier time than LNS20, albeit 1-2× slower in prover time. It is a good trade-off as the running time is only a few hundred milliseconds and could be reduced by advances in computing platforms, while ramifications of the proof size are enduring in scenarios like distributed ledgers where proofs are stored indefinitely. Compared with CKLR21, our proof size is 10-40× shorter. Admittedly, Bulletproofs feature a much smaller proof size according to the existing experimental results. Nevertheless, the running time of SHARP-PQ is at least 10× faster. Moreover, fairly comparing systems with and without post-quantum security is inherently perplexing.

1. Similar to other post-quantum works [16], [19], [14], [13], we prove the security of SHARP-PQ in the ROM based on the hardness of a quantum-safe problem. Similar to [20], we show the round-by-round soundness [21] of SHARP-PQ. It has been proven that the hash-based non-interactive argument obtained via the BCS transformation is secure in the quantum ROM [22] if the underlying interactive argument has this soundness [21].

TABLE 3. PERFORMANCE IMPROVEMENT OF SHARP-PQ

	LNS20 [15]	CKLR21 [13]	Bulletproofs [12]
Prover time	1-2× worse	data unavailable	10× better
Verifier time	1-2× better	data unavailable	10× better
Proof size	10-20× better	10-40× better	10-100× worse

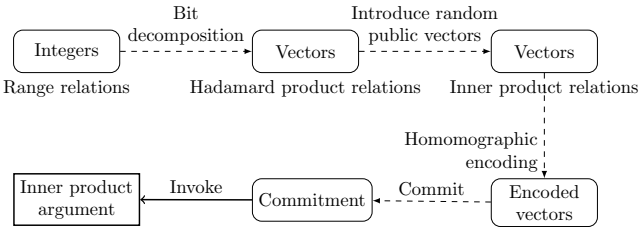


Figure 1. A framework of range proofs based on u-ary decomposition

2. Technical Overview

2.1. u-ary Decomposition Framework for Ranges

Figure 1 outlines the major steps in our framework for a series of "transformations" for range proofs based on u-ary decomposition, which is implicit in Bulletproofs [12]. To prove an integer V with upper bound u^m is in $[0, u^n - 1]$, a prover first decomposes V into its representation vector $\boldsymbol{v} \in \mathbb{F}^m$ on field $\mathbb{F}$ and commits to $\boldsymbol{v}$ and $\boldsymbol{v'}$:

$$\boldsymbol{v'} = \boldsymbol{v} \circ (\boldsymbol{v} - \boldsymbol{1^m}) \circ \cdots \circ (\boldsymbol{v} - \boldsymbol{x^m}), x \leftarrow u - 1$$

where $\boldsymbol{c^m} = (c, c, \ldots, c) \in \mathbb{F}^m$. The prover needs to show:

$$\boldsymbol{v} \circ (\boldsymbol{1^{m-n}} || \boldsymbol{0^n}) = \boldsymbol{0^m}, \quad \boldsymbol{v'} = \boldsymbol{0^m}, \tag{1}$$

where $||$ means vector concatenation. The first equation guarantees that the $m - n$ entries in $\boldsymbol{v}$ are all 0s. The second shows that each entry in $\boldsymbol{v}$ is an element in the set $\{0, 1, \ldots, u-1\}$. The prover thus transforms the range relations into equivalent Hadamard product relations.

The Hadamard relations are then converted into two inner product relations defined by a public random $\boldsymbol{r} \in \mathbb{F}^m$:

$$\langle \boldsymbol{v} \circ (\boldsymbol{1^{m-n}} || \boldsymbol{0^n}), \boldsymbol{r} \rangle = 0, \quad \langle \boldsymbol{v'}, \boldsymbol{r} \rangle = 0. \tag{2}$$

Vectors $\boldsymbol{v}$ and $\boldsymbol{v'}$ are committed using vector commitments, and put into an inner product argument (IPA).

We next emphasize two aspects of the framework.

(1) The encoding should endow an additional attribute to the encoded vectors to enable a verifier to check the inner product relation by querying only *partial* elements of the whole vector, leading to succinctness. Current lattice-based ZKRPs use commitments [23], [24] without encoding, which are not succinct. In DLOG [25], [12], $\boldsymbol{v}$ can be encoded by a Pedersen hash $\boldsymbol{g^v}$, where $\boldsymbol{g} = (g_1, g_2, \ldots, g_m)$ and $g_i = g^{\mu_i}$, $i \in [m]$, for group generator g. The linear relation between $\boldsymbol{v}$ and $(\mu_1, \mu_2, \ldots, \mu_m)$ guarantees the knowledge soundness [12] with logarithmic communication.

Consider committing $\boldsymbol{v} \in \mathbb{F}^m$ directly using a Merkle tree to prove $\langle \boldsymbol{v}, \boldsymbol{r} \rangle = c$ for public $\boldsymbol{r}$ and c, it is not succinct

since all entries on $\boldsymbol{v}$ need to be opened. If a single one is not given, the inner product can be any element other than c with high probability. Hash-based IPAs (*e.g.*, Virgo [17] and other generalized ZKPs [19], [16]) use RS code, which has a property [26] critical to succinctness, *i.e.*, succinct queries to an oracle suffice to determine whether the oracle is a valid codeword with high probability.

(2) Typically, the inputs of IPA are $\mathsf{Encode}[\boldsymbol{v}]$ and $\mathsf{Encode}[\boldsymbol{v'}]$ as codewords of $\boldsymbol{v}$ and $\boldsymbol{v'}$, but the verifier needs additional protocols and hence overhead to check the consistency between the two codewords. Recall that the encoding method in Bulletproofs has additive homomorphism, *i.e.*, given $\boldsymbol{g}$ and the codeword $\boldsymbol{g^v}$, one can compute codeword $\boldsymbol{g^{v-1^m}}$ via $\boldsymbol{g^v}/\boldsymbol{g^{1^m}}$. However, it does not have Hadamard homomorphism to construct $\boldsymbol{g^{v\circ(v-1^m)}}$ given $\boldsymbol{g}$ and $\boldsymbol{g^v}$.

For binary-based ($u = 2$) ranges in Equation (2), Bulletproofs avoids the Hadamard product of $\boldsymbol{v}$ and $\boldsymbol{v-1^m}$ by transforming $\langle \boldsymbol{v}\circ(\boldsymbol{v-1^m}), \boldsymbol{r}\rangle = 0$ into $\langle \boldsymbol{v}, (\boldsymbol{v-1^m})\circ\boldsymbol{r}\rangle = 0$. Bulletproofs then finds a non-trivial method to build $\boldsymbol{g}^{(\boldsymbol{v-1^m})\circ\boldsymbol{r}}$ using $\boldsymbol{g^{v-1^m}}$ and $\boldsymbol{r}$ by base transformation. However, when $u \geq 3$, Bulletproofs have to handle the Hadamard product of multiple secret vectors related to $\boldsymbol{v}$ and can not achieve a u-ary range proof without additional protocols [27]. We thus need to devise a new approach.

2.2. Techniques in SHARP-PQ

We now introduce the techniques in SHARP-PQ. Following our framework in Figure 1, we mainly focus on the encoding and hash-based IPA. To our knowledge, Virgo [17] is currently the only hash-based IPA. However, when employing Virgo, we face several challenges, as we consider range relations instead of only inner product relations. The challenges and corresponding solutions are described below.

(A) We first exploit the homomorphic properties of the RS code. The RS code has two common codeword generation methods. On one hand, an RS codeword can be generated by evaluating an encoding polynomial where the polynomial coefficient represents the message vector. This method is commonly used in consensus protocols [28], [29] as it is comparably efficient. However, although this generation method has additive homomorphism, it does not have the Hadamard homomorphism. To address this, we adopt the second method: interpret the message vector as polynomial evaluations, perform interpolation to obtain the encoding polynomial, and then evaluate the polynomial to generate the RS codeword. In this way, the RS code has both additive and Hadamard homomorphism. Now the verifier can obtain $\mathsf{Encode}[\boldsymbol{v'}]$ by computing $\mathsf{Encode}[\boldsymbol{v}]\circ\cdots\circ\mathsf{Encode}[\boldsymbol{v}-\boldsymbol{x^m}]$. Note that although this homomorphism may be used implicitly in some hash-based works [19], [30], it has not been discussed formally before, as far as we know.

(B) We next construct an efficient batch IPA. For a range proof, especially in the batch setting, there are multiple inner product relations. A straightforward way to prove these relations is to run an IPA multiple times, which incurs high computation and communication overhead. We observe that the main idea of Virgo is to transform an inner product relation into a sum-check relation for a univariate polynomial and run a univariate sum-check protocol [16]. We follow a similar idea and rely on the batch univariate sum-check protocol [16] to reduce multiple sum-check relations into a single one using a random linear combination and finally construct a batch IPA.

However, the above batch IPA is inefficient in complexity. We mainly focus on improving the complexity below.

We first reduce the communication complexity related to Merkle trees from $O(t\log n) + O(\log^2 n)$ to $O(\log^2 n)$ for instance number t and range dimension n. Note that previous works [16], [17], [30] mainly consider and improve hash-based protocols at the level of oracles instead of how to commit oracles. A straightforward way to commit the t RS codewords for multiple secret vectors is to construct a Merkle tree for every codeword. As a result, the communication overhead related to Merkle trees is linear to the number of range relations (*i.e.*, instance number). To reduce this overhead, our key observation is that if the verifier queries an entry in one of the RS codewords, it would query every entry in the same position for all the codewords. These entries would be used to construct a virtual oracle access to the combined secret codeword in the reduced univariate sum-check relation. Based on this observation, we aggregate entries in different RS codewords that would be queried together into one Merkle tree leaf. As a result, a single Merkle tree suffices, and the communication complexity related to Merkle trees is reduced.

We also propose a new method to reduce the verifier complexity from quasi-linear to linear. The current verifier complexity is $O(n\log n)$ as it needs to run FFTs to generate the RS codeword for the public vector $\boldsymbol{r}$ as in Equation (2). Virgo reduces the verifier complexity to poly-logarithmic by delegation computation [31] of FFTs. However, this would introduce additional proof size and computation for the prover. Our experiments show that the proof size caused by the delegation dominates our ZKRP. We propose another method to reduce the verifier complexity from quasi-linear to linear almost for free. Our key observation is that instead of the whole codeword, the verifier only needs to obtain $O(\lambda)$ common entries on each codeword, where λ is the security parameter. Now, the verifier can use Lagrange basis functions to compute these entries instead of FFTs, reducing the complexity from $O(n\log n)$ to $O(n)$ without additional overhead on proof size. This is at least at the same level as the state of the art.

(C) Finally, we construct a proof for arbitrary ranges. For $V \in [A, B-1]$, we first transform it to $V - A \in [0, 2^n-1] \wedge V - B + 2^n \in [0, 2^n-1]$ for $B \in (2^{n-1}, 2^n)$ inspired by [32]. Our first attempt is to invoke range proofs for $V-A$ and $V-B+2^n$. However, it is tough to construct RS codewords for these inputs using the codeword for the representation vector $\boldsymbol{v}$ of V. This is because the subtraction of integers does not equal the subtraction of representation vectors, and so does the subtraction of codewords. To prove the validity of new ranges, we introduce new secrets $C \leftarrow V - A$ and $D \leftarrow V - B + 2^n$ and construct new inner

product relations according to the definition of decimal and binary transformation. Specifically, $\langle \boldsymbol{v}-\boldsymbol{a}-\boldsymbol{c}, \boldsymbol{t}\rangle = 0$ implies $C = V - A$ for $\boldsymbol{t} = (2^{m-1}, 2^{m-2}, \ldots, 2, 1)$, and similar for D. Therefore, we can first use our batch IPA to prove the validity of C and D, then complete the arbitrary range proof by invoking a proof for C and D. Here, we again rely on the additive homomorphic property of RS codes to build $\mathsf{Encode}[\boldsymbol{v} - \boldsymbol{a} - \boldsymbol{c}]$.

3. Background

Let $\mathbb{F}$ be a finite field. Let italic letters with caps such as $\hat{f}$ represent polynomials on $\mathbb{F}$. We use bold lowercase letters such as $\boldsymbol{a}$ to represent vectors, and a_i denotes the i-th element of $\boldsymbol{a}$. Use $\boldsymbol{bi_u}$ to represent the binary representation vector of a decimal integer u. Define $\boldsymbol{c^m} = (c, c, \ldots, c) \in \mathbb{F}^m$. For a vector $\boldsymbol{a} \in \mathbb{F}^n$, denote $\boldsymbol{a}_{[:n-k]}$ by $(a_1, a_2, \ldots, a_{n-k}) \in \mathbb{F}^{n-k}$ for integers $0 < k < n$. For $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{F}^n$, we use $\langle \boldsymbol{a}, \boldsymbol{b}\rangle$ and $\boldsymbol{a} \circ \boldsymbol{b}$ to denote the inner product and Hadamard product of $\boldsymbol{a}$ and $\boldsymbol{b}$, respectively. Given a multiplicative coset H of $\mathbb{F}$, let $\hat{p}|_H$ be evaluations of $\hat{p}$ on H. Let $\hat{p}|_H[i]$ be the i-th entry of $\hat{p}|_H$.

We denote the security parameter by λ and "PPT" means probabilistic polynomial time. Use $\mathsf{negl}(\cdot)$ to denote a negligible function, which means for all polynomials $\hat{f}$, $\mathsf{negl}(k) < 1/\hat{f}(k)$ for sufficiently large integer k. Let $y \leftarrow A(x)$ represent that on input x, algorithm A outputs y. Use $y \xleftarrow{\$} S$ to denote uniformly and randomly picking y from a set S. For positive integers $m < n$, denote the set $\{1, 2, \ldots, m\}$ by $[m]$ and denote $\{m, m{+}1, \ldots, n\}$ by $[m, n]$.

Merkle tree [33] is a primitive enabling to commit a vector and open it at several indexes with a logarithmic-size proof. Using a Merkle tree to commit a vector $\boldsymbol{v}$ consists of three algorithms [17]: (a) $\mathsf{root}_{\boldsymbol{v}} \leftarrow \mathsf{MT.Commit}(\boldsymbol{v})$; (b) $(\{v_i\}_{i\in\mathcal{I}}, \pi^{\boldsymbol{v}}_{\mathcal{I}}) \leftarrow \mathsf{MT.Open}(\mathcal{I}, \boldsymbol{v})$; (c) $0/1 \leftarrow \mathsf{MT.Verify}(\mathsf{root}_{\boldsymbol{v}}, \mathcal{I}, \{v_i\}_{i\in\mathcal{I}}, \pi^{\boldsymbol{v}}_{\mathcal{I}})$, where $\mathsf{root}_{\boldsymbol{v}}$ is Merkle tree root, $\mathcal{I}$ is the query location set, and $\pi^{\boldsymbol{v}}_{\mathcal{I}}$ is the verification path. This paper uses Merkle trees constructed by collision-resistant and non-invertible hash functions.

Reed-Solomon code. Given a code rate $\rho \in (0, 1)$ and $L = \{\eta_1, \ldots, \eta_{|L|}\} \in \mathbb{F}^{|L|}$, $\mathrm{RS}[L, \rho] \in \mathbb{F}^{|L|}$ means $\{\hat{f}(\eta_1), \ldots, \hat{f}(\eta_{|L|}) | \deg(\hat{f}) < \rho|L|\}$. This paper uses L as a multiplicative coset of $\mathbb{F}$. Let H be another multiplicative coset with size $|H| = \rho|L|$ ($H \cap L = \phi$). The procedure of encoding vector $\boldsymbol{v} \in \mathbb{F}^{|H|}$ is first to find an *encoding polynomial* $\hat{v}$ with a prearranged degree such that $\hat{v}|_H = \boldsymbol{v}$. Then evaluate $\hat{v}$ on L to obtain the *codeword* of $\boldsymbol{v}$, *i.e.*, $\hat{v}|_L$. The evaluation and interpolation of encoding polynomials are achieved by the Fast Fourier Transform (FFT) and its inverse (IFFT).

3.1. Honest-Verifier Zero-Knowledge Argument

An honest-verifier zero-knowledge argument of knowledge (ZKAoK) for an NP relation $\mathcal{R}$ is a tuple of algorithms $(\mathcal{G}, \mathcal{P}, \mathcal{V})$. $\mathcal{G}$ represents a public parameter generation algorithm. $\mathcal{P}$ and $\mathcal{V}$ represent a PPT prover and verifier, respectively. $\mathcal{P}$ tries to convince $\mathcal{V}$ that $\mathcal{P}$ knows $\mathtt{w}$ such that $(\mathtt{x}, \mathtt{w}) \in \mathcal{R}$ for a public statement $\mathtt{x}$ but leaks no extra knowledge beyond the validity.

Definition 3.1 (Honest-verifier ZKAoK [17])**.** *$(\mathcal{G}, \mathcal{P}, \mathcal{V})$ is an honest-verifier ZKAoK for an NP relation $\mathcal{R}$ and its corresponding language $\mathcal{L}_{\mathcal{R}}$ if the following holds.*

- ***Completeness.*** *For every $\mathsf{pp} \leftarrow \mathcal{G}(1^\lambda)$ and every $(\mathtt{x}, \mathtt{w}) \in \mathcal{R}$, $\Pr[\langle \mathcal{P}(\mathtt{w}), \mathcal{V}\rangle(\mathsf{pp}, \mathtt{x}) = 1] = 1 - \mathsf{negl}(\lambda)$. We say $(\mathcal{G}, \mathcal{P}, \mathcal{V})$ is perfectly complete if the probability is 1.*
- ***Soundness:*** *For any PPT prover $\mathcal{P}^*$, every $\mathsf{pp} \leftarrow \mathcal{G}(1^\lambda)$ and every $\mathtt{x} \notin \mathcal{L}_{\mathcal{R}}$, $\Pr[\langle \mathcal{P}^*(), \mathcal{V}\rangle(\mathsf{pp}, \mathtt{x}) = 1] \le \mathsf{negl}(\lambda)$.*
- ***Argument of knowledge:*** *For any malicious PPT prover $\mathcal{P}^*$, there exists an expected polynomial time extractor $\mathcal{E}$ such that for every $\mathsf{pp} \leftarrow \mathcal{G}(1^\lambda)$ and any $\mathtt{x}$, $\Pr[\langle \mathcal{P}^*(), \mathcal{V}\rangle(\mathsf{pp}, \mathtt{x}) = 1 \wedge (\mathtt{x}, \mathtt{w}) \notin \mathcal{R} \,|\, \mathtt{w} \leftarrow \mathcal{E}^{\mathcal{P}^*}(\mathsf{pp}, \mathtt{x})] \le \mathsf{negl}(\lambda)$. Here $\mathcal{E}^{\mathcal{P}^*}$ means $\mathcal{E}$ has access to the entire executing process and randomness of $\mathcal{P}^*$.*
- ***Honest-verifier zero-knowledge:*** *There exists a PPT simulator $\mathcal{S}$ such that for any honest $\mathcal{V}$, every $(\mathtt{x}, \mathtt{w}) \in \mathcal{R}$ and $\mathsf{pp} \leftarrow \mathcal{G}(1^\lambda)$, $\{\langle \mathcal{P}(\mathtt{w}), \mathcal{V}\rangle(\mathsf{pp}, \mathtt{x})\} \overset{c}{\approx} \{\mathcal{S}^{\mathcal{V}}(\mathsf{pp}, \mathtt{x})\}$. Here $\mathcal{S}^{\mathcal{V}}$ denotes that the simulator $\mathcal{S}$ is given the randomness of $\mathcal{V}$ from a polynomial-size space, and $\overset{c}{\approx}$ means computationally indistinguishable.*

We call $(\mathcal{G}, \mathcal{P}, \mathcal{V})$ succinct *if the communication between $\mathcal{P}$ and $\mathcal{V}$ is poly-logarithmic to $|\mathtt{w}|$.*

3.2. Interactive Oracle Proof (IOP)

An IOP [18] with k round is a combination of interactive proof [34] and probabilistic checkable proof [35], where $\mathcal{P}$ sends oracle π_1 to $\mathcal{V}$ in the first round. In the i-th round $(1 < i \le k)$, $\mathcal{V}$ sends a random challenge m_{i-1}, then $\mathcal{P}$ returns an oracle π_i. In the end, $\mathcal{V}$, who has oracle access to $\boldsymbol{\pi} = (\pi_1, \ldots, \pi_k)$, queries several locations of each oracle and decides to accept or reject.

This paper involves two variants of IOPs are involved, *RS-encoded IOP* and *IOP of proximity*. The former is an IOP where all the oracles are RS codes. The latter is an IOP allowing a small distance between a prover's secret and the valid witness for the soundness.

3.3. Univariate Sum-check

Given two multiplicative cosets $H, L \subset \mathbb{F}$ $(|L| > |H|)$, a univariate polynomial $\hat{f}$ with degree $k > |H|$ and a claimed sum μ, the univariate sum-check protocol proves that $\sum_{a\in H} \hat{f}(a) = \mu$. In particular, a prover uniquely decomposes $\hat{f}(x)$ as $x\cdot\hat{g}(x) + \zeta + \hat{Z}_H(x)\hat{h}(x)$, where $\hat{Z}_H(x) = \prod_{a\in H}(x - a)$, $\deg(\hat{g}) < |H| - 1$ and ζ is a constant. Then given the oracle access to $\hat{f}|_L, \hat{h}|_L$, the verifier checks if $\hat{p}|_L \in \mathrm{RS}[L, (|H| - 1)/|L|]$ and $\hat{h}|_L \in \mathrm{RS}[L, (k - |H| + 1)/|L|]$, where $\hat{p}(x) = (|H|\cdot\hat{f}(x) - \mu - |H|\cdot\hat{Z}_H(x)\hat{h}(x))/x$. We recall this protocol in Appendix A.

When transforming the above RS-encoded IOP to an IOP, it remains to check that the oracles $\hat{f}|_L, \hat{h}|_L, \hat{p}|_L$ are valid RS codewords. This is achieved by the protocol called *fast Reed-Solomon IOP of proximity* (FRI) [26].

Given claimed degree $k_1, \ldots, k_t$, a multiplicative coset L and codewords $\hat{v}_1|_L, \ldots, \hat{v}_t|_L$, an FRI protocol allows a verifier with oracle access to these codewords to check whether for all $j \in [t]$, $\hat{v}_j|_L \in \mathrm{RS}[L, (k_j+1)/|L|]$. We recall the full FRI protocol in Appendix B.

Lemma 3.1. *[16, Theorem 8.1] FRI is an argument with perfect completeness. Given oracle access to evaluations of every prover's message at $\kappa = O(\ell)$ points, where $\ell = O(\lambda)$ is the query repetition number and $k = \max\{k_1, \ldots, k_t\}$, its soundness error is $O(|L|/|\mathbb{F}| + \mathsf{negl}(\ell, k/|L|))$. Its prover and verifier complexity is $O(|L| \log |L|)\mathbb{F}_o$ and $O(t \log |L|)\mathbb{F}_o + O(t \log |L| + \log^2 |L|)\mathsf{H}$, respectively. Its communication complexity is $O(t \log |L|)|\mathbb{F}| + O(t \log |L| \log^2 |L|)|\mathsf{H}|$.*

We denote the FRI protocol for the univariate sum-check protocol as $\langle \mathsf{FRI}.\mathcal{P}(\hat{f}, \hat{h}, \hat{p}), \mathsf{FRI}.\mathcal{V}^{\hat{f}|_L, \hat{h}|_L, \hat{p}|_L} \rangle (k+1, k-|H|+1, |H|-1)$.

4. Construction of SHARP-PQ

Now, we present our construction of SHARP-PQ. We first propose a batch IPA in Section 4.1, which is the core component of our protocol. We then use the batch IPA to construct a range proof for $[0, u^n - 1]$ in Section 4.2, a batch range proof in Section 4.3, and a proof for arbitrary ranges in Section 4.4. Finally, we add the zero-knowledge property for the above range proofs in Section 4.5.

4.1. A Batch Inner Product Argument

Recall in Section 2, a range proof can be constructed from an IPA, and this IPA is expected to support batch processing for vectors. Considering these vectors as evaluations of polynomials on a specific domain, all the inner product relations can be transformed into univariate sum-check relations. Note that as shown in Figure 1 and Equation 2, the encoding polynomials of these vectors can have different degrees. The batch univariate sum-check protocol [16] inspires us to construct such a batch IPA. Specifically, let the secret encoding polynomials be $\hat{v}_1, \ldots, \hat{v}_t$ with degrees $k_1, \ldots, k_t$. Let the public encoding polynomials be $\hat{r}_1, \ldots, \hat{r}_t$ with degrees $k_{t+1}, \ldots, k_{2t}$. Suppose that a prover $\mathcal{P}$ wants to prove that for all $j \in [t]$, $\sum_{a \in H} \hat{v}_j(a) \cdot \hat{r}_j(a) = y_j$. To accomplish this, the verifier $\mathcal{V}$ chooses and sends random challenges $\beta_1, \ldots, \beta_t$. Let $\hat{q} \leftarrow \sum_{j=1}^{t} \beta_j \hat{v}_j \cdot \hat{r}_j$, $\mathcal{P}$ and $\mathcal{V}$ then invoke the univariate sum-check protocol to prove $\sum_{a \in H} \hat{q}(a) = \sum_{j=1}^{t} \beta_j y_j$. Although the polynomials $\{\hat{v}_i \cdot \hat{r}_i\}_{i \in [t]}$ have different degrees, the soundness error can only be related to the maximum degree. We define the batch inner product relation before presenting the whole protocol.

Definition 4.1 (Batch inner product relation $\mathcal{R}_{\text{B-IPA}}$)**.** *We call $(\mathtt{x}, \mathtt{w}) \in \mathcal{R}_{\text{B-IPA}}$ if $\sum_{a \in H} \hat{v}_i(a)\hat{r}_i(a) = y_i$ for all $i \in [t]$, where $\mathtt{x} = (\mathbb{F}, H, L, \{k_i\}_{i \in [2t]}, \{\hat{r}_i\}_{i \in [t]}, \{y_i\}_{i \in [t]})$ and $\mathtt{w} = \{\hat{v}_i\}_{i \in [t]}$. L, H are multiplicative cosets of $\mathbb{F}$. For all $i \in [t]$, $\deg(\hat{v}_i) = k_i$ and $\deg(\hat{r}_i) = k_{i+t}$. $|L| > 2k_{\max} = 2\max\{k_i + k_{t+i}\}_{i \in [t]}$.*

Batch IPA $\langle \mathsf{IPA_B}.\mathcal{P}(\mathtt{w}), \mathsf{IPA_B}.\mathcal{V} \rangle (\mathtt{x})$ for relation $\mathcal{R}_{\text{B-IPA}}$.

1) $\mathcal{P}$ evaluates $\hat{v}_j|_L \leftarrow \mathrm{FFT}(\hat{v}_j, L)$ for all $j \in [t]$. $\mathcal{P}$ then computes $\mathsf{root}_{\mathbb{V}|_L} \leftarrow \mathsf{MT.Commit}(\mathbb{V}|_L)$. $\mathbb{V}|_L \in \mathbb{F}^{t \times |L|}$ is a matrix where the i-th row is $\hat{v}_i|_L[1], \ldots, \hat{v}_i|_L[|L|]$ and $\mathsf{MT.Commit}(\mathbb{V}|_L)$ means a commitment generation algorithm where every column of $\mathbb{V}|_L$ is put into one leaf. $\mathcal{P}$ sends $\mathsf{root}_{\mathbb{V}|_L}$ to $\mathcal{V}$.
2) $\mathcal{V}$ evaluates $\hat{r}_j|_L \leftarrow \mathrm{FFT}(\hat{r}_j, L)$ for all $j \in [t]$. $\mathcal{V}$ randomly picks $\beta_1, \ldots, \beta_t \xleftarrow{\$} \mathbb{F}$ and sends them to $\mathcal{P}$.
3) $\mathcal{P}$ computes $\hat{q} \leftarrow \sum_{j=1}^{t} \beta_j \hat{v}_j \cdot \hat{r}_j$, and decomposes $\hat{q}(x)$ as $x \cdot \hat{g}(x) + \zeta + \hat{Z}_H(x)\hat{h}(x)$. $\mathcal{P}$ computes $\hat{p}(x) \leftarrow (|H| \cdot \hat{q}(x) - \mu - |H| \cdot \hat{Z}_H(x)\hat{h}(x))/x$, $\hat{h}|_L \leftarrow \mathrm{FFT}(\hat{h}, L)$ and $\mathsf{root}_{\hat{h}|_L} \leftarrow \mathsf{MT.Commit}(\hat{h}|_L)$. $\mathcal{P}$ sends $\mathsf{root}_{\hat{h}|_L}$ to $\mathcal{V}$.
4) $\mathcal{P}$ and $\mathcal{V}$ invoke the FRI protocol $\langle \mathsf{FRI}.\mathcal{P}(\hat{q}, \hat{h}, \hat{p}), \mathsf{FRI}.\mathcal{V}^{\hat{q}|_L, \hat{h}|_L, \hat{p}|_L} \rangle (k_{\max}+1, k_{\max} - |H| + 1, |H| - 1)$. $\mathcal{V}$ outputs reject and aborts if the FRI protocol does not pass; otherwise, $\mathcal{V}$ randomly picks and sends a location set $\mathcal{I} \subset [|L|] (|\mathcal{I}| = \kappa = O(\ell))$ to $\mathcal{P}$.
5) $\mathcal{P}$ executes $(\{\mathbb{V}|_L[i]\}_{i \in \mathcal{I}}, \pi_{\mathcal{I}}^{\mathbb{V}|_L}) \leftarrow \mathsf{MT.Open}(\mathcal{I}, \mathbb{V}|_L)$ and $(\{h|_L[i]\}_{i \in \mathcal{I}}, \pi_{\mathcal{I}}^{\hat{h}|_L}) \leftarrow \mathsf{MT.Open}(\mathcal{I}, \hat{h}|_L)$, then sends all opening information to $\mathcal{V}$.
6) $\mathcal{V}$ executes $\mathsf{MT.Verify}(\mathsf{root}_{\mathbb{V}|_L}, \mathcal{I}, \{\mathbb{V}|_L[i]\}_{i \in \mathcal{I}}, \pi_{\mathcal{I}}^{\mathbb{V}|_L})$ and $\mathsf{MT.Verify}(\mathsf{root}_{\hat{h}|_L}, \mathcal{I}, \{h|_L[i]\}_{i \in \mathcal{I}}, \pi_{\mathcal{I}}^{\hat{h}|_L})$. $\mathcal{V}$ accepts if these algorithms output accept; rejects otherwise.

Theorem 4.1. *The protocol $\langle \mathsf{IPA_B}.\mathcal{P}(\mathtt{w}), \mathsf{IPA_B}.\mathcal{V} \rangle (\mathtt{x})$ is an argument of knowledge with perfect completeness and soundness error $1/|\mathbb{F}| + O(|L|/|\mathbb{F}|) + \mathsf{negl}(\ell, k_{\max}/|L|)$ in the random oracle model.*

Proof Sketch. **Completeness.** If for every $j \in [t]$, $\sum_{a \in H} v_j(a) r_j(a) = y_j$, then $\sum_{a \in H} \hat{q}(a) = \sum_{j=1}^{t} \beta_j y_j$, which is a univariate sum-check relation. By the completeness of the univariate sum-check protocol, the completeness follows. Note that due to the linearity of the RS codes, the verifier is able to construct the oracle access to $\hat{q}|_L$ according to public codewords $\hat{r}_1|_L, \ldots, \hat{r}_t|_L$ and queries to $\hat{v}_1|_L, \ldots, \hat{v}_t|_L$.

Soundness. Soundness error comes from two cases.

- **Case** $\sum_{a \in H} \hat{q}(a) = \sum_{j=1}^{t} \beta_j y_j$: Without loss of generality, we suppose that $\sum_{a \in H} \hat{v}_j(a)\hat{r}_j(a) = y'_j$ for all $j \in [t]$, and for an inconsistency set $Q \subset [t]$, $y'_q \neq y_q$ holds for $q \in Q$. For simplicity, we assume that $t \in Q$. Then, for all $\beta_1, \ldots, \beta_{t-1}$ randomly chosen by the verifier, the probability of $\sum_{j=1}^{t} \beta_j y'_j = \sum_{j=1}^{t} \beta_j y_j$ holds if and only if $\beta_t = \big(\beta_1(y_1 - y'_1) + \ldots + \beta_{t-1}(y_{t-1} - y'_{t-1})\big)/(y'_t - y_t)$. This happens with probability $1/|\mathbb{F}|$.
- **Case** $\sum_{a \in H} \hat{q}(a) \neq \sum_{j=1}^{t} \beta_j y_j$: The soundness error comes from three aspects below.
  - If the RS-encoded IOP is invalid, the verifier will always reject according to [16, Theorem 5.2].

- If the FRI is invalid, the soundness error is bounded by ϵ_{FRI} according to Lemma 3.1, where $k = k_{\max}$.
- If the root of the Merkle tree is invalid or any verification path is incorrect, the soundness error is bounded by $\mathsf{negl}(\lambda)$ according to the collision-resistant property of underlying hash functions.

Combining both cases using a union-bound argument, we can conclude the total soundness error.

Argument of knowledge. This property comes from the extractability of Merkle trees, as proved in [18, §A]. Given the root and sufficiently many verification paths, there exists an efficient procedure similar to "Valiant's extractor" [36], which could extract all the committed leaves in the Merkle tree. Once these leaves are extracted, one can use efficient decoding algorithms such as Berlekamp-Welch to obtain the secret polynomials. The argument of knowledge thus follows. We give a formal proof in Appendix C. □

A proof-size optimization. A direct way to commit $(\hat{v}_1|_L, \ldots, \hat{v}_t|_L)$ is to construct t Merkle trees for every secret encoding polynomial. We observe that in the FRI protocol, the $O(\ell)$ queries for every $\hat{v}_i|_L$ are used for constructing queries to $\hat{q}|_L, \hat{h}|_L$ and $\hat{p}|_L$. More importantly, to query $\hat{q}|_L[i]$ for some i, $\hat{v}_1|_L[i], \ldots, \hat{v}_t|_L[i]$ will be all queried. Thus, $\hat{v}_1|_L[i], \ldots, \hat{v}_j|_L[i]$ could be put into a single leaf. As a result, we only need a single Merkle tree to commit all the secret codewords instead of t trees. The communication complexity related to Merkle trees is hence reduced from $O(t \log |L|) + O(\log^2 |L|)$ to $O(\log^2 |L|)$, according to Lemma 3.1.

4.2. A Range Proof for $[0, u^n - 1]$

Now, we use the batch IPA to construct a proof for arbitrary-based ranges. Given a secret integer $V \in [0, u^n - 1]$ with upper bound u^m, we show that $\boldsymbol{v}$, the u-ary representation of V, satisfies the following for $x \leftarrow u - 1$:

$$\boldsymbol{v} \circ (\boldsymbol{v} - \mathbf{1}^{\boldsymbol{m}}) \circ \cdots \circ (\boldsymbol{v} - \boldsymbol{x}^{\boldsymbol{m}}) = \boldsymbol{v} \circ (\mathbf{1}^{\boldsymbol{m-n}} || \mathbf{0}^{\boldsymbol{n}}) = \mathbf{0}^{\boldsymbol{m}}. \quad (3)$$

Further, by random linear combination, it suffices to prove

$$\langle \boldsymbol{v} \circ (\boldsymbol{v} - \mathbf{1}^{\boldsymbol{m}}) \circ \cdots \circ (\boldsymbol{v} - \boldsymbol{x}^{\boldsymbol{m}}), \boldsymbol{r} \rangle = \langle \boldsymbol{v}, \boldsymbol{r}_{[:m-n]} || \mathbf{0}^{\boldsymbol{n}} \rangle = 0 \quad (4)$$

with soundness error $1/|\mathbb{F}|$ for random vector $\boldsymbol{r}$.

Equation (4) consists of two inner product relations and can be batched by our batch IPA. Let the encoding polynomials of $\boldsymbol{v}, \boldsymbol{r}, \boldsymbol{r}_{[:m-n]} || \mathbf{0}^{\boldsymbol{n}}$ be $\hat{v}, \hat{r}, \hat{s}$, respectively. Fix the encoding polynomial of $\boldsymbol{w} \leftarrow \boldsymbol{v} \circ (\boldsymbol{v} - \mathbf{1}^{\boldsymbol{m}}) \circ \cdots \circ (\boldsymbol{v} - \boldsymbol{x}^{\boldsymbol{m}})$ as $\hat{w} \leftarrow \hat{v} \cdot (\hat{v} - 1) \cdots (\hat{v} - x)$. We define the range relation and formally present the protocol below.

Definition 4.2 (Range relation $\mathcal{R}_{\mathsf{RP}}$)**.** *We call* $(\mathtt{x}, \mathtt{w}) \in \mathcal{R}_{\mathsf{RP}}$ *if* $V \in [0, u^n - 1]$ *where* $\mathtt{x} = (\mathbb{F}, H, L, m, n, [0, u^n - 1])$ *and* $\mathtt{w} = V$*. L, H are multiplicative cosets of $\mathbb{F}$, $|L| = c_1|H| > (u+1)(|H| - 1)$, $m = c_2|H| \geq n$ for constants c_1 and c_2.*

Range proof $\langle \mathsf{RP}.\mathcal{P}(\mathtt{w}), \mathsf{RP}.\mathcal{V} \rangle(\mathtt{x})$ for $\mathcal{R}_{\mathsf{RP}}$.

1) $\mathcal{P}$ generates the u-ary representation vector $\boldsymbol{v}$ of V, secret polynomials $\hat{v} \leftarrow \mathsf{IFFT}(\boldsymbol{v}, H)$ and $\hat{w} \leftarrow \hat{v}(\hat{v} - 1) \cdots (\hat{v} - x)$. $\mathcal{P}$ then computes $\hat{v}|_L \leftarrow \mathsf{FFT}(\hat{v}, L)$ and $\mathsf{root}_{\hat{v}|_L} \leftarrow \mathsf{MT.Commit}(\hat{v}|_L)$. $\mathcal{P}$ sends $\mathsf{root}_{\hat{v}|_L}$ to $\mathcal{V}$.
2) $\mathcal{V}$ randomly picks $\boldsymbol{r} \xleftarrow{\$} \mathbb{F}^m$ and sends it to $\mathcal{P}$.
3) $\mathcal{P}$ and $\mathcal{V}$ compute $\hat{r} \leftarrow \mathsf{IFFT}(\boldsymbol{r}, H)$, $\boldsymbol{s} \leftarrow \boldsymbol{r}_{[:m-n]} || \mathbf{0}^{\boldsymbol{n}}$ and $\hat{s} \leftarrow \mathsf{IFFT}(\boldsymbol{s}, H)$. They then invoke the batch IPA $\langle \mathsf{IPA_B}.\mathcal{P}(\mathtt{w}_{\mathsf{IPA}}), \mathsf{IPA_B}.\mathcal{V} \rangle(\mathtt{x}_{\mathsf{IPA}})$ to prove Equation (4), where $\mathtt{x}_{\mathsf{IPA}} = \big(\mathbb{F}, H, L, \big(u(|H| - 1), |H| - 1, |H| - 1, |H| - 1\big), \big(\hat{r}, \hat{s}\big), (0, 0)\big)$, $\mathtt{w}_{\mathsf{IPA}} = (\hat{w}, \hat{v})$.
   $\mathcal{V}$ accepts if the batch IPA passes; otherwise rejects.

Theorem 4.2. *The protocol* $\langle \mathsf{RP}.\mathcal{P}(\mathtt{w}), \mathsf{RP}.\mathcal{V} \rangle(\mathtt{x})$ *is an argument of knowledge with perfect completeness and soundness error* $2/|\mathbb{F}| + O(|L|/|\mathbb{F}|) + \mathsf{negl}(\ell, (u+1)(|H| - 1)/|L|)$ *in the random oracle model.*

Proof Sketch. **Completeness.** If Equation (3) holds, Equation (4) would satisfy. The completeness hence holds due to that of the batch IPA. Note that any entry in $\hat{w}|_L$ could be constructed by querying the corresponding entry in $\hat{v}|_L$ according to the homomorphism of RS codes.

Soundness. Soundness error comes from two cases.

• **Case 1**. Both inner product relations in Equation (4) are satisfied, while at least one of the Hadamard relations in Equation (3) does not satisfy. Concretely, if $\boldsymbol{v} \circ (\boldsymbol{v} - \mathbf{1}^{\boldsymbol{m}}) \circ \cdots \circ (\boldsymbol{v} - \boldsymbol{x}^{\boldsymbol{m}}) \neq \mathbf{0}^{\boldsymbol{m}}$ or $\boldsymbol{v} \circ (\mathbf{1}^{\boldsymbol{m-n}} || \mathbf{0}^{\boldsymbol{n}}) \neq \mathbf{0}^{\boldsymbol{m}}$, Equation (4) holds with probability bounded by $1/|\mathbb{F}|$ due to the random choice of $\boldsymbol{r}$ similar to the first case in the proof of Theorem 4.1.

• **Case 2**. At least one of the inner product relations in Equation (4) does not satisfy. This soundness error is bounded by $1/|\mathbb{F}| + O(|L|/|\mathbb{F}|) + \mathsf{negl}(\ell, (u+1)(|H| - 1)/|L|)$ according to Theorem 4.1.

Combining both cases using a union-bound argument, we can conclude the total soundness error.

Round-by-round soundness. Protocol $\langle \mathsf{RP}.\mathcal{P}(\mathtt{w}), \mathsf{RP}.\mathcal{V} \rangle(\mathtt{x})$ has a stronger soundness property called round-by-round (RBR) soundness [21], and arguments with this property based on IOP are provably secure in the quantum random oracle model [22]. We recall some backgrounds and prove this property in Appendix D.

Argument of knowledge. This property follows from that of the batch IPA. After extracting polynomial $\hat{v}$, $\boldsymbol{v} \leftarrow \hat{v}|_H$ is efficiently computable and the integer V represented by $\boldsymbol{v}$ can be obtained. □

Complexity. The overhead of our range proof comes mainly from the batch IPA. The prover complexity is $O(n \log n)$ due to the FFT/IFFT operations. The communication complexity is $O(\log n)\,|\mathbb{F}| + O(\log^2 n)\,|\mathsf{H}|$ according to Lemma 3.1.

The current verifier complexity is $O(n \log n)$ due to the FFT/IFFT operations for public vectors $\boldsymbol{r}$ and $\boldsymbol{s}$. It could decrease to linear. In the range proof, the verifier only needs $\kappa = O(\ell)$ entries in $\hat{r}|_L$ instead of the whole codeword, where ℓ is the query repetition number in the FRI. Queries to $\hat{s}|_L$ are similar, which we omit below. Suppose that the verifier queries $\hat{r}(\tau_1), \ldots, \hat{r}(\tau_\kappa)$, where $\{\tau_1, \ldots, \tau_\kappa\} \subset L$. We observe that these entries are

computable in linear time using Lagrange basis functions. Specifically, $\hat{r}$ is defined by $\hat{r}(x) \leftarrow \sum_{i=1}^{|H|} \hat{\lambda}_i(x) \cdot r_i$, where $\hat{\lambda}_i(x) \leftarrow \prod_{j=1,j\neq i}^{|H|}(x - r_j)/\prod_{j=1,j\neq i}^{|H|}(r_i - r_j)$. Using the above definition, one can compute $\hat{r}(\tau_k)$ ($k \in [\kappa]$) in $O(n)$ time. The total time complexity is hence $O(n\kappa) = O(n)$, where $\kappa = O(\ell) = O(\lambda)$.

Reducing the verifier complexity to poly-logarithmic. The verifier complexity can be further reduced to $O(\log^2 n)$ if using delegations as in Virgo [17]. However, unlike Virgo, we handle a challenge vector instead of a single challenge field element. The details are listed below.

The main computation overhead of the verifier in the range proof for $[0, u^n - 1]$ includes:
(1) reading the public statement $\mathtt{x} = (\mathbb{F}, H, L, m, n, [0, u^n - 1])$, which takes $O(n)$ time as $|H|, |L| = O(n)$;
(2) picking a random vector $\boldsymbol{r} \xleftarrow{\$} \mathbb{F}^m$, takeing $O(n)$ time;
(3) computing the IFFT and FFTs for public vectors, which takes $O(n \log n)$ time;
(4) invoking the FRI, which takes $O(\log^2 n)$ time.

To reduce the verifier complexity, we need to reduce the computation complexity of (1), (2), and (3).

For (1), we observe that every multiplicative coset (we take L as an example) can be determined by three values: the size $|L|$, the $|L|$-th unit root ω, and the coset shift. Thus, a verifier only needs $O(1)$ time to read the statement. Note that the verifier does not need to compute all the elements in the coset to run the FFT/IFFT, as these FFT/IFFT operations are delegated, as explained below.

For (2), instead of spending linear time picking the random vector $\boldsymbol{r}$, a verifier can only randomly pick one element $r \in \mathbb{F}$. The prover invokes the remaining protocol with $\boldsymbol{r} \leftarrow (r, r^2, \ldots, r^m)$. The soundness error of the random linear combination would increase to $m/|\mathbb{F}|$ instead of $1/|\mathbb{F}|$.

For (3), the prover and verifier can invoke the circuit evaluation delegation protocol known as GKR protocol [31] to complete the delegation of computation of FFT/IFFT. The communication complexity of the GKR protocol is $O(d \log |C|)$, where d is the circuit depth and $|C|$ is the circuit size. The prover complexity is $O(|C|)$ and the verifier complexity is about $O(d \log |C|)$ [17].

To invoke the GKR protocol, we construct a circuit computing $\boldsymbol{r}, \boldsymbol{s}, \mathrm{IFFT}(\boldsymbol{r})$, and $\mathrm{IFFT}(\boldsymbol{s})$. $\boldsymbol{r}, \boldsymbol{s}$ can be computed using a circuit with depth $O(\log n)$ and size $O(n)$. Specially, $\boldsymbol{r}$ is computed by $\boldsymbol{r_{i+1}} \leftarrow \boldsymbol{r_i} \otimes (1, r^{2^i})$ for $i \in [0, \log m - 1]$, where $\boldsymbol{r_0} = r$, $\boldsymbol{r_{\log m}} = \boldsymbol{r}$, and $\otimes$ means tensor product. As shown [17], the IFFT and FFT computation can be described as a circuit with depth $O(\log n)$ and size $O(n \log n)$. Therefore, using the GKR protocol to delegate the computation incurs additional $O(\log^2 n)$ communication complexity, resulting in an $O(\log^2 n)$ verifier complexity.

4.3. A Batch Range Proof for Multiple Ranges

Based on Section 4.2, this section gives a batch range proof for multiple range relations. Fix integers $V_1 \in [0, u_1^{n_1} - 1], \ldots, V_t \in [0, u_t^{n_t} - 1]$ with upper bound $u_{\max}^m - 1$, where $u_{\max} = \max\{u_1, \ldots, u_t\}$ and $m \geq \max\{n_1, \ldots, n_t\}$. We show that for all $j \in [t]$,

$$\boldsymbol{v_j} \circ \cdots \circ (\boldsymbol{v_j} - \boldsymbol{x_j^m}) = \boldsymbol{v_j} \circ (\boldsymbol{1^{m-n_j}} || \boldsymbol{0^{n_j}}) = \boldsymbol{0^m}, \quad (5)$$

where $x_j \leftarrow u_j - 1$. These constraints can be converted into

$$\langle \boldsymbol{v_j} \circ \cdots \circ (\boldsymbol{v_j} - \boldsymbol{x_j^m}), \boldsymbol{r} \rangle = \langle \boldsymbol{v_j}, (\boldsymbol{r^{m-n_j}} || \boldsymbol{0^{n_j}}) \rangle = 0. \quad (6)$$

Note that Equation (6) consists of multiple inner product relations and can be proved by our batch IPA. For all $j \in [t]$, let $\hat{v}_j$ and $\hat{s}_j$ be the encoding polynomials of $\boldsymbol{v_j}$ and $\boldsymbol{r^{m-n_j}} || \boldsymbol{0^{n_j}}$, respectively. Let $\hat{w}_j \leftarrow \hat{v}_j(\hat{v}_j - 1) \cdots (\hat{v}_j - x_j)$. We define the batch range relation and formally present the batch range proof below.

Definition 4.3 (Batch range relation $\mathcal{R}_{\text{B-RP}}$)**.** *We call* $(\mathtt{x}, \mathtt{w}) \in \mathcal{R}_{\text{B-RP}}$ *if* $V_j \in [0, u_j^n - 1]$ *for all* $j \in [t]$, *where* $\mathtt{x} = \left(\mathbb{F}, H, L, m, t, \{n_1, \ldots, n_t\}, \{[0, u_j^n - 1]\}_{j\in[t]}\right)$ *and* $\mathtt{w} = (V_1, \ldots, V_t)$. L, H *are multiplicative cosets of* $\mathbb{F}$, $|L| = c_1|H| > (u_{\max} + 1)(|H| - 1)$, $m = c_2|H| \geq n$ *for constants* c_1 *and* c_2.

Batch range proof $\langle \mathsf{RP_B}.\mathcal{P}(\mathtt{w}), \mathsf{RP_B}.\mathcal{V} \rangle(\mathtt{x})$ for $\mathcal{R}_{\text{B-RP}}$.

1) $\mathcal{P}$ generates the u_j-ary representation vector $\boldsymbol{v_j}$ of V_j, secret polynomials $\hat{v}_j \leftarrow \mathrm{IFFT}(\boldsymbol{v_j}, H)$ and $\hat{w}_j \leftarrow \hat{v}_j(\hat{v}_j - 1) \cdots (\hat{v}_j - u_j - 1)$ for all $j \in [t]$. $\mathcal{P}$ computes $\mathsf{root}_{\mathbb{V}|_L} \leftarrow \mathsf{MT.Commit}(\mathbb{V}|_L)$ and sends it to $\mathcal{V}$.
2) $\mathcal{V}$ randomly picks $\boldsymbol{r} \xleftarrow{\$} \mathbb{F}^m$ and sends it to $\mathcal{P}$.
3) $\mathcal{P}$ and $\mathcal{V}$ compute $\hat{r} \leftarrow \mathrm{IFFT}(\boldsymbol{r}, H)$. They also compute $\boldsymbol{s_j} \leftarrow \boldsymbol{r}_{[:m-n_j]} || \boldsymbol{0^n}$ and $\hat{s}_j \leftarrow \mathrm{IFFT}(\boldsymbol{s_j}, H)$ for all $j \in [t]$. They invoke the batch IPA $\langle \mathsf{IPA_B}.\mathcal{P}(\mathtt{w}_{\mathsf{IPA}}), \mathsf{IPA_B}.\mathcal{V} \rangle(\mathtt{x}_{\mathsf{IPA}})$ to prove Equation (6), where $\mathtt{x}_{\mathsf{IPA}} = (\mathbb{F}, H, L, \{k_j\}_{j\in[4t]}, \{\hat{r}_j\}_{j\in[2t]}, \{y_j\}_{j\in[2t]})$ and $\mathtt{w}_{\mathsf{IPA}} = (\hat{w}_1, \ldots, \hat{w}_t, \hat{v}_1, \ldots, \hat{v}_t)$. Here, we have

$$k_{j,j\in[t]} = u_j(|H| - 1), \quad k_{j,j\in[t+1,4t]} = |H| - 1,$$
$$\hat{r}_{j,j\in[t]} = \hat{r}, \quad \hat{r}_{j,j\in[t+1,2t]} = \hat{s}_j, \quad y_{j,j\in[2t]} = 0.$$

$\mathcal{V}$ accepts if the batch IPA passes; rejects otherwise.

Theorem 4.3. *The protocol* $\langle \mathsf{RP_B}.\mathcal{P}(\mathtt{w}), \mathsf{RP_B}.\mathcal{V} \rangle(\mathtt{x})$ *is an argument of knowledge with perfect completeness and soundness error* $2/|\mathbb{F}| + O(|L|/|\mathbb{F}|) + \mathsf{negl}(\ell, (u_{\max} + 1)(|H| - 1)/|L|)$ *in the random oracle model.*

The security proof for the batch range proofs is similar to that of Theorem 4.2 and is omitted. Note that the largest degree of the polynomial invoked in the univariate sum-check protocol is $u_{\max} + 1$. Hence, the soundness holds.

Complexity. The prover complexity is $O(tn \log n)$. The verifier complexity is $O(tn \log n)$ and can be reduced to $O(tn)$ or $O(\log^2(tn))$ using the method in Section 4.2.

By using the proof size optimization method for Merkle trees in Section 4.1, the communication complexity is $O(t \log n)\,|\mathbb{F}| + O(\log^2 n)\,|\mathsf{H}|$. This is much better than repeatedly running the range proof in Section 4.2, which is $O(t \log n)\,|\mathbb{F}| + O(t \log^2 n)\,|\mathsf{H}|$.

4.4. A Range Proof for Arbitrary Ranges

This section presents a more generalized range proof to prove $V \in [A, B-1]$ for arbitrary non-negative integers A and B. Inspired by [32], if we have a range proof for $[0, u^n-1]$ as described in Section 4.2, it is feasible to handle the range $[A, B-1]$.

Specifically, suppose that $u^{n-1} < B < u^n$. To show $V \in [A, B-1]$, it suffices to prove $V \in [A, A+u^n-1] \wedge V \in [B-u^n, B-1]$ as $A > 0 > B-u^n$ and $A+u^n-1 \geq u^n-1 > B-1$. This is equivalent to $V-A \in [0, u^n-1] \wedge V-B+u^n \in [0, u^n-1]$. For simplicity, we assume $u = 2$ below.

To prove these two relations, our first try is to use the homomorphism of the RS codes and the batch range proof in Section 4.3. Specifically, given the oracle access to the codeword for the binary representation vector of V, *i.e.*, $\boldsymbol{v}$, the verifier tries to construct oracle access to the codewords of $\boldsymbol{v}-\boldsymbol{a}$ and $\boldsymbol{v}-\boldsymbol{b}+\boldsymbol{bi_{2^n}}$, where $\boldsymbol{a}, \boldsymbol{b}$, and $\boldsymbol{bi_{2^n}}$ are the binary representation vectors of A, B, and 2^n, respectively. However, taking $\boldsymbol{v}-\boldsymbol{a}$ as an example, there are not only 0 and 1 in $\boldsymbol{v}-\boldsymbol{a}$ but also $q-1$ if the underlying field is $\mathbb{Z}_q$. The positions of $0, 1$, and $q-1$ do not have an apparent regularity. This brings difficulties for the construction of Hadamard products or inner product relations, as well as the design of range proofs.

Instead of constructing the codeword of $\boldsymbol{v}-\boldsymbol{a}$ directly, the prover $\mathcal{P}$ can additionally send the commitments to the binary representation vectors $\boldsymbol{c}, \boldsymbol{d}$ of $C = V-A$ and $D = V-B+2^n$. $\mathcal{P}$ first shows that C, D are valid by proving $\langle \boldsymbol{v}-\boldsymbol{a}-\boldsymbol{c}, \boldsymbol{t}\rangle = 0$ and $\langle \boldsymbol{v}-\boldsymbol{b}+\boldsymbol{bi_{2^n}}-\boldsymbol{d}, \boldsymbol{t}\rangle = 0$, where $\boldsymbol{t} = (2^{m-1}, 2^{m-2}, \ldots, 2, 1) \in \mathbb{F}^m$. These inner product relations can be proved by the batch IPA. After that, $\mathcal{P}$ uses the batch range proof in Section 4.3 to show that C, D are in respective ranges.

Specifically, let $\boldsymbol{v}, \boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}, \boldsymbol{d}$ be the binary representation vectors of V, A, B, C, D, respectively. The prover and verifier invoke the batch IPA to prove the following inner product relations:

$$\begin{aligned}
&\langle \boldsymbol{f} \circ (\boldsymbol{f}-\boldsymbol{1^m}), \boldsymbol{r}\rangle = 0, \boldsymbol{f} \in \{\boldsymbol{v}, \boldsymbol{c}, \boldsymbol{d}\},\\
&\langle \boldsymbol{c}, \boldsymbol{r}_{[:m-n]}||\boldsymbol{0^n}\rangle = \langle \boldsymbol{d}, \boldsymbol{r}_{[:m-n]}||\boldsymbol{0^n}\rangle = 0,\\
&\langle \boldsymbol{v}-\boldsymbol{a}-\boldsymbol{c}, \boldsymbol{t}\rangle = \langle \boldsymbol{v}-\boldsymbol{b}+\boldsymbol{bi_{2^n}}-\boldsymbol{d}, \boldsymbol{t}\rangle = 0.
\end{aligned}$$

The first three equations prove $\boldsymbol{v}, \boldsymbol{c}, \boldsymbol{d}$ are all binary vectors. The following two show that C, D are consistent with V, A, B. The last two equations prove the range validity.

We first give some notations and present the whole protocol. For $\boldsymbol{\alpha} \in \{\boldsymbol{v}, \boldsymbol{c}, \boldsymbol{d}\}$, let $\hat{v}_{\boldsymbol{\alpha}}$ be the encoding polynomial of $\boldsymbol{\alpha}$, and $\hat{w}_{\boldsymbol{\alpha}} \leftarrow \hat{v}_{\boldsymbol{\alpha}}(\hat{v}_{\boldsymbol{\alpha}}-1)$. Fix $\hat{u}_{\boldsymbol{c}}$ and $\hat{u}_{\boldsymbol{d}}$ as the encoding polynomials of $\boldsymbol{v}-\boldsymbol{a}-\boldsymbol{c}$ and $\boldsymbol{v}-\boldsymbol{b}+\boldsymbol{bi_{2^n}}-\boldsymbol{d}$, respectively. Let $\hat{r}, \hat{s}, \hat{t}$ be the public encoding polynomials of vectors $\boldsymbol{r}, \boldsymbol{r}_{[:m-n]}||\boldsymbol{0^n}$ and $\boldsymbol{t}$, respectively.

Definition 4.4 (Arbitrary range relation $\mathcal{R}_{\text{A-RP}}$)**.** *We call* $(\mathtt{x}, \mathtt{w}) \in$ *if* $V \in [A, B-1]$*, where* $\mathtt{x} = (\mathbb{F}, H, L, m, n, A, B)$ *and* $\mathtt{w} = V$*. L, H are multiplicative cosets, $|L| = c_1|H| > 3(|H|-1)$ and $m = c_2|H| \geq n$ for constants c_1, c_2.*

Arbitrary range proof $\langle \mathsf{RP_A}.\mathcal{P}(\mathtt{w}), \mathsf{RP_A}.\mathcal{V}\rangle(\mathtt{x})$ for $\mathcal{R}_{\text{A-RP}}$.

1) $\mathcal{P}$ computes $C \leftarrow V-A$, $D \leftarrow V-B+2^n$, and generates the binary vectors $\boldsymbol{v}, \boldsymbol{c}, \boldsymbol{d}$. For $\boldsymbol{\alpha} \in \{\boldsymbol{v}, \boldsymbol{c}, \boldsymbol{d}\}$, $\mathcal{P}$ computes secret polynomials $\hat{v}_{\boldsymbol{\alpha}} \leftarrow \text{IFFT}(\boldsymbol{\alpha}, H)$ and $\hat{w}_{\boldsymbol{\alpha}} \leftarrow \hat{v}_{\boldsymbol{\alpha}}(\hat{v}_{\boldsymbol{\alpha}}-1)$. $\mathcal{P}$ computes and executes $\mathbb{V}|_L \in \mathbb{F}^{3\times|L|} \leftarrow \left[\hat{v}_{\boldsymbol{v}}|_L\, \hat{v}_{\boldsymbol{c}}|_L\, \hat{v}_{\boldsymbol{d}}|_L\right]^\top$ and $\mathsf{root}_{\mathbb{V}|_L} \leftarrow \mathsf{MT.Commit}(\mathbb{V}|_L)$. $\mathcal{P}$ then sends $\mathsf{root}_{\mathbb{V}|_L}$ to $\mathcal{V}$.
2) $\mathcal{V}$ randomly picks $\boldsymbol{r} \xleftarrow{\$} \mathbb{F}^m$ and sends it to $\mathcal{P}$. $\mathcal{V}$ computes encoding polynomials and codewords of binary vectors $\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{bi_{2^n}}$.
3) $\mathcal{P}$ and $\mathcal{V}$ compute $\hat{r} \leftarrow \text{IFFT}(\boldsymbol{r}, H)$, $\boldsymbol{s} \leftarrow \boldsymbol{r}_{[:m-n]}||\boldsymbol{0^n}$, $\hat{s} \leftarrow \text{IFFT}(\boldsymbol{s}, H)$ and $\hat{t} \leftarrow \text{IFFT}(\boldsymbol{t}, H)$. Then they invoke the batch IPA, where $\mathtt{x}_{\mathsf{IPA}} = (\mathbb{F}, H, L, \{k_j\}_{j\in[14]}, \{\hat{r}_j\}_{j\in[7]}, \{y_j\}_{j\in[7]})$, $\mathtt{w}_{\mathsf{IPA}} = (\{\hat{v}_j\}_{j\in[7]})$. Here

$$\begin{aligned}
k_{j,j\in[3]} &= 2|H|-2, \quad k_{j,j\in[4,14]} = |H|-1,\\
\{\hat{r}_j\}_{j\in[7]} &= \hat{r}, \hat{r}, \hat{r}, \hat{s}, \hat{s}, \hat{t}, \hat{t}, \quad y_{j,j\in[7]} = 0,\\
\{\hat{v}_j\}_{j\in[7]} &= \hat{w}_{\boldsymbol{v}}, \hat{w}_{\boldsymbol{c}}, \hat{w}_{\boldsymbol{d}}, \hat{v}_{\boldsymbol{c}}, \hat{v}_{\boldsymbol{d}}, \hat{u}_{\boldsymbol{c}}, \hat{u}_{\boldsymbol{d}}.
\end{aligned}$$

$\mathcal{V}$ accepts if the batch IPA passes; rejects otherwise.

Theorem 4.4. *The protocol* $\langle \mathsf{RP_A}.\mathcal{P}(\mathtt{w}), \mathsf{RP_A}.\mathcal{V}\rangle(\mathtt{x})$ *is an argument of knowledge with perfect completeness and soundness error* $2/|\mathbb{F}| + O(|L|/|\mathbb{F}|) + \mathsf{negl}(\ell, (3|H|-3)/|L|)$ *in the random oracle model.*

The security proof for the arbitrary range proof is similar to the proof for Theorem 4.2, which we omit here.

Complexity. The prover complexity is $O(n \log n)$. The verifier complexity is $O(n \log n)$ and can be reduced to linear or poly-logarithmic using delegations [17]. The communication complexity is $O(\log n)\,|\mathbb{F}| + O(\log^2 n)\,|\mathsf{H}|$.

Generalization. The arbitrary range proof could be further generalized to prove that multiple integers are in multiple respective ranges, *i.e.*, to prove that for all $i \in [t]$, $V_i \in [A_i, B_i-1]$ for non-negative integers $\{A_i\}_{i\in[t]}$ and $\{B_i\}_{i\in[t]}$. The high-level idea is to transform every range relation $V_i \in [A_i, B_i-1]$ into multiple inner product relations, then use the batch IPA to prove them.

4.5. Adding Zero-Knowledge

We have presented three range proofs in Section 4.2-4.4, but they are not zero-knowledge. Taking the batch range proof in Section 4.3 as an example, two places leak knowledge about $\{V_j\}_{j\in[t]}$: (1) When opening the Merkle tree for $\mathbb{V}|_L$, the verifier sees $\kappa = O(\ell)$ evaluations, which are defined by $\{V_j\}_{j\in[t]}$; (2) The proofs of the FRI reveal information about the target polynomial in the univariate sum-check protocol $\hat{q}$, which is related to $\{\hat{v}_j\}_{j\in[t]}$.

To achieve zero-knowledge property, we adopt the standard approaches in [17]. Specifically, to eliminate the first leakage, the prover picks a random polynomial $\hat{\delta}_j$ with degree κ and masks $\hat{v}_j$ as $\hat{v}'_j \leftarrow \hat{v}_j + \hat{Z}_H \cdot \hat{\delta}_j$ and note that $\hat{v}_j|_H = \hat{v}'_j|_H$ for every $j \in [t]$ and the completeness still follows. Now any κ entries on $\hat{v}'_j|_L$ do not leak knowledge about $\hat{v}_j|_H$ because of the masking polynomial $\hat{\delta}_j$. To eliminate the second leakage, the prover samples a random

polynomial $\hat{\gamma}$ of degree $(u_{\max}+1)(|H|-1)+u_{\max}\kappa$ and runs the FRI protocol on the random linear combination of $\{\hat{v}'_j\}_{j\in[t]}$ and $\hat{\gamma}$. Now, the target polynomial $\hat{q}$ is masked by $\hat{\gamma}$, and the FRI protocol leaks no information about $\hat{q}$.

Zero-knowledge range proof $\langle \mathsf{RP}_{\mathsf{zk}}.\mathcal{P}(\mathtt{w}), \mathsf{RP}_{\mathsf{zk}}.\mathcal{V}\rangle(\mathtt{x})$ for $\mathcal{R}_{\text{B-RP}}$ stated in Definition 4.3.

1) $\mathcal{P}$ computes the u_j-ary representation vector $\boldsymbol{v_j}$ of V_j and secret polynomials $\hat{v}_j \leftarrow \text{IFFT}(\boldsymbol{v_j}, H)$ for all $j \in [t]$. $\mathcal{P}$ sets $\hat{v}'_j(x) \leftarrow \hat{v}_j(x) + \hat{Z}_H(x)\cdot\hat{\delta}_j(x)$ for random $\hat{\delta}_j$ with degree $\kappa = O(\ell)$ and generates $\hat{w}'_j \leftarrow \hat{v}'_j(\hat{v}'_j-1)\cdots(\hat{v}'_j-u_j)$. $\mathcal{P}$ randomly picks $\hat{\gamma}$ with degree $(u_{\max}+1)(|H|-1)+u_{\max}\kappa$, and computes $\Gamma \leftarrow \sum_{a\in H}\hat{\gamma}(a)$, $\hat{\gamma}|_L \leftarrow \text{FFT}(\hat{\gamma}, L)$. $\mathcal{P}$ evaluates $\mathbb{V}'||\hat{\gamma}|_L \leftarrow \left[\hat{v}'_1|_L, \ldots, \hat{v}'_t|_L, \hat{\gamma}|_L\right]^\top$ and executes $\mathsf{root}_{\mathbb{V}'||\hat{\gamma}|_L} \leftarrow \mathsf{MT.Commit}(\mathbb{V}'||\hat{\gamma}|_L)$. Finally, $\mathcal{P}$ sends Γ and $\mathsf{root}_{\mathbb{V}'||\hat{\gamma}|_L}$ to $\mathcal{V}$.
2) $\mathcal{V}$ picks $\boldsymbol{r} \xleftarrow{\$} \mathbb{F}^m$, $\beta_1, \ldots, \beta_{2t} \xleftarrow{\$} \mathbb{F}$ and sends them to $\mathcal{P}$.
3) $\mathcal{P}$ and $\mathcal{V}$ compute $\hat{r} \leftarrow \text{IFFT}(\boldsymbol{r}, H)$. They also compute $\boldsymbol{s_j} \leftarrow \boldsymbol{r}_{[:m-n_j]}||\boldsymbol{0^n}$ and $\hat{s}_j \leftarrow \text{IFFT}(\boldsymbol{s_j}, H)$ for all $j \in [t]$.
4) $\mathcal{P}$ computes $\hat{q} \leftarrow \sum_{j=1}^{t}\beta_j\hat{w}'_j\hat{r}_j + \sum_{j=t+1}^{2t}\beta_j\hat{v}'_{j-t}\hat{r}_j + \hat{\gamma}$ and decomposes $\hat{q}$ as $x\cdot\hat{g}(x)+\zeta+\hat{Z}_H(x)\hat{h}(x)$. $\mathcal{P}$ then computes $\hat{p}(x) \leftarrow (|H|\cdot\hat{q}(x) - \mu - |H|\cdot\hat{Z}_H(x)\hat{h}(x))/x$, $\hat{h}|_L \leftarrow \text{FFT}(\hat{h}, L)$ and $\mathsf{root}_{\hat{h}|_L} \leftarrow \mathsf{MT.Commit}(\hat{h}|_L)$. $\mathcal{P}$ sends $\mathsf{root}_{\hat{h}|_L}$ to $\mathcal{V}$.
5) $\mathcal{P}$ and $\mathcal{V}$ invoke $\langle \mathsf{FRI}.\mathcal{P}(\hat{q},\hat{h},\hat{p}), \mathsf{FRI}.\mathcal{V}^{\hat{q}|_L,\hat{h}|_L,\hat{p}|_L}\rangle(\mathtt{x}_{\mathsf{FRI}})$, where $\mathtt{x}_{\mathsf{FRI}} = \big((u_{\max}+1)(|H|-1)+u_{\max}\kappa+1, u_{\max}(|H|-1)+u_{\max}\kappa, |H|-1\big)$. $\mathcal{V}$ outputs reject and aborts if the FRI protocol does not pass; otherwise, $\mathcal{V}$ sends a random location set $\mathcal{I} \subset [|L|](|\mathcal{I}|=\kappa)$ to $\mathcal{P}$.
6) $\mathcal{P}$ executes $(\{\mathbb{V}'||\hat{\gamma}|_L[i]\}_{i\in\mathcal{I}}, \pi_{\mathcal{I}}^{\mathbb{V}'||\hat{\gamma}|_L}) \leftarrow \mathsf{MT.Open}(\mathcal{I}, \mathbb{V}'||\hat{\gamma}|_L)$ and $(\{h|_L[i]\}_{i\in\mathcal{I}}, \pi_{\mathcal{I}}^{\hat{h}|_L}) \leftarrow \mathsf{MT.Open}(\mathcal{I}, \hat{h}|_L)$, then sends all opening information to $\mathcal{V}$.
7) $\mathcal{V}$ executes $\mathsf{MT.Verify}(\mathsf{root}_{\mathbb{V}'||\hat{\gamma}|_L}, \mathcal{I}, \{\mathbb{V}'||\hat{\gamma}|_L\}_{i\in\mathcal{I}}, \pi_{\mathcal{I}}^{\mathbb{V}'||\hat{\gamma}|_L})$ and $\mathsf{MT.Verify}(\mathsf{root}_{\hat{h}|_L}, \mathcal{I}, \{\hat{h}|_L[i]\}_{i\in\mathcal{I}}, \pi_{\mathcal{I}}^{\hat{h}|_L})$. $\mathcal{V}$ accepts if both output accept; rejects otherwise.

Theorem 4.5. *The protocol* $\langle \mathsf{RP}_{\mathsf{zk}}.\mathcal{P}(\mathtt{w}), \mathsf{RP}_{\mathsf{zk}}.\mathcal{V}\rangle(\mathtt{x})$ *is an honest-verifier ZKAoK with perfect completeness and soundness error* $2/|\mathbb{F}| + O(|L|/|\mathbb{F}|) + \mathsf{negl}(\ell, ((u_{\max}+1)(|H|-1)+u_{\max}\kappa)/|L|)$ *in the random oracle model.*

We have presented a high-level idea for zero knowledge before. Completeness holds due to the linearity of the RS code and the fact that $\hat{\delta}_j$ does not affect completeness since $\hat{v}'_j(a) = \hat{v}_j(a), \hat{w}'_j(a) = \hat{w}_j(a)$ for all $a \in H$ and $j \in [t]$. Soundness comes from the fact that the modified polynomial invoked in the univariate sum-check protocol is a random linear combination of secret polynomials $\{\hat{w}'_j, \hat{v}'_j\}_{j\in[t]}$ and random masking polynomial $\hat{\gamma}$. By random linear combination, the sum-check relation is satisfied with probability no more than $1/|\mathbb{F}|$ if any of the inner product relations in Equation (6) is invalid. If the sum-check relation is not satisfied, the soundness directly follows that of the univariate sum-check protocol. Appendix E gives the formal proof.

5. Evaluation

5.1. Optimizations for the Proof Size

The proof size of SHARP-PQ is mainly attributed to the FRI protocol introduced in Section 3.3. Implementing FRI directly leads to a "start-up" cost [15] of 50-100 KBs [26]. We adopt several improvement techniques [37], [38] and also present our optimizations.

Soundness choice. We rely on the conjectured soundness of FRI in [39], which is applied in the implementations of [30], [17], [37], [38]. Using this conjecture, the proof size would be reduced by about 50% compared to that using the provable soundness [39] at the same security level.

Putting evaluations into a single leaf. In each round of FRI, the verifier requires multiple evaluations on the committed codeword for each query. These evaluations constitute a coset uniquely determined by the given query. Once an evaluation in some coset is queried, all the evaluations within the entire coset are queried. Consequently, we group these evaluations into a single leaf of a Merkle tree, reducing the verification path size.

Finding the best reduction strategy. The reduction strategy describes the degree of decline amount in each round of FRI. In the original paper [26], the polynomial degree is halved every round until the polynomial becomes a constant. To minimize proof size, the degree can be reduced by a power of 2 instead of only 2 in each round. In addition, the FRI protocol may terminate before the polynomial degree reaches a constant [38]. Although these two modifications increase the number of field elements, they reduce the verification path size of Merkle trees. Our implementation utilizes an exhaustive search to determine the optimal reduction strategy for minimizing proof size.

Committing multiple RS codewords using one Merkle tree. There are multiple secret vectors in batch range proofs. A straightforward method to generate the commitment is to commit RS codewords separately for each vector using multiple Merkle trees. Observing that all the RS codewords are queried at common locations, we arrange each RS codeword in a single row to create a codeword matrix and put every column into one leaf of a Merkle tree. As a result, a single Merkle tree suffices regardless of the secret vector number, thereby reducing the proof size.

5.2. Implementations and Performance

We implement SHARP-PQ in C++[2] with $\sim$2,000 lines of code for the main protocol, including the IPA and FRI. We refer to libiop[3] for FFTs and frameworks of finite fields. We run all experiments without parallelization on an AMD Ryzen 3900X processor with 80 GB RAM. We report the average running time of 100 executions.

2. anonymous.4open.science/r/Sharp-PQ-4D04
3. github.com/scipr-lab/libiop

Choice of field. FRI protocol requires that there exist multiplicative cosets with order 2^k for large enough k. We use a prime field $\mathbb{F}_p$ where $p = 2^{64} - 2^{32} + 1$, which has a multiplicative coset with order 2^{32}. One advantage of using a prime field is that if the square of some element is 0, then the element is 0. As a result, the number of relations in range proofs is reduced, reducing the computation time. For example, it suffices to prove $\boldsymbol{v} \circ (\boldsymbol{v} - \mathbf{0}^{m-n} || \mathbf{1}^n) = \mathbf{0}^m$ instead of proving two relations in Equation (3) for $u = 2$. Another advantage of $\mathbb{F}_p$ is that the properties of $2^{64} \equiv 2^{32} - 1 \mod p$ and $2^{96} \equiv -1 \mod p$ would help accelerate multiplication and modular operations hence FFTs.

Choice of other parameters. The target security level is set as 120 bits. The soundness of FRI is based on Conjecture 8.4 [39]. We set the RS code rate to be $1/8$ for single range proofs and $1/16$ for batch proofs. Note that the best reduction strategies may vary for different schemes. To give a detailed calculation of the concrete proof size, we provide a numerical example in Appendix F. We do not use the optimization method to achieve poly-logarithmic verifier complexity for a better trade-off performance, which can incur a comparably large proof size.

Methodology. We compare SHARP-PQ with two lattice-based ZKRPs (*i.e.*, LNS20 [15] and CKLR21 [13]) and Bulletproofs [12]. The C++ implementation of LNS20[4] only gives a ZKP for 128-bit integer addition but lacks a ZKP for integers with other sizes and a ZKRP. According to [15, §B], we run two ZKPs for integer addition to construct a ZKRP for a single 128-bit range.[5] For batch proof, we run LNS20 repeatedly for batch processing. As CKLR21 is not open-source, we only compare its proof size with our scheme. Concerning Bulletproofs, we use the implementation[6] of [40] written in C, where the elliptic curve is BN128 and its security level is estimated to be 110 bits in practice [41].

Apart from ZKRPs, we also compare SHARP-PQ with two general hash-based ZKPs, *i.e.*, Ligero [19] and Aurora [16], both implemented in libiop. To construct range proofs, we reduce the range relation to an R1CS, invoking these ZKPs with the R1CS as a statement.

Comparison with ZKRPs. As shown in Table 4 and 5, SHARP-PQ improves by about 20% on LNS20[7] proof sizes for $n = 512$, and is also estimated better LNS20 for $n = 256$. Range proof for such "reasonably-sized" integers [15] may be useful in Ethereum and smart contracts where every integer is 256-bit. This gap grows larger for larger ranges as the communication complexity of SHARP-PQ is $O(\log^2 n)$ while LNS20 is $O(n)$. For batch range proof, the proof size is 10-20× smaller than LNS20 for arbitrary ranges and 10-40× smaller than CKLR21 for fixed ranges.

Compared with Bulletproofs, SHARP-PQ has at least 10× faster prover and verifier time, as there are only lightweight cryptographic operations such as FFTs without group exponentiation. However, SHARP-PQ can have a much larger proof size, which may be due to the inability of post-quantum cryptographic primitives to efficiently encode vectors. Specifically, in DLOG-based works, a vector on the field can be encoded as a group element, while it is tough to encode a vector into a codeword shorter than itself using RS codes or based on the lattice.

Compared with general ZKPs. Table 6 shows the performance advantage of SHARP-PQ over range proofs constructed from hash-based general ZKPs. Compared with Ligero, SHARP-PQ has a 10× smaller proof size, 10× faster prover time, and 10-100× quicker verifier time. The advantage is more explicit for batch ranges. Compared with Aurora, SHARP-PQ has a 10-100× faster prover time, 1-2× smaller proof size, and 2-10× quicker verifier time. This advantage may be because four secret oracles are sent in Aurora, while one oracle suffices in SHARP-PQ.

More detailed performance. Figure 2 shows the performance of SHARP-PQ for single fixed and arbitrary ranges. Both the proof generation and verification time scale linearly with the increase of range dimension. Even for ranges $[0, 2^{4096} - 1]$, our proof is practically efficient, with a prover time of 0.17 s (resp., 0.12 s), a verifier time of 0.04 s (resp., 0.04 s), and a proof size of 63.8 KB (resp., 57.3 KB) for fixed (resp., arbitrary) ranges.

TABLE 4. COMPARISON OF LNS20 [15], BULLETPROOFS [12] AND SHARP-PQ FOR PROVER ($\mathcal{P}$) AND VERIFIER ($\mathcal{V}$): λ IS SECURITY LEVEL (IN BIT), n IS RANGE DIMENSION, AND t IS INSTANCE NUMBER

Scheme			LNS20	Bulletproofs	SHARP-PQ
Range type			Arbitrary	Fixed	Arbitrary
(λ, n, t)	$(120, 32, 1)$	Proof size	11.80 KB	0.59 KB	25.90 KB
	$(120, 128, 1)$	Proof size	26.40 KB	0.72 KB	34.30 KB
		$\mathcal{P}$'s time	0.002 s	0.050 s	0.010 s
		$\mathcal{V}$'s time	0.0003 s	0.0200 s	0.0040 s
	$(120, 512, 1)$	Proof size	51.30 KB	0.84 KB	41.90 KB
	$(120, 128, 32)$	Proof size	844.80 KB	1.03 KB	73.80 KB
		$\mathcal{P}$'s time	0.09 s	1.48 s	0.12 s
		$\mathcal{V}$'s time	0.010 s	0.550 s	0.009 s
	$(120, 128, 64)$	Proof size	1.69 MB	1.09 KB	0.12 MB
		$\mathcal{P}$'s time	0.16 s	2.87 s	0.22 s
		$\mathcal{V}$'s time	0.02 s	1.04 s	0.01 s
	$(120, 128, 128)$	Proof size	3.38 MB	1.16 KB	0.21 MB
		$\mathcal{P}$'s time	0.32 s	5.72 s	0.42 s
		$\mathcal{V}$'s time	0.04 s	2.15 s	0.03 s

TABLE 5. PROOF SIZES OF CKLR21 [13] AND SHARP-PQ ($n = 64$)

Scheme	CKLR21	SHARP-PQ	Scheme	CKLR21	SHARP-PQ
$(80, 180)^\dagger$	2.09 MB	0.07 MB	$(128, 180)$	4.52 MB	0.12 MB
$(80, 500)$	2.29 MB	0.17 MB	$(128, 500)$	4.87 MB	0.27 MB
$(80, 1000)$	2.57 MB	0.32 MB	$(128, 1000)$	5.36 MB	0.52 MB

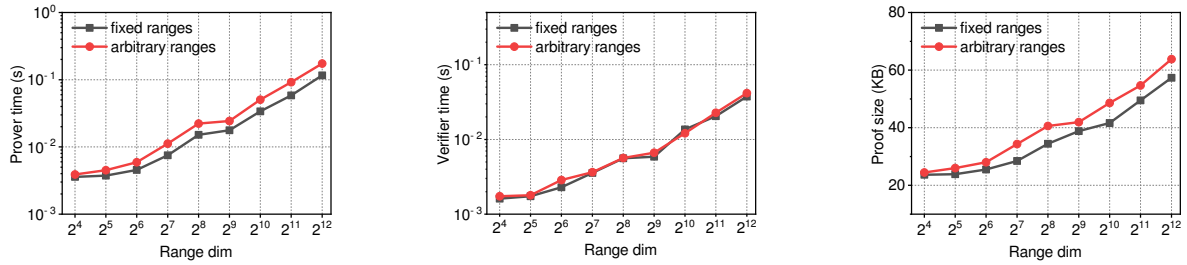
† This column denotes (λ, t) with the same meaning as Table 4.

4. github.com/gregorseiler/irelzk

5. It is non-trivial to achieve a ZKP for integers larger than 128 bits using the existing implementation. This is because the commitment number of ZKPs increases from 4 to $4 \times n/128$ to prevent the module from crossing the boundary [15] for $n > 128$, where n is the integer size. The running time of LNS20 can also be much worse with the increased number of commitments for a range dimension larger than 128.

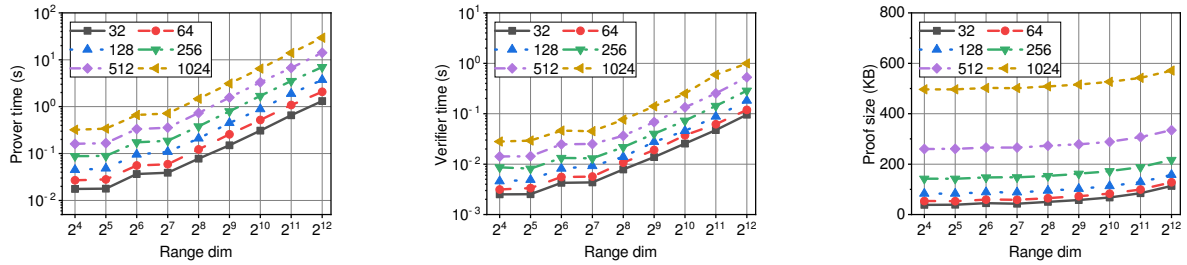
6. github.com/xevisalle/zpie

7. The ZKRP in LNS20 is indeed for $[-2^{n-1}, 2^{n-1} - 1]$, but the constraints to $\boldsymbol{v}$ are the same as those when proving $V \in [0, 2^n - 1]$.



(a) prover time (b) verifier time (c) proof size

Figure 2. Performance of SHARP-PQ for fixed ranges $[0, 2^n - 1]$ and $[A, B - 1] \subset [0, 2^n - 1]$ where $n = 2^k$ is range dimension and $k \in [4, 12]$



(a) prover time (b) verifier time (c) proof size

Figure 3. Performance of SHARP-PQ in the batch setting for ranges $[0, 2^n - 1]$ with selected instance numbers where $n = 2^k$ and $k \in \{4, \ldots, 12\}$

TABLE 6. COMPARISON OF SHARP-PQ, LIGERO, AND AURORA FOR FIXED RANGES ($n = 120$)

Scheme			Ligero [19]	Aurora [16]	SHARP-PQ
(λ, t)	$(32, 1)^\dagger$	proof size	235.2 KB	35.8 KB	23.9 KB
		prover time	0.040 s	0.030 s	0.003 s
		verifier time	0.020 s	0.003 s	0.001 s
	$(128, 1)$	proof size	376.3 KB	45.0 KB	28.5 KB
		prover time	0.070 s	0.090 s	0.010 s
		verifier time	0.050 s	0.005 s	0.003 s
	$(512, 1)$	proof size	559.7 KB	53.7 KB	38.8 KB
		prover time	0.140 s	0.290 s	0.020 s
		verifier time	0.110 s	0.008 s	0.005 s
	$(128, 32)$	proof size	1.58 MB	75.0 KB	43.1 KB
		prover time	0.640 s	2.200 s	0.040 s
		verifier time	0.450 s	0.040 s	0.004 s
	$(128, 64)$	proof size	1.8 MB	96.3 KB	58.6 KB
		prover time	0.850 s	4.750 s	0.060 s
		verifier time	0.550 s	0.080 s	0.006 s
	$(128, 128)$	proof size	2.9 MB	104.9 KB	88.0 KB
		prover time	1.65 s	9.51 s	0.11 s
		verifier time	1.12 s	0.15 s	0.01 s

† This column denotes (λ, t) with the same meaning as Table 4.

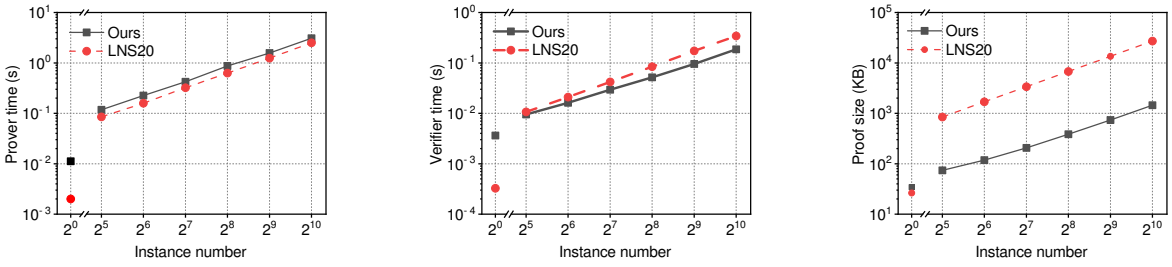
Figure 3 illustrates the efficiency of our batch range proof. Specifically, compared with the straightforward approach to build a batch range proof, *i.e.*, repeatedly running the range proof in Section 4.2, the proof generation time is $20\times$ faster, the verification time is $100\times$ faster, and the proof size is $100\times$ smaller.

Figure 4 shows the comparison of SHARP-PQ with LNS20 in the batch setting. The prover and verifier running time of SHARP-PQ is competitive with LNS20, with less than $1\times$ slower prover time and 1-2$\times$ faster verifier time. However, the proof size of SHARP-PQ is significantly better than LNS20, which is at least $10\times$ smaller. This is possible because that SHARP-PQ has an efficient batch method for multiple range relations, especially in the commitment way.

Figure 5 compares SHARP-PQ with Bulletproofs. Although with a 30-500$\times$ larger proof size, the prover and verifier time of SHARP-PQ is 30-50$\times$ and 50-200$\times$ faster, respectively. Bulletproofs only supports batch proofs where the instance numbers are powers of two [12], while SHARP-PQ allows arbitrary instance numbers. Furthermore, our scheme is provably secure only in the random oracle model and is plausibly post-quantum secure.

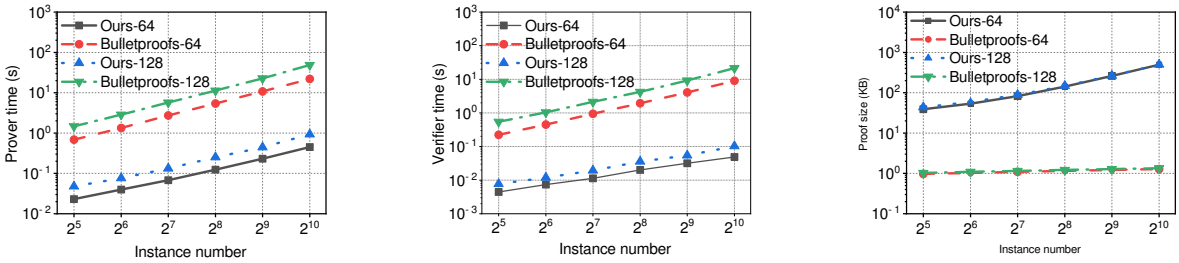
5.3. Application

We discuss ZKRPs in a typical scenario with a relation check, *i.e.*, given commitments to integers A, B, and C, the verifier that $A+B = C$, and a range check, *i.e.*, A, B, and C are all in a specified range. In confidential transactions [8], A, B can be input coin values, and C can be an output coin value, all hidden through commitments. The relation check guarantees the validity of the transaction balance. These coin values are often encoded as field elements. However, a negative integer can also be encoded as a valid field element and pass the relation check, enabling double-spending attacks creating money out of thin air [13]. To prevent these attacks, the range check is introduced to ensure each hidden output



(a) prover time (b) verifier time (c) proof size

Figure 4. Comparison of LNS20 and SHARP-PQ in the batch setting for $[A, B-1] \subset [0, 2^{128}-1]$



(a) prover time (b) verifier time (c) proof size

Figure 5. Comparison of Bulletproofs and SHARP-PQ in the batch setting: "-64"/"-128" refers to instantiation for $[0, 2^{64}-1]$/$[0, 2^{128}-1]$, respectively

coin value falls within a specific range. Note that multiple transactions can be checked at one time [8], asking for batch ZKRP.

In anonymous electronic auctions [5], every bid is hidden through commitments. An anonymous bid relation $A < C$ can be proved by $A + B = C$, and A, B, C are all in an expected range. Therefore, an auctioneer can act as a prover and show the validity of the auction by running relation checks and range checks. Note that there can be multiple bids; hence, the auctioneer may perform multiple above checks simultaneously.

With SHARP-PQ, it is possible to construct a new hash-based system for the scenario above. Firstly, relation checks can be completed through the linearity of the RS code. Specifically, given commitments to A, B, C (the Merkle tree roots to the RS codewords for $\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c} \in \mathbb{F}^m$, which are the binary representation vectors of A, B, C, respectively), one could verify $A + B = C$ by checking the inner product relation $\langle \boldsymbol{a} + \boldsymbol{b} - \boldsymbol{c}, \boldsymbol{t} \rangle = 0$ using our IPA, where $\boldsymbol{t} = (2^{m-1}, 2^{m-2}, \ldots, 2, 1)$. Secondly, the range check is a range proof and can be achieved by SHARP-PQ.

Methodology. We instantiate the relation and range checks using Bulletproofs, LNS20, and SHARP-PQ. For Bulletproofs, we omit the cost for the relation check, as the homomorphism of underlying commitments can easily accomplish it. In LNS20, the range check is completed by the proof for integer addition [15, §4]. In the case of SHARP-PQ, as the relation and range checks are both inspections of inner product relations, they could be combined and verified using a single batch IPA.

TABLE 7. EFFICIENCY OF RELATION AND RANGE CHECKS FOR $[0, 2^{128}-1]$ AND t INSTANCES USING DIFFERENT ZKRPS

t	Scheme	Commitment size	Prover time	Verifier time	Proof size
32	Bulletproofs	0.25 KB	1.48 s	0.55 s	1.03 KB
32	LNS20	0.83 KB	0.12 s	0.02 s	1.27 MB
32	SHARP-PQ	0.50 KB	0.12 s	0.01 s	0.63 MB
64	Bulletproofs	1.00 KB	2.87 s	1.04 s	1.09 KB
64	LNS20	3.30 KB	0.24 s	0.03 s	2.53 MB
64	SHARP-PQ	2.00 KB	0.22 s	0.02 s	1.25 MB
128	Bulletproofs	4.00 KB	5.72 s	2.15 s	1.16 KB
128	LNS20	13.20 KB	0.49 s	0.07 s	5.07 MB
128	SHARP-PQ	8.00 KB	0.43 s	0.04 s	2.48 MB

Performance. Table 7 displays the performance relation and range checks using different ZKRPs for $[0, 2^{128}-1]$. Compared with LNS20, SHARP-PQ is more efficient, with a 65% smaller commitment size, close faster prover time, 30-50% faster verifier time, and 50% smaller proof size. This efficiency is mainly because SHARP-PQ supports efficient batch processing, and all the transaction balance and range checks can be combined and completed simultaneously using the batch IPA. This demonstrates the benefits of employing SHARP-PQ to construct post-quantum systems, such as confidential transactions on a post-quantum blockchain.

Compared with Bulletproofs, despite the $1000\times$ larger proof size, the prover and verifier time of SHARP-PQ are both at least $10\times$ faster than Bulletproofs. This efficiency may be of interest in applications with high demands for

running time, such as anonymous electronic auctions.

6. Related Work

Zero-Knowledge Range Proofs. Two main approaches for constructing ZKRPs are square decomposition and u-ary decomposition. The first approach relies on the idea that to prove a secret integer V lies in a range $[A, B-1]$. It suffices to show that both $V-A$ and $B-1-V$ are non-negative. Further, if an integer X is non-negative, $4X+1$ is the sum of three squares [2]. A common issue of this line of works [42], [2], [43] is the requirement of RSA groups or class groups with a hard-to-factor discriminant, which leads to a large proof size and poor performance in practice.

Recently, Couteau *et al.* [13] propose highly efficient ZKRPs based on DLOG or LWE and SIS assumptions. In the DLOG setting, the proof size for a 64-bit range with 128-bit security is less than 1 KB, as competitive as Bulletproofs [12]. With the same parameters, the post-quantum batch ZKRPs based on LWE and SIS assumptions improve over state of the art in a batch setting and have a concrete proof size of 5.36 MB for 1000 instances.

The u-ary decomposition approach is used in the latest state-of-the-art protocols [12], [44], [45]. Bünz *et al.* [12] introduce Bulletproofs with logarithmic communication complexity under the DLOG assumption and high computational efficiency. The main cryptographic primitive of Bulletproofs is the DLOG-based inner product argument [25], which is later improved by a line of works [46], [45]. As for post-quantum ZKRPs using this method, Attema *et al.* [14] propose an argument for proving element-product relations between committed values and use it to construct a range proof based on the module LWE and SIS assumptions. Based on the same assumptions, Lyubashevsky *et al.* [15] present a practical latticed-based ZKP showing the satisfiability of additive relationships for committed integers and construct a post-quantum ZKRP for arbitrary ranges using this ZKP. For an arbitrary range $[A, B-1] \subset [0, 2^{512}-1]$ with 120-bit security, the proof size is 51.3 KB.

Inner Product Argument allows a prover to convince a verifier that the inner product of two encoded vectors (one secret and one public in usual) equals a public scalar. One typical IPA is based on the DLOG assumption where vectors are encoded by Pedersen hash [25]. In this line of works, IPAs [25], [12], [46], [47] are with logarithmic communication complexity and linear verifier complexity. The verifier complexity is reduced to the logarithmic level in [45], but it relies on structured commitment keys, hence trusted setups. In this paper, we follow the IPA implicit in [17], where vectors are encoded by RS code. This IPA is constructed by the univariate sum-check protocol in [16].

7. Conclusion

We propose SHARP-PQ, a transparent, plausibly post-quantum secure hash-based ZKRP with poly-logarithmic communication complexity. SHARP-PQ supports arbitrary range proofs in a batch setting efficiently and may contribute to post-quantum applications. Future works include constructing a succinct batch hash-based ZKRP with communication complexity sub-linear to the instance number. Using our proposed framework, it is also possible to build succinct ZKRPs in other categories, such as lattice-based ones.

References

[1] E. F. Brickell, D. Chaum, I. Damgård, and J. van de Graaf, "Gradual and verifiable release of a secret," in *CRYPTO '87, Santa Barbara, California, USA*. Springer, 1987, pp. 156–166. 1

[2] J. Groth, "Non-interactive zero-knowledge arguments for voting," in *ACNS 2005*, vol. 3531. New York, NY, USA: Springer, 2005, pp. 467–482. 1, 6

[3] I. Damgård, M. Jurik, and J. B. Nielsen, "A generalization of Paillier's public-key system with applications to electronic voting," *Int. J. Inf. Sec.*, vol. 9, no. 6, pp. 371–385, 2010. 1

[4] H. Lipmaa, N. Asokan, and V. Niemi, "Secure vickrey auctions without threshold trust," in *FC 2002*. Southampton, Bermuda: Springer, 2002, pp. 87–101. 1

[5] S. Micali and M. O. Rabin, "Cryptography miracles, secure auctions, matching problem verification," *Commun. ACM*, vol. 57, no. 2, pp. 85–93, 2014. 1, 5.3

[6] J. Camenisch, S. Hohenberger, and A. Lysyanskaya, "Compact E-Cash," in *EUROCRYPT 2005*. Aarhus, Denmark: Springer, 2005, pp. 302–321. 1

[7] M. H. Au, W. Susilo, Y. Mu, and S. S. M. Chow, "Constant-size dynamic k-times anonymous authentication," *IEEE Syst. J.*, vol. 7, no. 2, pp. 249–261, 2013. 1

[8] koe, K. M. Alonso, and S. Noether, "Zero to Monero: Second edition," 2020. [Online]. Available: https://www.getmonero.org/library/Zero-to-Monero-2-0-0.pdf 1, 5.3

[9] B. Bünz, S. Agrawal, M. Zamani, and D. Boneh, "Zether: Towards privacy in a smart contract world," in *FC 2020*. Kota Kinabalu, Malaysia: Springer, 2020, pp. 423–443. 1

[10] D. Boneh, B. Bünz, and B. Fisch, "Batching techniques for accumulators with applications to IOPs and stateless blockchains," in *CRYPTO 2019, Santa Barbara, CA, USA*. Springer, 2019, pp. 561–586. 1

[11] NIST, "Post-quantum cryptography," 2016. [Online]. Available: https://csrc.nist.gov/Projects/Post-Quantum-Cryptography 1

[12] B. Bünz, J. Bootle, D. Boneh, A. Poelstra, P. Wuille, and G. Maxwell, "Bulletproofs: Short proofs for confidential transactions and more," in *SP 2018*. San Francisco, California, USA: IEEE Computer Society, 2018, pp. 315–334. 1, 1.1, 2, 3, 2.1, 2.1, 5.2, 4, 5.2, 6

[13] G. Couteau, M. Klooß, H. Lin, and M. Reichle, "Efficient range proofs with transparent setup from bounded integer commitments," in *EUROCRYPT 2021*. Zagreb, Croatia: Springer, 2021, pp. 247–277. 1, 1.1, 2, 1, 3, 5.2, 5, 5.3, 6

[14] T. Attema, V. Lyubashevsky, and G. Seiler, "Practical product proofs for lattice commitments," in *CRYPTO 2020*. Santa Barbara, CA, USA: Springer, 2020, pp. 470–499. 1, 1, 6

[15] V. Lyubashevsky, N. K. Nguyen, and G. Seiler, "Practical lattice-based zero-knowledge proofs for integer relations," in *CCS 2020*. Virtual Event, USA: ACM, 2020, pp. 1051–1070. 1, 2, 3, 5.1, 5.2, 5.2, 5, 4, 5.3, 6

[16] E. Ben-Sasson, A. Chiesa, M. Riabzev, N. Spooner, M. Virza, and N. P. Ward, "Aurora: Transparent succinct arguments for R1CS," in *EUROCRYPT 2019*. Darmstadt, Germany: Springer, 2019, pp. 103–128, full version at https://ia.cr/2018/828. 1, 1.1, 1, 2.1, 2.2, 3.1, 4.1, 4.1, 5.2, 6, 6, A.1, A.2

[17] J. Zhang, T. Xie, Y. Zhang, and D. Song, "Transparent polynomial delegation and its applications to zero knowledge proof," in *SP 2020*. San Francisco, CA, USA: IEEE, 2020, pp. 859–876. 1, 2.1, 2.2, 3, 3.1, 4.2, 4.4, 4.5, 5.1, 6, C

[18] E. Ben-Sasson, A. Chiesa, and N. Spooner, "Interactive oracle proofs," in *TCC 2016-B*. Beijing, China: Springer, 2016, pp. 31–60. 1.1, 3.2, 4.1, C

[19] S. Ames, C. Hazay, Y. Ishai, and M. Venkitasubramaniam, "Ligero: Lightweight sublinear arguments without a trusted setup," in *CCS 2017A*. Dallas, TX, USA: ACM, 2017, pp. 2087–2104. 1, 2.1, 2.2, 5.2, 6

[20] A. Chiesa, D. Ojha, and N. Spooner, "Fractal: Post-quantum and transparent recursive proofs from holography," in *EUROCRYPT 2020*. Springer, 2020, pp. 769–793. 1, D, D.1, D.2, D.2

[21] R. Canetti, Y. Chen, J. Holmgren, A. Lombardi, G. N. Rothblum, and R. D. Rothblum, "Fiat-Shamir from simpler assumptions," 2018. [Online]. Available: https://eprint.iacr.org/2018/1004 1, 4.2, D

[22] A. Chiesa, P. Manohar, and N. Spooner, "Succinct arguments in the quantum random oracle model," in *TCC 2019*. Nuremberg, Germany: Springer, 2019, pp. 1–29. 1, 4.2, D, D.1

[23] C. Baum, I. Damgård, V. Lyubashevsky, S. Oechsner, and C. Peikert, "More efficient commitments from structured lattice assumptions," in *SCN 2018*. Springer, 2018, pp. 368–385. 2.1

[24] R. Yang, M. H. Au, Z. Zhang, Q. Xu, Z. Yu, and W. Whyte, "Efficient lattice-based zero-knowledge arguments with standard soundness: Construction and applications," in *CRYPTO 2019, Santa Barbara, CA, USA*. Springer, 2019, pp. 147–175. 2.1

[25] J. Bootle, A. Cerulli, P. Chaidos, J. Groth, and C. Petit, "Efficient zero-knowledge arguments for arithmetic circuits in the discrete log setting," in *EUROCRYPT 2016*. Vienna, Austria: Springer, 2016, pp. 327–357. 2.1, 6

[26] E. Ben-Sasson, I. Bentov, Y. Horesh, and M. Riabzev, "Fast Reed-Solomon interactive oracle proofs of proximity," in *ICALP 2018*, 2018, pp. 14:1–14:17. 2.1, 3.3, 5.1, B, B.1

[27] L. Eagen, "Bulletproofs++: Next generation confidential transactions via reciprocal set membership arguments," 2022. [Online]. Available: https://eprint.iacr.org/2022/510 2.1

[28] S. Das, Z. Xiang, and L. Ren, "Asynchronous data dissemination and its applications," in *CCS 2021, Virtual Event, Republic of Korea*. ACM, 2021, pp. 2705–2721. 2.2

[29] N. Alhaddad, S. Das, S. Duan, L. Ren, M. Varia, Z. Xiang, and H. Zhang, "Balanced byzantine reliable broadcast with near-optimal communication and improved computation," in *PODC 2022 Salerno, Italy*. ACM, 2022, pp. 399–417. 2.2

[30] R. Bhadauria, Z. Fang, C. Hazay, M. Venkitasubramaniam, T. Xie, and Y. Zhang, "Ligero++: A new optimized sublinear IOP," in *CCS 2020*. Virtual Event, USA: ACM, 2020, pp. 2025–2038. 2.2, 5.1

[31] S. Goldwasser, Y. T. Kalai, and G. N. Rothblum, "Delegating computation: Interactive proofs for muggles," *J. ACM*, vol. 62, no. 4, pp. 27:1–27:64, 2015. 2.2, 4.2

[32] J. Camenisch, R. Chaabouni, and abhi shelat, "Efficient protocols for set membership and range proofs," in *ASIACRYPT 2008*. Melbourne, Australia: Springer, 2008, pp. 234–252. 2.2, 4.4

[33] R. C. Merkle, "A certified digital signature," in *CRYPTO 1989*. Santa Barbara, California, USA: Springer, 1989, pp. 218–238. 3

[34] S. Goldwasser, S. Micali, and C. Rackoff, "The knowledge complexity of interactive proof-systems (extended abstract)," in *STOC 1985*. Providence, Rhode Island, USA: ACM, 1985, pp. 291–304. 3.2

[35] L. Babai, L. Fortnow, L. A. Levin, and M. Szegedy, "Checking computations in polylogarithmic time," in *STOC 1991*. New Orleans, Louisiana, USA: ACM, 1991, pp. 21–31. 3.2

[36] P. Valiant, "Incrementally verifiable computation or proofs of knowledge imply time/space efficiency," in *TCC 2008*. New York, USA: Springer, 2008, pp. 1–18. 4.1

[37] StarkWare Team, "ethSTARK documentation version 1.1," 2021. [Online]. Available: https://eprint.iacr.org/2021/582 5.1, B.1, B.1

[38] Polygon Zero Team, "Plonky2: Fast recursive arguments with PLONK and FRI," 2022. [Online]. Available: https://github.com/mir-protocol/plonky2/blob/main/plonky2/plonky2.pdf 5.1, B.1

[39] E. Ben-Sasson, D. Carmon, Y. Ishai, S. Kopparty, and S. Saraf, "Proximity gaps for Reed-Solomon codes," in *FOCS 2020*. Durham, NC, USA: IEEE, 2020, pp. 900–909. 5.1, 5.2, B, B.1, B.1, B.1

[40] X. Salleras and V. Daza, "ZPiE: Zero-knowledge proofs in embedded systems," 2021. [Online]. Available: https://eprint.iacr.org/2021/1382 5.2

[41] A. Menezes, P. Sarkar, and S. Singh, "Challenges with assessing the impact of NFS advances on the security of pairing-based cryptography," in *Mycrypt 2016*. Kuala Lumpur, Malaysia: Springer, 2016, pp. 83–108. 5.2

[42] H. Lipmaa, "On diophantine complexity and statistical zero-knowledge arguments," in *ASIACRYPT 2003*. Taipei, Taiwan: Springer, 2003, pp. 398–415. 6

[43] G. Couteau, T. Peters, and D. Pointcheval, "Removing the strong RSA assumption from arguments over the integers," in *EUROCRYPT 2017*. Paris, France: Springer, 2017, pp. 321–350. 6

[44] H. Chung, K. Han, C. Ju, M. Kim, and J. H. Seo, "Bulletproofs+: Shorter proofs for a privacy-enhanced distributed ledger," *IEEE Access*, vol. 10, pp. 42 067–42 082, 2022. 6

[45] V. Daza, C. Ràfols, and A. Zacharakis, "Updateable inner product argument with logarithmic verifier and applications," in *PKC 2020*. Edinburgh, UK: Springer, 2020, pp. 527–557. 6

[46] M. Hoffmann, M. Klooß, and A. Rupp, "Efficient zero-knowledge arguments in the discrete log setting, revisited," in *CCS 2019*. London, UK: ACM, 2019, pp. 2093–2110. 6

[47] Z. Zhang, Z. Zhou, W. Li, and H. Tao, "An optimized inner product argument with more application scenarios," in *ICICS 2021*. Chongqing, China: Springer, 2021, pp. 341–357. 6

[48] D. B. Chandler, J. Wu, and Q. Xiang, "Power sums over subspaces of finite fields," *Finite Fields Their Appl.*, vol. 18, no. 4, pp. 791–799, 2012. A.2

Appendix A. The Univariate Sum-check Protocol

We give a formal RS-encoded IOP for univariate sum-check in Protocol 1 and the security properties of this protocol are presented in Lemma A.1.

Protocol 1 An RS-encoded IOP for univariate sum-check

Inputs: $\left((\mathbb{F}, L, H, k, \mu), \hat{f}\right)$, where $\sum_{a\in H} \hat{f}(a) = \mu$.

1) $\mathcal{P}$ uniquely decomposes $\hat{f}$ as $\hat{f}(x) = x\cdot\hat{g}(x)+\zeta+\hat{Z}_H(x)\hat{h}(x)$, where $\deg(\hat{g}) < |H| - 1$, ζ is a constant, and $\deg(\hat{h}) < k + 1 - |H|$.
2) $\mathcal{P}$ sends oracle $\hat{h}|_L \in \mathrm{RS}[L, (k + 1 - |H|)/|L|]$ to $\mathcal{V}$.
3) $\mathcal{V}$ outputs accept if and only if $\hat{p}|_L \in \mathrm{RS}[L, (|H| - 1)/|L|]$, where

$$\hat{p}(x) = \frac{|H| \cdot \hat{f}(x) - \mu - |H| \cdot \hat{Z}_H(x)\hat{h}(x)}{x}. \tag{7}$$

Otherwise, $\mathcal{V}$ outputs reject.

Lemma A.1 ([16]). *Protocol 1 is an RS-encoded IOP with perfect completeness and soundness.*

Proof. **Completeness.** It follows from the following lemma.

Lemma A.2 ([48], [16]). *Let H be a multiplicative coset of $\mathbb{F}$, and let $\hat{g}$ be a univariate polynomial over $\mathbb{F}$ with a degree strictly less than $|H|$. Then $\sum_{a\in H} \hat{g}(a) = |H| \cdot \hat{g}(0)$.*

By definition of $\hat{g}, \hat{h}$ and Lemma A.2, we have

$$\begin{aligned}\mu &= \sum_{a\in H} \left(a \cdot \hat{g}(a) + \zeta + \hat{Z}_H(a)\hat{h}(a)\right) \\ &= \sum_{a\in H} (a \cdot \hat{g}(a)) + |H|\zeta + 0 = |H|\zeta.\end{aligned}$$

Therefore, we have $\zeta = \mu/|H|$. By definition of $\hat{p}$, we have

$$\hat{p}(x) = \frac{|H| \cdot \hat{f}(x) - \mu - |H| \cdot \hat{Z}_H(x)\hat{h}(x)}{x} = \hat{g}(x).$$

Hence $\hat{p}|_L \in \mathrm{RS}[L, (|H| - 1)/|L|]$.

Soundness. Suppose $\sum_{a\in H} \hat{f}(a) = \mu' \neq \mu$. We prove that for all $\hat{h}$ satisfying $\deg(\hat{h}) < k + 1 - |H|$, it holds that $\deg(\hat{p}) \geq |H| - 1$.

By contradiction, if $\hat{p}$ exists such that $\deg(\hat{p}) < |H| - 1$, then by Lemma A.2 we have $\sum_{a\in H} a\cdot\hat{p}(a) = 0\cdot\hat{p}(0)\cdot|H| = 0$. However, by Equation (7), we have

$$\sum_{a\in H} a \cdot \hat{p}(a) = |H| \cdot \sum_{a\in H} \hat{f}(a) - |H| \cdot \mu = |H| \cdot (\mu' - \mu) \neq 0.$$

This is a contradiction. □

Appendix B. FRI Oracle of Proximity

FRI [26], in full length *fast Reed-Solomon interactive oracle proof of proximity*, is a low-degree test protocol to check the validity of multiple RS codes. Formally, given target degrees $k_1, \ldots, k_t$ and codewords $\hat{v}_1|_L, \ldots, \hat{v}_t|_L$, a prover convinces a verifier with oracle access to these codewords that for all $i \in [t]$, $\hat{v}_i|_L$ is $\delta|L|$-close to $\mathrm{RS}[L, (k_i+1)/|L|]$. Here $\delta \in (0, 1)$ represents the proximity parameter. We first recall the FRI protocol to check whether $\hat{f}|_L \in \mathrm{RS}[L, (k + 1)/|L|]$ for a polynomial with degree bounded by $k + 1$.

In each round of FRI, the degree of $\hat{f}_i$ decreases by a constant 2^{η_i}. The vector $\boldsymbol{\eta} = \{\eta_1, \ldots, \eta_\mathsf{r}\}$ is called *reduction strategy*. Specifically, $\boldsymbol{\eta} = \mathbf{1}^\mathsf{r}$ in Protocol 2.

Batch-FRI. For multiple polynomials $\hat{f}_1, \ldots, \hat{f}_t$ with multiple degrees $k_1, \ldots, k_t$, one can use batch-FRI [39] to test $\hat{f}_1|_L, \ldots, \hat{f}_t|_L$ are all valid RS codes using a random linear combination. Concretely, $\mathcal{V}$ samples a random challenge vector $\boldsymbol{\lambda} \xleftarrow{\$} \mathbb{F}^t$. $\mathcal{P}$ computes $\hat{f}'(X) = \sum_{i=1}^t \lambda_i \cdot \hat{f}_i(X) \cdot X^{k_{\max}-k_i}$, where $k_{\max} = \max\{k_1, \ldots, k_t\}$. $\mathcal{P}$ and $\mathcal{V}$ continue with FRI to check whether $\hat{f}'|_L \in \mathrm{RS}[L, (k_{\max} + 1)/|L|]$. Note that the oracle access to $\hat{f}'|_L$ can be constructed by the oracle access to $\hat{f}_1|_L, \ldots, \hat{f}_t|_L$.

Complexity of batch-FRI. Let the instance number be t; the reduction strategy be $\boldsymbol{\eta}$, the round number be r, the query repetition number be ℓ, and the codeword domain be L. The proof size of batch-FRI is $t\ell \cdot \sum_{i=1}^{\mathsf{r}} 2^{\eta_i} |\mathbb{F}| + \frac{k_{\max}}{\sum_{i=1}^{\mathsf{r}} \eta_i} |\mathbb{F}| + t \cdot \mathsf{Tree}(|L|, \ell \cdot 2^{\eta_1}) |\mathsf{H}| + \sum_{i=2}^{\mathsf{r}} \mathsf{Tree}(|L_i|, \ell \cdot 2^{\eta_i}) |\mathsf{H}|$, where $\mathsf{Tree}(m, n)$ means the length of verification path when opening n entries in a m-length vector committed by the Merkle tree.

Protocol 2 FRI protocol to test if $\hat{f}|_L \in \mathrm{RS}[L,(k+1)/|L|]$

Inputs: $\big((\mathbb{F},L,d),(\hat{f})\big)$. L is a multiplicative coset. Without generality, we assume $k+1=2^{\mathsf{r}}$ for some positive integer r.

1) P sets $L_0 = L, \hat{f}_0 = \hat{f}$, computes $\hat{f}_0|_{L_0}$, executes $\mathsf{rt}_0 \leftarrow \mathsf{MT.Commit}(\hat{f}_0|_{L_0})$ and sends rt_0 to $\mathcal{V}$.
2) Set $i=1$. While $i \le \mathsf{r}+1$:
   a) $\mathcal{P}$ decomposes uniquely $\hat{f}_{i-1}$ as $\hat{f}_{i-1}(X) = \hat{g}_i(X^2) + X \cdot \hat{h}_i(X^2)$.
   b) $\mathcal{V}$ sends a uniformly random $\alpha_i \xleftarrow{\$} \mathbb{F}$ to $\mathcal{P}$.
   c) $\mathcal{P}$ computes $\hat{f}_i(X) \leftarrow \hat{g}_i(X) + \alpha_i \cdot \hat{h}_i(X)$. If $i \neq \mathsf{r}+1$, $\mathcal{P}$ computes $\hat{f}_i|_{L_i}$ where $L_i = \{x^2 | x \in L_{i-1}\}$, executes $\mathsf{rt}_i \leftarrow \mathsf{MT.Commit}(\hat{f}_i|_{L_i})$ and sends rt_i to $\mathcal{V}$. If $i = \mathsf{r}+1$, $\mathcal{P}$ directly sends $\hat{f}_i$ to $\mathcal{V}$.
   d) Set $i \leftarrow i+1$.
3) Repeat the following ℓ times:
   a) $\mathcal{V}$ samples random $\beta \xleftarrow{\$} L_0$.
   b) Set $i=1$. While $i \le \mathsf{r}+1$:
      i) $\mathcal{P}$ opens $\hat{f}_{i-1}(\pm\beta^{2^{i-1}})$ and $\hat{f}_i(\beta^{2^i})$ to $\mathcal{V}$ by $\mathsf{MT.Open}$. $\mathcal{V}$ checks the correctness of related Merkle trees by $\mathsf{MT.Verify}$. If the verification algorithm does not pass, $\mathcal{V}$ outputs reject and aborts.
      ii) $\mathcal{V}$ checks whether $(\pm\beta^{2^{i-1}}, \pm\hat{f}_{i-1}(\beta^{2^{i-1}}))$, $(\alpha_i, \hat{f}_i(\beta^{2^i}))$ are on a common constant polynomial. If not, $\mathcal{V}$ outputs reject and aborts.
      iii) Set $i \leftarrow i+1$.
4) If all the above verification passes, $\mathcal{V}$ outputs accept.

We next give an example where the reduction strategy is $\boldsymbol{\eta} = \mathbf{1}^{\mathsf{r}}$. Suppose the code rate is ρ then $\mathsf{r} = \log \rho|L|$. The codeword length in the i-th round is $|L_i| = |L|/2^{i-1}$. Since $\mathsf{Tree}(|L_i|, \ell \cdot 2^{\eta_i}) \le \ell \cdot 2^{\eta_i} \cdot \log|L_i|$, the concrete proof size is bounded by $2t\ell \cdot \log\rho|L|\,|\mathbb{F}| + \frac{k_{\max}}{2^{\mathsf{r}}}|\mathbb{F}| + 2t\log|L||\mathsf{H}| + 2\ell\sum_{i=2}^{\mathsf{r}}\log|L_i||\mathsf{H}|$. Therefore, the communication complexity is $O(t\log|L|)\,|\mathbb{F}| + O(t\log|L|)\,|\mathsf{H}| + O(\log^2|L|)\,|\mathsf{H}|$. The main costs for the verifier are verifying the Merkle trees to construct the virtual oracle for $\hat{f}$ and invoking the remaining rounds of FRI. The verifier complexity is $O(t\log|L|)\,\mathbb{F}_o + O(t\log|L|)\,\mathsf{H} + O(\log^2|L|)\,\mathsf{H}$.

B.1. Soundness of batch-FRI

The soundness of batch-FRI is closely related to the proof size of SHARP-PQ. We give two versions of it [39]: a provable one and a conjectured one.

The provable soundness. For the proximity parameter close to the Johnson bound, the soundness error of the batch-FRI is as follows according to [39, Theorem 8.3].

$$\epsilon = \frac{(j+\frac{1}{2})^7}{2\cdot\rho^{\frac{3}{2}}}\cdot\frac{|L|^2}{|\mathbb{F}|} + \frac{(2j+1)\cdot(|L|+1)}{\sqrt{\rho}}\cdot\frac{\sum_{i=1}^{\mathsf{r}}2^{\eta_i}}{|\mathbb{F}|} + (1-\delta)^{\ell},$$

where the soundness error of the interactive phase is

$$\epsilon_I = \frac{(j+\frac{1}{2})^7}{2\cdot\rho^{\frac{3}{2}}}\cdot\frac{|L|^2}{|\mathbb{F}|} + \frac{(2j+1)\cdot(|L|+1)\cdot\sum_{i=1}^{\mathsf{r}}2^{\eta_i}}{|\mathbb{F}|}, \tag{8}$$

and the soundness error of the query phase is $\epsilon_q = (1-\delta)^{\ell}$, where $\delta = 1-\sqrt{\rho}\cdot(1+\frac{1}{2j}), j \ge 3$. To achieve the target security level, the interactive repetition can be repeated e times, and $|\mathbb{F}|$ in Equation (8) will be substituted with $|\mathbb{F}|^e$.

The conjectured soundness. In their line of works on FRI [26], [39], the authors make several conjectures on the soundness of FRI for proximity parameters above the Johnson bound. The most recent conjecture is Conjecture 8.4 in [39]. By setting $c_1 = c_2 = 1$ in this conjecture, we obtain the following conclusion. This soundness is used in both estark [37, §5.10.1] and plonky2 [38].

Lemma B.1 ([37], [39]). *The conjectured security bit λ for a batch-FRI is the minimum of three values: (1) $-\ell\log_2\rho$, where ρ is the code rate and ℓ is the query repetition number; (2) $|\mathsf{H}|/2$, where $|\mathsf{H}|$ is the output size of the hash functions in the Merkle tree; (3) $k_{\max}/|\mathbb{F}|^e$, where $k_{\max}$ is the maximum degree of the polynomial to be tested and e means the interactive repetition number.*

Appendix C. Supplementary Proof of Theorem 4.1

Proof. **Argument of knowledge.** We present a formal proof similar to [17], [18]. For any PPT adversary $\mathcal{P}^*$, there exists a PPT extractor $\mathcal{E}$ such that given the random access tape of $\mathcal{P}^*$, for every statement $\mathtt{x} = \big(\mathbb{F}, H, L, \{k_j\}_{j\in[2t]}, \{\hat{r}_j\}_{j\in[t]}, \{y_j\}_{j\in[t]}\big)$ generated by $\mathcal{P}^*$, the following probability is $\mathsf{negl}(\lambda)$:

$$\Pr\left[\begin{array}{l}\mathsf{root}^* \leftarrow \mathcal{P}^*(1^\lambda,\mathtt{x}), \langle\mathcal{P}^*(),\mathcal{V}\rangle(\mathsf{pp},\mathtt{x}) = 1,\\ \{\hat{v}_j\}_{j\in[t]} \leftarrow \mathcal{E}(1^\lambda,\mathtt{x}) :\\ \mathsf{MT.Commit}(\mathbb{V}|_L) \neq \mathsf{root}^* \vee (\mathtt{x}, \{\hat{v}_j\}_{j\in[t]}) \notin \mathcal{R}_{\text{B-IPA}}\end{array}\right].$$

Suppose that the Merkle tree is based on a random oracle $\mathcal{H}: \{0,1\}^{2\lambda} \to \{0,1\}^{\lambda}$. We then construct a PPT extractor $\mathcal{E}$ with the same random type of $\mathcal{P}^*$ working as follows:

1) $\mathcal{E}$ simulates the way the same as $\mathcal{P}^*$ queries $\mathcal{H}$, and let $(q_1, q_2, \ldots, q_{\max})$ be the queries made by $\mathcal{P}^*$ to $\mathcal{H}$ in the order they are made where duplicates are omitted. The order means that queries should not be generated from parent nodes to child ones. Define $q_i \in \mathcal{H}(q_j)$ if the first λ bits or the last λ bits of q_i is $\mathcal{H}(q_j)$. That is, q_i could be the parent node of q_j. If there exists some $i \neq j$, $\mathcal{H}(q_i) = \mathcal{H}(q_j)$ or some $i \le j$, $q_i \in \mathcal{H}(q_j)$, $\mathcal{E}$ outputs random polynomials $\hat{v}_1, \ldots, \hat{v}_t$ and aborts.
2) $\mathcal{E}$ constructs a directed graph G according to the query set $Q = \{q_1, \ldots, q_{\max}\}$. There is an edge from q_i to q_j in G if and only if $q_i \in \mathcal{H}(q_j)$. The outdegree of each node is at most 2. When $\mathcal{P}^*$ generates $\mathsf{root}_{\mathbb{V}|_L}$ in the batch IPA $\langle\mathsf{IPA}_{\mathsf{B}}.\mathcal{P}(\mathtt{w}), \mathsf{IPA}_{\mathsf{B}}.\mathcal{V}\rangle(\mathtt{x})$, if $\mathsf{root}_{\mathbb{V}|_L}$ does not equal to $\mathcal{H}(q)$ for some $q \in Q$ with depth $\lceil\log_2 t\rceil$ of the binary tree, $\mathcal{E}$ outputs random polynomials as $\hat{v}_1, \ldots, \hat{v}_t$ and aborts, otherwise we suppose that $\mathcal{H}(q_r) = \mathsf{root}_{\mathbb{V}|_L}$ holds for some r. If any of the verification paths is invalid, $\mathcal{E}$ outputs random polynomials $\hat{v}_1, \ldots, \hat{v}_t$ and aborts.
3) $\mathcal{E}$ reads all leaves from the root q_r. If there exists any missing leaf, $\mathcal{E}$ outputs random polynomials as $\hat{v}_1, \ldots, \hat{v}_t$

and aborts; otherwise, it concatenates these leaf strings as $(\hat{v}_1|_L, \ldots, \hat{v}_t|_L)$, and decodes the string using an efficient RS decoding algorithm. Therefore, $\mathcal{E}$ could efficiently output $\hat{v}_1, \ldots, \hat{v}_t$.

Let E_1 be the event $\langle \mathcal{P}^*, \mathcal{V} \rangle(\mathsf{pp}, \mathtt{x}) = 1$, and E_2 be the event $(\mathtt{x}, \{\hat{v}_j\}_{j \in [t]}) \notin \mathcal{R}_{\text{B-IPA}}$. We show $\Pr[E_1 \wedge E_2] \leq \mathsf{negl}(\lambda)$.

Suppose that $\mathcal{E}$ aborts before constructing the graph G. If for some $i \neq j$, $\mathcal{H}(q_i) = \mathcal{H}(q_j)$, then $\mathcal{E}$ finds a collision. This probability is $\mathsf{negl}(\lambda)$ due to the collision-resistant property of $\mathcal{H}$. If for some $i \leq j$, $q_i \in \mathcal{H}(q_j)$, then $\mathcal{E}$ could generate a Merkle tree violating the logical order. This probability is $\mathsf{negl}(\lambda)$ since $\mathcal{H}$ is non-invertible.

Otherwise, we suppose that $\mathcal{E}$ has successfully constructed a graph. If some node on a verification path does not lie in the graph G, $\mathcal{P}^*$ has to guess the value to construct a valid verification path, and this probability is $\mathsf{negl}(\lambda)$ since $\mathcal{H}$ is non-invertible. Additionally, if any leaf is missing, then $\mathcal{V}$ will be convinced with probability $\mathsf{negl}(\lambda)$ once it queries this leaf. The probability that this leaf is not be queried by $\mathcal{V}$ is at most $(1 - 1/|L|)^\ell = \mathsf{negl}(\lambda)$ as $\ell = O(\lambda)$. If $\mathcal{E}$ does not abort, it could always extract some $\hat{v}_1, \ldots, \hat{v}_t$ efficiently. In this case, $\mathcal{V}$ accepts the statement with probability $\mathsf{negl}(\lambda)$ with an appropriate choice of parameters according to the soundness.

Therefore, we have $\Pr[E_1 \wedge E_2]$

$$
\begin{aligned}
&= \Pr[E_1 \wedge E_2 \,|\, \mathcal{E} \text{ aborts}] + \Pr[E_1 \wedge E_2 \,|\, \mathcal{E} \text{ does not abort}] \\
&\leq \Pr[\mathcal{E} \text{ aborts}] + \Pr[E_1 \wedge E_2 \,|\, \mathcal{E} \text{ does not abort}] \\
&\leq \mathsf{negl}(\lambda) + \mathsf{negl}(\lambda) = \mathsf{negl}(\lambda). \qquad \square
\end{aligned}
$$

Appendix D. Proof for the Round-by-round Soundness of Theorem 4.2

In this section, we recall some backgrounds of round-by-round (RBR) soundness and prove that our range proof $\langle \mathsf{RP}.\mathcal{P}(\mathtt{w}), \mathsf{RP}.\mathcal{V} \rangle(\mathtt{x})$ has such a property.

Chiesa, Manohar, and Spooner [22] prove that non-interactive arguments obtained via the BCS construction applied to interactive IOPs with round-by-round (RBR) soundness [21] are unconditionally secure in the quantum random oracle model. Further, Fractal [20] proves that an IOP composed of an RS-encoded IOP and FRI has RBR soundness if the RS-encoded IOP has this property. Note that $\langle \mathsf{RP}.\mathcal{P}(\mathtt{w}), \mathsf{RP}.\mathcal{V} \rangle(\mathtt{x})$ is exactly such an IOP. That is to say, if the underlying RS-encoded IOP of the range proof for $[0, u^n - 1]$ has RBR soundness, then the whole scheme is post-quantum secure. Next, we recall the definition of RBR soundness and then give such proof.

D.1. Round-by-round Soundness

Definition D.1 (Transcript [20]). *A transcript of an* k*-round IOP* $(\mathcal{G}, \mathcal{P}, \mathcal{V})$ *is a tuple of the form* $(\Pi_1, m_1, \ldots, \Pi_i, m_i)$ *for* $\in [\mathsf{k}]$*, where each* Π_j *is a prover (oracle) message and each* m_j *is a verifier message. We denote the empty transcript by* $\emptyset$*. A transcript is full if* $i = \mathsf{k}$*.*

Definition D.2 (RBR soundness [20]). *An IOP* $(\mathcal{G}, \mathcal{P}, \mathcal{V})$ *for a relation* $\mathcal{R}$ *has RBR soundness error* ϵ_{rbr} *if there exists a function* State *from the set of transcripts to* $\{\mathsf{accept}, \mathsf{reject}\}$ *such that for every transcript* tr*:*

- *if* $(\mathtt{x}, \mathtt{w}) \notin \mathcal{R}$*, and* $\mathsf{tr} = \emptyset$*, then* $\mathsf{State}(\mathtt{x}, \mathsf{tr}) = \mathsf{reject}$*.*
- *if* $\mathsf{State}(\mathtt{x}, \mathsf{tr}) = \mathsf{reject}$*, then* $\mathsf{rbr}(\mathsf{tr}) \leq \epsilon_{\mathsf{rbr}}$*, where*

$$\mathsf{rbr}(\mathsf{tr}) = \max_{\Pi} \Pr_{m}[\mathsf{State}(\mathtt{x}, \mathsf{tr} || \Pi || m) = \mathsf{accept}].$$

- *if* $\mathsf{State}(\mathtt{x}, \mathsf{tr}) = \mathsf{reject}$ *and* tr *is a full transcript,* $\mathcal{V}$ *rejects.*

The notion of round-by-round soundness of RS-encoded IOP is as above, except that the maximum in the definition of rbr is taken over $\Pi \in \mathrm{RS}[L, \rho_i]$ for code rate ρ_i and a transcript of $i-1$ rounds, and the third condition above only needs to hold for full transcripts tr where $\Pi_i \in \mathrm{RS}[L, \rho_i]$ for all i [22].

D.2. RBR Soundness of the Range Proof

The range proof $\langle \mathsf{RP}.\mathcal{P}(\mathtt{w}), \mathsf{RP}.\mathcal{V} \rangle(\mathtt{x})$ is a combination of an RS-encoded IOP and an FRI, where the RS-encoded IOP is as follows.

1) $\mathcal{P}$ sends an RS codeword oracle $\hat{v}|_L$.
2) $\mathcal{V}$ sends a random vector $\boldsymbol{r}$ and random elements β_1, β_2.
3) $\mathcal{P}$ sends RS codeword oracles $\hat{q}|_L, \hat{h}|_L$, where $\hat{q} = \beta_1 \cdot \hat{v}\hat{r} + \beta_2 \cdot \hat{v}(\hat{v} - 1) \cdots (\hat{v} - u - 1)\hat{s}$. $\hat{h}$ is the quotient polynomial of $\hat{q}$ divided by $\hat{Z}_H$.
4) $\mathcal{V}$ checks if $\hat{q}|_L \in \mathrm{RS}[L, (u+1)(|H|-1)/|L|]$, $\hat{h}|_L \in \mathrm{RS}[L, u(|H|-1)/|L|]$ and $\hat{p}|_L \in \mathrm{RS}[L, (|H|-1)/|L|]$, where $\hat{p}$ is defined in Section 3.3.

We next prove the RBR of the above RS-encoded IOP adopted from [20]. The State function takes the inputs

$$(\mathbb{F}, H, L, m, n, [0, u^n - 1]), (\hat{v}|_L, (\boldsymbol{r}, \beta_1, \beta_2), (\hat{q}|_L, \hat{h}|_L, \hat{p}|_L))$$

and runs as follows:

1) If at least one relation in Equation (5) does not satisfy but both relations in Equation (6) are valid, output accept.
2) If $\sum_{a \in H} \hat{v}(a)(\hat{v}(a) - 1) \cdots (\hat{v}(a) - u - 1)\hat{r}(a) \neq 0$ or $\sum_{a \in H} \hat{v}(a) \cdot \hat{s}(a) \neq 0$ but $\sum_{a \in H} \hat{q}(a) = 0$, output accept.
3) Otherwise, output reject. (Clearly, $\mathsf{State}(\emptyset) = \mathsf{reject}$.)

Suppose that $V \notin [0, u^n - 1]$, then at least one Hadamard relation in Equation (5) is invalid, and so the probability of moving to accept of the first case is at most $1/|\mathbb{F}|$. Similarly, the probability of moving to accept of the second case is also at most $1/|\mathbb{F}|$. Further, suppose it returns reject, then the univariate sum-check relation is not satisfied, and a verifier will reject with probability 1 by the perfect soundness of the univariate sum-check protocol in Appendix A, which bounds the RBR soundness of the RS-encoded IOP by $2/|\mathbb{F}|$.

Appendix E. Proof of Theorem 4.5

Proof. **Completeness.** It follows from the completeness of the batch IPA $\langle \mathsf{IPA}_{\mathsf{B}}.\mathcal{P}(\mathtt{w}), \mathsf{IPA}_{\mathsf{B}}.\mathcal{V} \rangle(\mathtt{x})$. Note that $\beta_j \hat{w}'_j \cdot \hat{r}_j =$

$\beta_j \hat{w}_j \cdot \hat{r}_j$ for $j \in [t]$ and $\beta_j \hat{v}'_{j-t} \cdot \hat{r}_j = \beta_j \hat{v}_{j-t} \cdot \hat{r}_j$ for $j \in [t+1, 2t]$ hold for every $a \in H$, thus $\hat{\delta}_j$ plays no part on completeness.

Soundness. The soundness error comes from two cases.

- **Case 1**. Suppose that all inner product relations are satisfied. This soundness error is bounded by $1/|\mathbb{F}|$ due to the random choice of $\boldsymbol{r}$.
- **Case 2**. The inner product relations in Equation (6) are not all satisfied. The soundness error can be divided into two subcases.
  - The univariate sum-check relation is satisfied. For any random choices $\beta_1, \ldots, \beta_{2j} \in \mathbb{F}$ and $\hat{\gamma}^*$ carefully selected by $\mathcal{P}^*$ satisfying $\sum_{a \in H} \hat{\gamma}^*(a) = \Gamma^*$, $\sum_{a \in H} (\sum_{j=1}^{t} \beta_j \hat{w}'_j \hat{r}_j + \sum_{j=t+1}^{2t} \beta_j \hat{v}'_{j-t} \hat{r}_j) + \Gamma^* = \sum_{j=1}^{2t} y_j + \Gamma$ if and only if $\beta_1, \beta_2, \ldots, \beta_{2t}$ satisfies a specific relation, which happens with a probability at most $1/|\mathbb{F}|$. That is to say, the probability that the univariate sum-check protocol is satisfied is bounded by $1/|\mathbb{F}|$.
  - The univariate sum-check relation is not satisfied and the FRI is invalid. The soundness error is bounded by $\epsilon_{\text{FRI}} = O(|L|/|\mathbb{F}|) + \mathsf{negl}(\ell, ((u_{\max}+1)(|H|-1) + u_{\max}\kappa)/|L|)$. This comes from the fact that the secret polynomial with the largest degree is $\hat{w}_i = \hat{v}_i(\hat{v}_i - 1) \cdots (\hat{v}_i - u_{\max})$ for some $i \in [t]$ and $\deg(\hat{w}_i) = (u_{\max}+1)(|H|-1) + u_{\max}\kappa$.

Combining both cases using a union-bound argument, we can conclude the total soundness error.

Argument of knowledge. The argument of knowledge property follows from that of the range proof. Note that the masking polynomial $\hat{\gamma}$ is also encoded and committed by the Merkle tree and can be extracted. The argument of knowledge hence follows.

Honest-verifier zero-knowledge. We describe a simulator $\mathcal{S}$, which is given $\big(\mathbb{F}, L, H, m, t, \{n_1, \ldots, n_t\}, \{[0, u_j^{n_j} - 1]\}\big)$ and the queried location set $\mathcal{I}$ as input, computationally simulates the view of a PPT verifier $\mathcal{V}$ in the real protocol.

1) $\mathcal{S}$ picks random polynomials $\hat{v}'_{j,\mathsf{sim}}$ with degree less than $u_j|H| + u_j\kappa - (u_j - 1)$. For each $j \in [t]$, $\mathcal{S}$ evaluates $v'_{j,\mathsf{sim}}|_L$. It then constructs $\mathbb{V}'_{\mathsf{sim}}$. $\mathcal{S}$ generates $\hat{w}'_{1,\mathsf{sim}}$, $\hat{w}'_{2,\mathsf{sim}}, \ldots, \hat{w}'_{t,\mathsf{sim}}$ as the honest prover does.
2) $\mathcal{S}$ randomly samples a polynomial $\hat{\gamma}_{\mathsf{sim}}$ of degree $(u_{\max}+1)(|H|-1) + u_{\max}\kappa$, sends $\Gamma_{\mathsf{sim}} = \sum_{a \in H} \hat{\gamma}_{\mathsf{sim}}(a)$ to $\mathcal{V}^*$, and executes $\mathsf{root}_{\mathbb{V}'_{\mathsf{sim}}||\hat{\gamma}_{\mathsf{sim}}|_L} \leftarrow \mathsf{MT.Commit}(\mathbb{V}'_{\mathsf{sim}}||\hat{\gamma}_{\mathsf{sim}}|_L)$.
3) Receive $\boldsymbol{r}, \beta_1, \beta_2, \ldots, \beta_{2t}$ from $\mathcal{V}$.
4) Generate $\{\hat{r}_j\}_{j \in [2t]}$ as the honest prover does.
5) $\mathcal{S}$ randomly picks a polynomial $\hat{p}_{\mathsf{sim}}$ of degree less than $|H|-1$. Given the challenge location set $\mathcal{I}$ generated by $\mathcal{V}$, for each $a_i \in \mathcal{I}$, $\mathcal{S}$ computes h_i such that $x \cdot \hat{p}_{\mathsf{sim}}(a_i) = \sum_{j=1}^{t} \beta_j \hat{w}'_{j,\mathsf{sim}}(a_i) \cdot \hat{r}_j(a_i) + \sum_{j=t+1}^{2t} \beta_j \hat{v}'_{j-t,\mathsf{sim}}(a_i) \cdot \hat{r}_j(a_i) + \hat{\gamma}_{\mathsf{sim}}(a_i) - \Gamma_{\mathsf{sim}} - \hat{Z}_H(a_i) h_i$. $\mathcal{S}$ picks a random polynomial $\hat{h}_{\mathsf{sim}}$ with degree less than $u_{\max}(|H|-1) + u_{\max}\kappa$ such that for each $a_i \in \mathcal{I}$, $\hat{h}_{\mathsf{sim}}(a_i) = h_i$. $\mathcal{S}$ sends $\mathsf{root}_{\hat{h}_{\mathsf{sim}}|_L} \leftarrow \mathsf{MT.Commit}(\hat{h}_{\mathsf{sim}}|_L)$.
6) $\mathcal{S}$ simulates the FRI as an honest prover with $\mathcal{V}$.
7) $\mathcal{S}$ opens $\big(\{\mathbb{V}'_{\mathsf{sim}}||\hat{\gamma}_{\mathsf{sim}}|_L[i]\}_{i \in \mathcal{I}}, \pi_{\mathcal{I}}^{\mathbb{V}'_{\mathsf{sim}}||\hat{\gamma}_{\mathsf{sim}}|_L}\big) \leftarrow \mathsf{MT.Open}(\mathcal{I}, \mathbb{V}'_{\mathsf{sim}}||\hat{\gamma}_{\mathsf{sim}}|_L)$ and $\big(\{\hat{h}_{\mathsf{sim}}|_L[i]\}_{i \in \mathcal{I}}, \pi_{\mathcal{I}}^{\hat{h}_{\mathsf{sim}}|_L}\big) \leftarrow \mathsf{MT.Open}(\mathcal{I}, \hat{h}_{\mathsf{sim}}|_L)$.

To prove zero-knowledge property, for any $j \in [t]$, $\hat{v}'_{j,\mathsf{sim}}$ and $\hat{v}'_j$ are uniformly distributed, and $\hat{w}'_{j,\mathsf{sim}}, \hat{w}'_j$ are also uniformly distributed. Steps 2, 3, 4, and 6 are the same as the real world in the range proof.

In steps 5 and 7, $\hat{\gamma}_{\mathsf{sim}}$ and $\hat{\gamma}$ are both randomly picked; hence the roots and opened points are indistinguishable. As the degrees of $\hat{v}'_{j,\mathsf{sim}}, \hat{w}'_{j,\mathsf{sim}}$ and $\hat{h}_{\mathsf{sim}}$ are increased by at least κ for $j \in [t]$, any opened κ evaluations of $\hat{v}'_{j,\mathsf{sim}}, \hat{w}'_{j,\mathsf{sim}}$ and $\hat{h}_{\mathsf{sim}}$ are independent and randomly distributed, which is indistinguishable from the real world. Thus, the view of step 7 is also simulated by $\mathcal{S}$.

□

Appendix F.
A Numerical Example

In the interactive version, we give a numerical example of the concrete proof size of SHARP-PQ. Suppose that the field size $|\mathbb{F}|$ is less than 2^{64}, the range is $[0, 2^{2^9} - 1]$, the code rate ρ is 2^{-3}, and the security level λ is 120-bit. For the ZKRP described in Section 4.5, we perform exhaustive research to find an appropriate reduction strategy and set $\boldsymbol{\eta} = \{2, 3\}$. The soundness error of the transformation from the Hadamard relations to inner product relations is $1/|\mathbb{F}|$. Hence, it needs 2 random vectors from the verifier to achieve the target security level. Based on the conjecture in Appendix B, the query repetition number ℓ needs to satisfy $-\log_2 \rho \cdot \ell > \lambda$, and we set $\ell = 41$. It suffices to repeat the interactive phase of FRI 3 times. This achieves an interactive soundness error of $2^{11}/2^{63 \times 3} = 2^{-178}$. We use a hash function with an output size of 256 bits. The degree of the secret polynomial $\hat{v}$ is bounded by $k = 2^9$, and the max degree of the polynomials invoked in the univariate sum-check protocol is $3k - 2 + 2 \cdot 2^{\eta_1} \cdot \ell < 2^{11}$. Therefore, we can set $m = |H| = k_{\max} = 2^{11}$ and $|L| = 2^{14}$. The total proof size is

$$
\begin{aligned}
&1|\mathsf{H}| + 1|\mathsf{H}| + 2\mathsf{Tree}(|L|/2^{\eta_1}, \ell)\,|\mathsf{H}| + 3\mathsf{Tree}(|L|/2^{\eta_1+\eta_2}, \ell)\,|\mathsf{H}| \\
&+ 3\ell \cdot 2^{\eta_1}\,|\mathbb{F}| + 3 \cdot \ell \cdot 2^{\eta_2}\,|\mathbb{F}| + 3 \cdot 2^{k_{\max}-\eta_1-\eta_2}\,|\mathbb{F}| \\
&\approx 833\,|\mathsf{H}| + 1635\,|\mathbb{F}| = 38.8\,\text{KB},
\end{aligned}
$$

which is an approximation since the size of the verification path is not fixed due to the random queries to the same Merkle tree. Note that with the provable bound, the proof size is 64.3 KB, 40% larger than the 38.8 KB.