KiloNova: Non-Uniform Zero-Knowledge PCD from Generic Folding Schemes

Tianyu Zheng[1], Shang Gao[1], Yu Guo[2], and Bin Xiao[1]

[1] The Hong Kong Polytechnic University
[2] SECBIT Labs

Abstract. Most existing folding/accumulation schemes focus on implementing Incrementally Verifiable Computation (IVC). Proof-carrying Data (PCD), as a generalization of IVC, enables sequential computation performance by multiple distrusting parties, thereby offering a robust primitive tool in real-world applications. However, building non-uniform PCD from folding schemes faces many technical challenges, particularly in handling cross terms and preserving zero knowledge.

This paper introduces KiloNova, a non-uniform PCD system with zero knowledge built from generic folding schemes. Motivated by HyperNova (Kothapalli et al. ePrint 2023), we derive a variant of the Customizable Constraint System with linear claims on circuits and inputs to avoid cross terms. With the new constraint system, we propose a generic folding scheme for multiple instances of different circuits and ensure the zero-knowledge with various effective methods. Consequently, we build a non-uniform zero-knowledge PCD scheme from the generic folding scheme and improve its performance with some optimization techniques, such as circuit aggregation and delegation. We propose a new construction for zero-knowledge PCD that does not use a zero-knowledge argument system and has little influence on complexity. The theoretical evaluation shows our non-uniform zk-PCD scheme outperforms previous models. A single multi-scalar multiplication dominates the prover cost at each step. The recursive circuit is dominated by $O(\log(n))$ random-oracle-like hashes and $O(k)$ scalar multiplications, where n is the circuit input length and k is the instance number at each step.

1 Introduction

Recently, there has been a surge of interest in the realization of *Incremental Verifiable Computation* (IVC), a cryptographic primitive that runs sequential computations [1] while allowing efficient verification of the execution at any point. As a generalization of IVC to directed acyclic graphs, the *Proof-Carrying Data* (PCD) enables multiple distrusting parties to perform computations sequentially. This property endows PCD as a more powerful tool in multi-party applications such as distributed computation [2, 3] and blockchain technology [4–6]. Meanwhile, the ability to handle multiple instances in each round provides a broader spectrum of tradeoffs for system performance, as discussed in Protogalaxy [7].

Several effective constructions based on folding/accumulation schemes have been proposed for IVC in recent studies [8–10]. However, they all face efficiency problems caused by cross terms (also known as error terms in [10]) more or less when employed in PCD. Briefly, the cross terms are additional elements introduced in folding non-linear relations. For example, when folding two quadratic instances (w_i, t_i) such that $w_i^2 = t_i$ for $i = 1, 2$, the folded instance $(w, t) = (w_1 + rw_2, t_1 + rt_2)$ with a random challenge r does not satisfy $w^2 = t^2$. The prover has to compute and send an extra term as $2w_1w_2$ for verification. Obviously, the number of cross terms grows in $O(d^s)$ when folding s instances in d-degree relations and raises severe efficiency problems [7]. Moreover, the zk-EVM project [11], one of the most significant applications of IVC/PCD, proposes new requirements for handling different computations at each step, i.e., non-uniform circuits, and providing zero knowledge for privacy preservation. These requirements are difficult to meet efficiently with previous designs.

Folding/Accumulation schemes. The traditional approach for constructing IVC/PCD employs a general-purpose SNARK at each step i to attest the correctness of the proof output by step $i-1$ recursively. This requires implementing the whole verification logic in the SNARK proving circuit (more specifically, the recursive circuit), which incurs a significant overhead as the verification may include costly non-native operations such as elliptic curve pairings [12]. A recent line of work proposes a more practical idea to "defer" the expensive operations in the proof verification and run them together at the end of IVC, including Halo [12], Halo infinite [13], BCMS20 [14], BCLMS21 [15], Nova [8], HyperNova [9], Protostar [10], etc.

The fundamental concept behind these schemes is inspired by batch verification [16], which allows checking multiple proofs in a batch with almost the same cost as checking merely one proof. Concretely, instead of checking the proof from the prior step in the SNARK proving circuit, the prover defers it by applying a so-called folding/accumulation scheme and continues the incremental computation. As a result, the expensive SNARK verification in the recursive circuit is replaced with a cheaper claim to show the correctness of the folding scheme. Finally, the IVC verifier (or the decider), conducts a batch verification for all proofs deferred at each step. We briefly review these approaches and give a rough classification based on their explicit implementations in Figure 1.

The main block in blue outlines the common process of generating a (zk)SNARK proof with four main stages. Decided by the position at which "deferring" occurs, we marked the process with corresponding techniques and their related work. As a result, the schemes based on different techniques exhibit varying performance levels. Generally speaking, for techniques in the earlier stages, such as folding schemes in Nova [8], Protostar [10], and our work, the interactions between the prover and the verifier are relatively simple. Since most of the expensive computation is deferred to the final verifier, the prover has fewer computations and a smaller recursive circuit. Conversely, as techniques in the later stages only defer the instantiated polynomial oracles (it is sufficient to only discuss polynomials

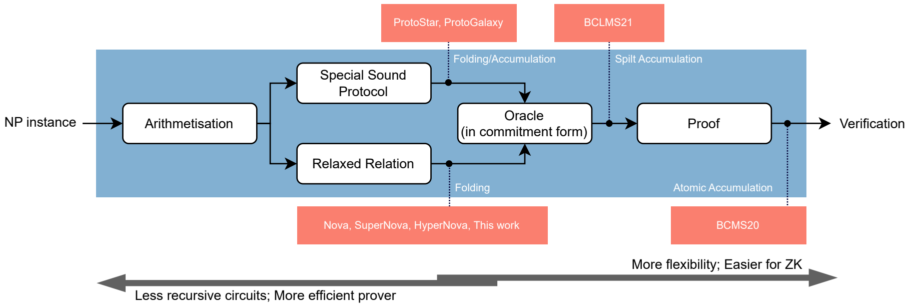

Fig. 1: Overview of existing folding/accumulation schemes.

because all known designs are based on the polynomial Interactive Oracle Proof model [17]), such as BCMS20 [14] and BCLMS21 [15], their verifiers are more likely to be homomorphic, i.e., less cross terms. As a result, these techniques become more flexible to batch multiple instances with different circuits and make it easier to achieve zero-knowledge properties.

For ease of exposition, we specify the notation "folding" for schemes that defer verification for instances without instantiations of polynomial oracles. Therefore, all schemes in Figure 1 except BCLMS21 [15], BCMS20 [14] and Halo [12] are counted as folding schemes. Correspondingly, we denote "accumulation" for schemes that already instantiated oracles and batch them in their instances, such as BCLMS21 and BCMS20. Although the difference seems trivial, it may significantly affect the performance of real-world applications. This is because commitment schemes are used to implement oracles in most modern systems from Interactive Oracle Proof, and the recursive circuits have to deal with non-native computation of commitments with expensive costs [12]. We illustrate this by comparing the performance of HyperNova and BCLMS21. In HyperNova, the folding algorithm (represented with the recursive circuit) only computes *one* group operation when folding two CCS instances, while the accumulation algorithm in BCLMS21 has to compute $O(t)$ group operation, where t is the matrix number. Therefore, one of the goals of this paper is to explore the position for balancing the performance, i.e., propose a technique with a relatively small prover cost (recursion overhead) while fulfilling the requirements for handling multiple non-uniform instances and retaining zero knowledge.

Challenges. Based on the overview above, we explain the challenges of constructing a PCD scheme from the folding/accumulation techniques. To construct a PCD, Halo [12] and its follow-up schemes [13,18] use recursive SNARKs based on efficient polynomial IOPs (Sonic [19] and Plonk [20]). To further reduce the recursion overhead, BCMS20 [14] and BCLMS21 [15] introduce atomic accumulation and split accumulation schemes to accumulate the expensive part of the verification of SNARK proofs (BCLMS21 removes the succinctness requirement and applies a NARK). However, these are only designed for R1CS relations. Until now, few approaches have achieved PCD based on Nova's folding scheme [8] due to the excessive cross terms introduced when folding multiple instances. To the best of our knowledge, the only work that explicitly constructs PCD from the folding scheme is [21], which extends HyperNova to PCD with some optimization for proving complexity. Unfortunately, it cannot deal with non-uniform circuits for zk-EVM and fails to provide zero knowledge.

To attest universal machine executions, SuperNova [22] realizes a non-uniform IVC by enhancing Nova [8] with a selector for the list of predefined functions (i.e., instructions). Protostar [10] introduces a more expressive folding/accumulation scheme for special sound protocols supporting Plonkish relations, presenting a non-uniform IVC. Its major drawback lies in the exponential growth in the number of cross terms with the number of instances, which hinders the construction of PCD from Protostar. Protogalaxy [7] reduces the cost when folding multiple instances by leveraging the property of the Lagrange base, thus making the recursion overhead tolerable for multi-instance situations. Though Protogalaxy seems to be a promising candidate for constructing non-uniform PCD, it still faces practical obstacles when considering explicit construction, such as the efficiency of generating Lagrange bases and proving the correctness of folding under Lagrange bases in the recursive circuit.

In addition to the difficulties in supporting non-uniform circuits, adding zero knowledge into folding-based IVC/PCD schemes is also challenging. Since most existing schemes focus on scenarios involving one prover, they only need to provide zero knowledge for the final verifier by applying a general-purpose zk-SNARK at the end of IVC. Consequently, for IVC/PCD systems run by multiple distrusting parties, zero knowledge between each two parties is not guaranteed. Such defect prevents a wider range of applications in privacy-preserving scenarios, such as anonymous De-Fi, confidential transactions, and trustless cross-chain bridges [23–25].

Based on the findings of Bünz et al. [15], it is efficient to compile any Non-Interactive Argument of Knowledge (NARK) with a folding/accumulation scheme into an IVC/PCD scheme. Two conditions are required to further achieve zero-knowledge: (1) the folding/accumulation scheme is zero-knowledge, and (2) the NARK itself is zero-knowledge. The predicament in achieving zero knowledge for folding-based IVC/PCD lies in the additional computation and communication costs incurred when transforming NARK instances into zero-knowledge forms. Specifically, attempting to achieve zero knowledge by simply masking the witness will also exacerbate the "cross term" problem in the folding schemes as mentioned above. Since this zero-knowledge transformation must be carried out before the folding scheme, we can not apply existing techniques in [7, 9] to reduce the number of cross terms. We elaborate on this predicament with more details in Section 4.3. In summary, proposing an efficient design for realizing zero-knowledge IVC/PCD schemes has both theoretical and practical merits.

From the above discussion, we derive our primary research questions as:

- Can we build an efficient PCD from folding schemes?
- Can we leverage PCD to handle non-uniform circuits for zk-EVM?
- Can we realize zero knowledge for the obtained PCD?

1.1 Our Approach

We answer all questions above positively and present KiloNova, a non-uniform zero-knowledge PCD from generic folding schemes. Our approach improves and generalizes HyperNova [9], an IVC from multi-folding schemes for Customizable Constraint System (CCS) instances [26]. To achieve this, we first design a folding scheme. The following theorem captures its cryptographic and efficiency characteristics.

Theorem 1. *There exists a constant-round, public-coin, zero-knowledge folding scheme for multiple relaxed CCS instances with non-uniform "structure" (i.e., CCS coefficient matrices). The prover's work is $O_\lambda(s \cdot N)$, and the verifier's work and the communication are both $O_\lambda(\log N)$, assuming the existence of any additively-homomorphic commitment scheme that provides $O_\lambda(1)$-sized commitments to N-sized vectors over $\mathbb{F}$ (e.g., Pedersen's commitments), where λ is the security parameter and s is the number of instances.*

Since the folding scheme is public coin, it can be transformed into a non-interactive version with statistical zero-knowledge in the random oracle model. Based on the non-interactive folding scheme, we can further build KiloNova, a zero-knowledge PCD scheme for a class of compliance predicates with a constant depth. We present the main techniques as follows.

(1) Relaxed CCS relation with efficient proofs. Motivated by the ideas in Nova [8] and HyperNova [9], we introduce a new relaxed CCS relation to enable our IVC scheme to deal with non-uniform circuits. Similar to the linearized committed CCS relation in HyperNova, this new relation is also reduced from the original CCS relation [26] by partially running an "early stopping" version of SuperSpartan [9]. The difference is that our protocol runs an extra round of sum-check protocol than HyperNova, which stops right before the oracle queries at the last step of the sum-check protocol. This modification leaves the verifier with instances containing independent linear claims of the inputs and circuit constraints, enabling efficient verification of folding multiple non-uniform instances (smaller recursive circuit) without cross terms and commitments of oracles. We denote the new relation as atomic CCS relations and present corresponding special sound protocols.

(2) Generic folding scheme with zero-knowledge. Based on the atomic CCS relations, we consider the folding process for multiple committed CCS or atomic CCS instances. Generally speaking, the folding scheme executes a special sound protocol for each instance in parallel and aggregates their sum-check protocols into one, except for the final queries. The remaining expensive query operations are folded into one instance, and the verification is "deferred" to the final verifier. Therefore, in each step of the recursive circuit, the prover only needs to claim the query results without proving their validity and fold them with a linear number of field operations.

Regarding zero-knowledge, we ensure the privacy of witnesses in each step of folding using various techniques. First, we adopt an existing approach to ensure zero-knowledge for sum-check protocols [27]. Second, for the final linear claims of sum-check protocols, we apply the random padding scheme in [28] to avoid extra computation resulting from the non-linearity parts in CCS relations. Finally, we use a masking instance folded with other instances to ensure the zero-knowledge of the folded instance.

(3) Non-uniform zk-PCD with decoupled circuits. We propose a non-uniform PCD that enables runtime circuit selection with the proving cost and recursive overhead independent of the sizes of "uninvoked" circuits. Additionally, we propose two optimization techniques to improve the performance of PCD system. The first aggregates multiple claims on atomic CCS matrices into one, reducing the recursive circuit for the subsequent node and the communication of different nodes in PCD system. The second runs an extra IVC in parallel to delegate the computation of structure folds (folding atomic CCS matrices) to a more powerful third party, thereby reducing the proving and communication costs for nodes in PCD system. To achieve zero knowledge for PCD, we modify the existing PCD scheme with split recursive circuits, allowing the prover to preserve zero knowledge for the witness by the zero-knowledge folding scheme rather than apply another zero-knowledge argument system. This new construction offers the first known approach to implementing zk-PCD from folding schemes. It can also be applied to IVC systems built from multi-folding schemes.

1.2 Performance Evaluation

We first compare the functionality of our approach with most of the known folding/accumulation schemes in Table 1. Our work proves in the same CCS language of HyperNova, which is expressive to generalize Plonkish, R1CS, and AIR without overheads simultaneously. Other expressive schemes, such as Protostar [10] and Protogalaxy [7], apply special sound protocols (SPS) that can express CCS language. In addition, our scheme efficiently supports both the non-uniform circuits and multi-folding, which are only known to be practical in BCLMS21 [15] and Protogalaxy [7]. However, no known construction achieves zero knowledge based on Protogalaxy. The last column indicates whether the scheme can achieve ZK-IVC or zk-PCD. Thus, schemes instantiated with zkSNARKs for proving the IVC/PCD proofs, such as Nova [8], do not have zero knowledge. Although the BCLMS21 [15] provides the same functionalities as our scheme, it is constructed from the split accumulation instead of the folding scheme, leading to a larger recursive circuit due to group operations. We illustrate this point with a concrete performance comparison at the end of this subsection.

Next, we compare the performance of our generic folding scheme with other recent work. Due to the different functionalities of these schemes, our first comparison involves an IVC system in CCS relations, where the system folds one new instance in each step and does not support non-uniform circuits and zero knowledge. The theoretical complexities of the IVC systems are given in Table 2. Note that the performance of our solution is commensurate with that of HyperNova. For degree d CCS instances with $m \times n$ circuit matrices, the prover needs

Table 1: Functionality comparison between existing folding/accumulation schemes

Schemes	**Language**	**Non-uniform**	**Multi-Folding**	**ZK**
Nova [8]	R1CS	No	No	No
SuperNova [22]	R1CS	Yes	No	No
HyperNova [9]	CCS	No	No/Yes in [21]	No
BCLMS21 [15]	R1CS	Yes	Yes	Yes
Protostar [10]	Degree-d gate	Yes	No/Expensive	No
Protogalaxy [7]	Degree-d gate	Yes	Yes	No
KiloNova	CCS	Yes	Yes	Yes

to compute a multi-scalar multiplication with $|\mathsf{wit}|$ $\mathbb{G}$ operations, where $|\mathsf{wit}|$ denotes the number of non-zero elements in the witness. For the recursive part, our scheme performs $\log n$ times more random-oracle-like hashes than HyperNova due to the additional $\log n$ rounds in the second sum-check protocol. The performance of ProtoStar equals our computation, while its recursive overhead is minimal with only $O(1)$ hashes.

Moreover, when a non-uniform PCD system that folds s many instances with different CCS structures is invoked, KiloNova significantly outperforms other schemes. For HyperNova, its multi-folding scheme can only fold multiple instances with the same CCS structure. Its direct application to non-uniform PCD will generate $O(m \cdot n^2)$ additional cross terms. For ProtoStar, Eagen et al. [7] pointed out that its performance drastically degenerates when handling multiple instances because the verifier degree exponentially increases with instance number as $O(d^s)$.

Table 2: Performance comparison between different IVC schemes

Criteria	**KiloNova**	**HyperNova**	**Protostar**
$\mathcal{P}$ native	$|\mathsf{wit}|$ $\mathbb{G}$ $O(|\mathsf{wit}| d \log^2 d)$ $\mathbb{F}$	$|\mathsf{wit}|$ $\mathbb{G}$ $O(|\mathsf{wit}| d \log^2 d)$ $\mathbb{F}$	$|\mathsf{wit}|$ $\mathbb{G}$ $O(|\mathsf{wit}| d \log^2 d)$ $\mathbb{F}$
$\mathcal{P}$ recursive	1 $\mathbb{G}$ $\log m + \log n$ RO $O(d \log m)$ $\mathbb{F}$	1 $\mathbb{G}$ $\log m$ RO $O(d \log m)$ $\mathbb{F}$	3 $\mathbb{G}$ $O(1)$ RO $(d + O(1))$ $\mathbb{F}$

To manifest the statement above, we further evaluate the performance of non-uniform PCD built from KiloNova and compare it with existing PCD schemes.

Comparison with BCLMS21. Bünz et al. introduce a PCD scheme in BCLMS21 [15] from the split accumulation scheme. Different from the folding scheme, this scheme accumulates the proof of a NARK for R1CS relations. Consequently, the verification cost of the obtained PCD scheme is relatively high, requiring 10 multi-scalar multiplication (MSM) of size m, while KiloNova only requires 1 MSM. In terms of prover cost and recursive overhead, BCLM21 needs to handle $O(r)$ group operations with a larger coefficient than our scheme. However, it avoids logarithmic random oracle queries. Notably, BCLM21 does not support d-degree circuits and lookup operations.

Comparison with Protogalaxy. Recently, another effective accumulation scheme named Protogalaxy [7] has been proposed. As the following-up work of Protostar [10], Protogalaxy reduces the cross terms from $O(d^s)$ to $O(ds)$ when folding s non-uniform instances by replacing the challenges with Lagrange bases at a random point. Moreover, the non-interactive folding scheme in Protogalaxy only requires $O(1)$ random oracle queries. While Protogalaxy appears promising for constructing non-uniform PCD, it faces challenges in explicit constructions. First, it needs to handle cross items with Lagrange bases, yielding a *quasi-linear* prover cost with instance number s and degree d while ours is *linear*. To amend this problem, Protogalaxy proposes an alternative construction based on sum-check by replacing Lagrange bases with $\widetilde{eq}(\cdot)$. However, the sum-check protocol increases the number of RO queries to $O(\log N)$. Besides, the prover cost is still constantly higher than our work because of computing extra s evaluations on $\widetilde{eq}(\cdot)$. Second, Protogalaxy does not support circuit aggregation as we proposed in Section 5.1 since the instances being folded are still non-linear. Lastly, the authors of Protogalaxy neither provide constructions for non-uniform PCD nor add zero knowledge, whereas KiloNova presents explicit descriptions and solves the potential technical problems.

2 Preliminaries

2.1 Notations

In this paper, we use λ to denote the security parameter. Accordingly, $\text{negl}(\lambda)$ denotes an unspecified function that is negligible in λ. We denote by $[n]$ the set $\{1, ..., n\} \subseteq \mathbb{N}$. Let $\mathbb{F}$ denote a finite field, e.g., $\mathbb{F}_p$ is a prime field for a large prime p. The bold-type lower-case letters denote vectors, e.g., $\boldsymbol{a} \in \mathbb{F}^n$ is a vector of elements $a_1, ..., a_n \in \mathbb{F}$. $\boldsymbol{a}[i]$ is also used to denote the i-th element of $\boldsymbol{a}$ when the element is not specified with a concrete value. To represent a set, we use $\{a_i\}_{i=1}^n$ as a short-hand for $\{a_1, ..., a_n\}$. For a finite set S, let $x \leftarrow\!\!\$\ S$ denote sampling x from S uniformly at random. We use "PPT algorithms" to refer to "Probabilistic Polynomial Time Algorithms".

2.2 Definitions for Polynomials

We recall some basic definitions for polynomials from [29] as follows. Let $f(\cdot) : \mathbb{F}^n \to \mathbb{F}$ be a *multivariate polynomial* with n input elements over $\mathbb{F}$, its total degree d is defined as the maximum degree over all monomials in $f(\cdot)$. Moreover, the degree of a polynomial in a specified variable x_i is the maximum exponent that x_i takes in any of the monomials in $f(\cdot)$. Particularly, a multivariate polynomial is a *multilinear* polynomial if the degree of the polynomial in each variable is at most one. To keep consistent with our notation for vectors, we use $f(\boldsymbol{x})$ to denote the polynomial $f(\cdot)$ with the specified input variable as vector $\boldsymbol{x}$. Next, we state the lemmas used in our paper.

Lemma 1 (Multilinear extensions [30]). *Let $f(\cdot) : \{0,1\}^n \to \mathbb{F}$ be a function that maps n-bit elements into an element of $\mathbb{F}$. The multilinear extension of $f(\cdot)$ is a unique multilinear n-variate polynomial $\tilde{f}(\cdot) : \mathbb{F}^n \to \mathbb{F}$ such that $\tilde{f}(\boldsymbol{x}) = f(\boldsymbol{x})$ for all $\boldsymbol{x} \in \{0,1\}^n$, which can be computed as follows.*

$$\tilde{f}(\boldsymbol{x}) = \sum_{\boldsymbol{e} \in \{0,1\}^n} f(\boldsymbol{e}) \cdot \widetilde{eq}(\boldsymbol{x}, \boldsymbol{e}),$$

where $\widetilde{eq}(\boldsymbol{x}, \boldsymbol{e}) = \prod_{i=1}^{n}(x_i \cdot e_i + (1 - x_i) \cdot (1 - e_i))$.

Lemma 2 (Schwartz-Zippel lemma [31]). *Assume $f(\cdot) : \mathbb{F}^n \to \mathbb{F}$ is a non-zero n-variate polynomial of degree at most d. Then on any finite set $S \subseteq \mathbb{F}$,*

$$\mathrm{Pr}_{\boldsymbol{x} \leftarrow_{\$} S^n}[f(\boldsymbol{x} = 0) \leq d/|S|],$$

where $\boldsymbol{x}$ is a randomly sampled vector from S^n and $|S|$ denotes the size of S.

2.3 Sum-check Protocol

The sum-check protocol is an interactive proof proposed by Lund et al. [32]. It has long attracted the attention of practitioners for its desirable performance, especially in a recent study on proof systems with linear proving time [29, 33]. Here, we only briefly review it. More technical details can be referred to [29].

Assume $f(\cdot) : \mathbb{F}^n \to \mathbb{F}$ as an n-variate low-degree polynomial with the max degree of d for each variable. The prover wants to convince the verifier of the following claim:

$$\mathrm{sum} = \sum_{x_1 \in \{0,1\}} \sum_{x_2 \in \{0,1\}} \cdots \sum_{x_n \in \{0,1\}} f(x_1, ..., x_n). \tag{1}$$

If the prover sends the polynomial $f(\cdot)$ directly, the verifier can compute the above sum by evaluating the polynomial on 2^n different inputs. The sum-check protocol provides us with a more efficient probabilistic algorithm with linear verification complexity. Generally speaking, the verifier chooses a random vector $\boldsymbol{r} \in \mathbb{F}^n$ as the challenges for the n-round interactions with the prover. At the final step, the verifier outputs a claim about the evaluation $f(\boldsymbol{r})$, i.e., $c \leftarrow \mathit{\Pi}_{\mathrm{sc}}(f, n, d, \mathrm{sum}, \boldsymbol{r})$. If $c = f(\boldsymbol{r})$ holds, then the verifier is convinced of the claim about the sum of $f(\cdot)$ in Equation (1).

According to previous work [29, 32], the sum-check protocol satisfies both completeness and soundness properties, and its communication cost takes $O(n \cdot d)$ element of $\mathbb{F}$.

2.4 Polynomial Commitment Scheme

We adapt the definition of the polynomial commitment scheme from [BFS20].

Definition 1 (Polynomial commitment (PC)). *A polynomial commitment (PC) scheme for multilinear polynomials is defined as a tuple of four protocols* $\mathsf{PC} = (\mathsf{Gen}, \mathsf{Commit}, \mathsf{Open}, \mathsf{Eval})$*:*

- $\mathsf{Gen}(1^\lambda, \ell) \to \mathsf{pp}$*: takes as input* ℓ *(the number of variables in a multilinear polynomial); produces public parameters* pp.
- $\mathsf{Commit}(\mathsf{pp}, f) \to C$*: takes as input an* ℓ*-variate multilinear polynomial* $f : \mathbb{F}^\ell \to \mathbb{F}$*; produces a commitment* C.
- $\mathsf{Open}(\mathsf{pp}, C, f) \to b$*: verifies the opening of commitment* C *to the* ℓ*-variate multilinear polynomial* f*; outputs* $b \in \{0, 1\}$.
- $\mathsf{Eval}(\mathsf{pp}, C, \boldsymbol{x}, y, \ell, f) \to b$ *is a protocol between a PPT prover* $\mathcal{P}$ *and verifier* $\mathcal{V}$*. Both* $\mathcal{V}$ *and* $\mathcal{P}$ *hold a commitment* C*, the number of variables* ℓ*, a scalar* $y \in \mathbb{F}$*, and* $\boldsymbol{x} \in \mathbb{F}^\ell$*.* $\mathcal{P}$ *additionally knows an* ℓ*-variate multilinear polynomial* f*.* $\mathcal{P}$ *attempts to convince* $\mathcal{V}$ *that* $f(\boldsymbol{x}) = y$*. At the end of the protocol,* $\mathcal{V}$ *outputs* $b \in \{0, 1\}$.

A PC is an extractable polynomial commitment scheme for multilinear polynomials over a finite field $\mathbb{F}$ if it satisfies completeness, binding, and knowledge soundness properties as defined in Appendix A.

1. Completeness. PC has completeness if for all ℓ-variate multilinear polynomial $g \in \mathbb{F}[\ell]$,

$$\Pr\left[\begin{array}{l|l} \mathsf{Eval}(\mathsf{pp}_{\mathsf{pc}}, C, \ell, r, v; f) = 1 & \mathsf{pp}_{\mathsf{pc}} \leftarrow \mathsf{Setup}(1^\lambda, \ell); \\ \wedge f(r) = v & C \leftarrow \mathsf{Commit}(\mathsf{pp}_{\mathsf{pc}}, f) \end{array}\right] = 1.$$

2. Binding. PC has binding if for any PPT adversary $\mathcal{A}$, size parameter $\ell > 1$,

$$\Pr\left[\begin{array}{l|l} b_0 = b_1 \neq 0 & \mathsf{pp}_{\mathsf{pc}} \leftarrow \mathsf{Setup}(1^\lambda, \ell); (C, f_0, f_1) \leftarrow \mathcal{A}(\mathsf{pp}_{\mathsf{pc}}); \\ \wedge f_0 \neq f_1 & b_0 \leftarrow \mathsf{Open}(\mathsf{pp}_{\mathsf{pc}}, C, f_0); b_1 \leftarrow \mathsf{Open}(\mathsf{pp}_{\mathsf{pc}}, C, f_1) \end{array}\right] \leq \mathsf{negl}(\lambda).$$

3. Knowledge soundness. PC has knowledge soundness if given $\mathsf{pp}_{\mathsf{pc}} \leftarrow \mathsf{Setup}(1^\lambda, \ell)$, Eval is a succinct argument of knowledge for NP relation

$$\mathcal{R}_{\mathsf{Eval}}(\mathsf{pp}_{\mathsf{pc}}) = \{(C, r, v; f) : f \in \mathbb{F}[\ell] \wedge f(r) = v \wedge \mathsf{Open}(\mathsf{pp}_{\mathsf{pc}}, C, f) = 1\}.$$

Definition 2. *A polynomial commitment scheme for multilinear polynomials* $\mathsf{PC} = (\mathsf{Setup}, \mathsf{Commit}, \mathsf{Open}, \mathsf{Eval})$ *is additively homomorphic if for all* ℓ *and public parameters* pp *produced from* $\mathsf{Setup}(1^\lambda, \ell)$*, and for any* $f_1, f_2 : \mathbb{F}^\ell \to \mathbb{F}$*,* $\mathsf{Commit}(\mathsf{pp}, f_1) + \mathsf{Commit}(\mathsf{pp}, f_2) = \mathsf{Commit}(\mathsf{pp}, f_1 + f_2)$.

2.5 Proof-Carrying Data

In this paper, we adopt the definition of PCD from [15,21]. We start with defining some necessary terminologies before presenting the definition.

Definition 3. *A* transcript T *is a directed acyclic graph with each vertex* $u \in V(\mathsf{T})$ *labeled by local data* $z_{\mathsf{loc}}^{(u)}$ *and each edge* $e \in E(\mathsf{T})$ *labeled by a message* $z^{(e)} \neq \perp$*. The output* $o(\mathsf{T})$ *of a transcript* T *is a message* $z^{(e)}$ *where* $e = (u, v)$ *is the lexicographically-first edge such that* v *is a sink.*

Definition 4. *A vertex* $u \in V(\mathsf{T})$ *is* φ-compliant *for* $\varphi \in \mathsf{F}$ *if for all outgoing edges* $e = (u, v) \in E(\mathsf{T})$*:*

- *(base case) if u has no incoming edges, $\varphi(z^{(e)}, z_{\mathsf{loc}}^{(u)}, \perp, ..., \perp)$ accepts,*
- *(recursive case) if u has incoming edges $e_1, ..., e_m$, $\varphi(z^{(e)}, z_{\mathsf{loc}^{(u)}}, z^{(e_1)}, ..., z^{(e_m)})$ accepts.*

We say that T is φ-compliant if all of its vertices are φ-compliant.

Definition 5 (Proof-Carrying Data [21]). *A proof-carrying data scheme for a class of compliance predicates F is a tuple of algorithms $\mathsf{PCD} = (\mathcal{G}, \mathcal{K}, \mathcal{P}, \mathcal{V})$ where*

- *$\mathcal{G}(1^\lambda) \to \mathsf{pp}$ on input security parameter λ, samples and outputs public parameter pp.*
- *$\mathcal{K}(\mathsf{pp}, \varphi) \to (\mathsf{pk}, \mathsf{vk})$ on input public parameter pp and a compliance predicate $\varphi \in \mathsf{F}$, outputs a prover key pk and a verifier key vk.*
- *$\mathcal{P}(\mathsf{pk}, z, z_{\mathsf{loc}}, \{z_i, \mathit{\Pi}_i\}_{i=1}^r) \to \mathit{\Pi}$ on input public key pk, message z of an outgoing edge, local data z_{loc}, messages $\{z_i\}_{i\in[r]}$ of incoming edges and their corresponding proofs $\{\mathit{\Pi}_i\}_{i\in[r]}$, outputs a new proof $\mathit{\Pi}$ to attest the correctness of z.*
- *$\mathcal{V}(\mathsf{vk}, z, \mathit{\Pi}) \to 0/1$ on input verifier key vk, message z and proof $\mathit{\Pi}$, outputs $0/1$ to reject or accept.*

A proof-carrying data scheme PCD should satisfy the perfect completeness, knowledge soundness, and zero-knowledge properties.

1. Perfect Completeness. PCD has perfect completeness if for every adversary $\mathcal{A}$,

$$\Pr\left[\mathcal{V}(\mathsf{vk}, z, \mathit{\Pi}) = 1 \;\middle|\; \begin{array}{r} \mathsf{pp} \leftarrow \mathcal{G}(1^\lambda); \\ (\varphi, z, z_{\mathsf{loc}}, \{z_i, \mathit{\Pi}_i\}_{i=1}^r) \leftarrow \mathcal{A}(\mathsf{pp}); \\ (\mathsf{pk}, \mathsf{vk}) \leftarrow \mathcal{K}(\mathsf{pp}, \varphi); \\ \varphi \in \mathsf{F}; \varphi(z, z_{\mathsf{loc}}, \{z_i\}_{i=1}^r) = 1; \\ \forall i \in [r], z_i = \perp \text{ or } \mathcal{V}(\mathsf{vk}, z_i, \mathit{\Pi}_i) = 1; \\ \mathit{\Pi} \leftarrow \mathcal{P}(\mathsf{pk}, z, z_{\mathsf{loc}}, \{z_i, \mathit{\Pi}_i\}_{i=1}^r) \end{array}\right] = 1.$$

2. Knowledge soundness. PCD has knowledge soundness (w.r.t. an auxiliary input distribution $\mathcal{D}$) if for every expected polynomial time adversary $\mathcal{P}^*$, there exists an expected polynomial time extractor $\mathsf{Ext}_{\mathcal{P}^*}$ such that for every set Z,

$$\Pr\left[\begin{array}{l} \varphi \in \mathsf{F} \\ \wedge(\mathsf{pp}, \mathsf{ai}, \varphi, \circ(\mathsf{T}), \mathsf{ao}) \in Z \\ \wedge \mathsf{T} \text{ is } \varphi\text{-compliant} \end{array} \;\middle|\; \begin{array}{r} \mathsf{pp} \leftarrow \mathcal{G}(1^\lambda); \\ \mathsf{ai} \leftarrow \mathcal{D}(\mathsf{pp}); \\ (\varphi, \mathsf{T}, \mathsf{ao}) \leftarrow \mathsf{Ext}_{\mathcal{P}^*}(\mathsf{pp}, \mathsf{ao}) \end{array}\right] \geq$$

$$\Pr\left[\begin{array}{l} \varphi \in \mathsf{F} \\ \wedge(\mathsf{pp}, \mathsf{ai}, \varphi, \circ, \mathsf{ao}) \in Z \\ \wedge \mathcal{V}(\mathsf{vk}, \circ, \mathit{\Pi}) = 1 \end{array} \;\middle|\; \begin{array}{r} \mathsf{pp} \leftarrow \mathcal{G}(1^\lambda); \\ \mathsf{ai} \leftarrow \mathcal{D}(\mathsf{pp}); \\ (\varphi, \circ, \mathit{\Pi}, \mathsf{ao}) \leftarrow \mathcal{P}^*(\mathsf{pp}, \mathsf{ai}); \\ (\mathsf{pk}, \mathsf{vk}) \leftarrow \mathcal{K}(\mathsf{pp}, \varphi) \end{array}\right] - \mathsf{negl}(\lambda).$$

3. Zero Knowledge. PCD has (statistical) zero knowledge if there exists a probabilistic polynomial-time simulator Sim such that for every polynomial-size honest adversary $\mathcal{A}$ the distributions below are computationally indistinguishable:

$$\left\{ (\mathsf{pp}, \Pi) \;\middle|\; \begin{array}{r} \mathsf{pp} \leftarrow \mathcal{G}(1^\lambda); \\ (\varphi, z, z_{\mathsf{loc}}, [z_i, \Pi_i]_{i=1}^r) \leftarrow \mathcal{A}(\mathsf{pp}); \\ (\mathsf{pk}, \mathsf{vk}) \leftarrow \mathcal{K}(\mathsf{pp}, \varphi); \\ \Pi \leftarrow \mathcal{P}(\mathsf{pk}, \varphi, z, z_{\mathsf{loc}}, [z_i, \Pi_i]_{i=1}^r) \end{array} \right\}$$

and

$$\left\{ (\mathsf{pp}, \Pi) \;\middle|\; \begin{array}{r} (\mathsf{pp}, \tau) \leftarrow \mathsf{Sim}(1^\lambda); \\ (\varphi, z, z_{\mathsf{loc}}, [z_i, \Pi_i]_{i=1}^r) \leftarrow \mathcal{A}(\mathsf{pp}); \\ \Pi \leftarrow \mathsf{Sim}(\mathsf{pp}, \varphi, z, \tau) \end{array} \right\}.$$

2.6 Customizable Constraint Systems

The customizable constraint system (CCS) is an intermediate representation of arithmetic circuits introduced by Setty et al. [26], which can simultaneously generalize R1CS, Plonkish, and AIR without overheads. However, directly implementing the CCS relation into a zero-knowledge proof is neither straightforward nor efficient. For modern SNARKs, practitioners usually combine a polynomial IOP [34] with a polynomial commitment scheme [35]. Therefore, encoding the CCS relation into low-degree polynomials and committing them correspondingly will accommodate it to a more friendly form for building zero-knowledge proofs. To fulfill these requirements, we present the definitions of the CCS relation and committed CCS relation following HyperNova [9] in this part.

Consider a CCS structure $\mathcal{S} = (m, n, N, l, t, q, d, \{M_j\}_{j\in[t]}, \{S_i\}_{i\in[q]}, \{c_i\}_{i\in[q]})$. Let $s_x = \log m$ and $s_y = \log n$. We interpret each M_j (for $j \in [t]$) as functions with the following signature: $\{0,1\}^{s_x} \times \{0,1\}^{s_y} \to \mathbb{F}$. For $j \in [t]$, let $\widetilde{M_j}$ denote the multilinear extension (MLE) of M_j i.e., $\widetilde{M_j}$ is the unique multilinear polynomial in $s_x + s_y$ variables such that

$$\widetilde{M_j}(\boldsymbol{x}, \boldsymbol{y}) = M_j(\boldsymbol{x}, \boldsymbol{y}), \forall \boldsymbol{x} \in \{0,1\}^{s_x}, \boldsymbol{y} \in \{0,1\}^{s_y}.$$

Similarly, for a purported witness $\mathsf{wit} \in \mathbb{F}^{n-l-1}$, let $\widetilde{w}$ denote the unique MLE of wit viewed as a function. WLOG, let $|\mathsf{wit}| = l + 1$. For ease of exposition, a CCS instance is split into a fixed "structure" $\mathcal{S}$ that describes constraints and a "context" consisting of the public input and output and other public parameters depending on concrete instances. Note that we use the different notion "context" from [9, 26] for clarity. The definitions are given below.

Definition 6 (CCS [26]). *We define the customizable constraint system (CCS) relation $\mathcal{R}_{\mathsf{CCS}}$ as follows. Let the public parameter consist of size bounds $m, n, N, l, t, q, d \in \mathbb{N}$ where $n > l$.*
An $\mathcal{R}_{\mathsf{CCS}}$ structure $\mathcal{S}$ consists of:

- *a sequence of matrices* $\{M_j \in \mathbb{F}^{m\times n}\}_{j\in[t]}$ *with at most* $N = \Omega(\max(m, n))$ *non-zero entries in total;*
- *a sequence of* q *multisets* $\{S_i\}_{i\in[q]}$*, where an element in each multiset is from the domain* $\{1, ..., t\}$ *and the cardinality of each multiset is at most* d*.*
- *a sequence of* q *constants* $\{c_i\}_{i\in[q]}$*, where each constant is from* $\mathbb{F}$*.*

An $\mathcal{R}_{\mathsf{CCS}}$ *instance consists of public input and output* $\mathsf{io} \in \mathbb{F}^l$*.*
An $\mathcal{R}_{\mathsf{CCS}}$ *witness consists of a vector* $\mathsf{wit} \in \mathbb{F}^{n-l-1}$*.*
An $\mathcal{R}_{\mathsf{CCS}}$ *instance (structure-context tuple)* $(\mathcal{S}, \mathsf{io})$ *is satisfied by an* $\mathcal{R}_{\mathsf{CCS}}$ *witness* wit *if*

$$\sum_{i\in[q]} c_i \cdot \bigcirc_{j\in S_i} M_j \cdot \boldsymbol{z} = \mathbf{0}, \tag{2}$$

where $\boldsymbol{z} = (\mathsf{wit}, 1, \mathsf{io}) \in \mathbb{F}^n$*,* $M_j \cdot \boldsymbol{z}$ *denotes matrix-vector multiplication,* $\bigcirc$ *denotes the Hadamard product between vectors, and* $\mathbf{0}$ *is an* m*-sized vector with entries equal to the additive identity in* $\mathbb{F}$*.*

Definition 7 (Committed CCS). *Let* $\mathsf{PC} = (\mathsf{Gen}, \mathsf{Commit}, \mathsf{Open}, \mathsf{Eval})$ *denote an additively-homomorphic polynomial commitment scheme for multilinear polynomials over a finite field* $\mathbb{F}$*. Denote the public parameters of size bounds as* $m, n, N, l, t, q, d \in \mathbb{N}$ *where* $n = 2 \cdot (l + 1)$ *and* $\mathsf{pp} \leftarrow \mathsf{Gen}(1^\lambda, s_y)$*. The committed customizable constraint system (CCCS) relation* $\mathcal{R}_{\mathsf{CCCS}}$ *is defined as follows.*

- *An* $\mathcal{R}_{\mathsf{CCCS}}$ *structure* $\mathcal{S}$ *consists of:*
  - *a sequence of sparse multilinear polynomials in* s_x+s_y *variables* $\{\widetilde{M}_j\}_{j\in[t]}$ *such that they evaluate to a non-zero value in at most* $N = \Omega(m)$ *locations over the Boolean hypercube* $\{0, 1\}^{s_x} \times \{0, 1\}^{s_y}$*.*
  - *a sequence of* q *multisets* $\{S_i\}_{i\in[q]}$*, where an element in each multiset is from the domain* $\{1, ..., t\}$ *and the cardinality of each multiset is at most* d*.*
  - *a sequence of* q *constants* $\{c_i\}_{i\in[q]}$*, where each constant is from* $\mathbb{F}$*.*
- *An* $\mathcal{R}_{\mathsf{CCCS}}$ *context is* (C, io) *where* C *is a commitment to a multilinear polynomial in* $s_y - 1$ *variables and* $\mathsf{io} \in \mathbb{F}^l$*.*
- *An* $\mathcal{R}_{\mathsf{CCCS}}$ *witness consists of a multilinear polynomial* $\widetilde{\mathsf{wit}}$ *in* s_y-1 *variables.*

An $\mathcal{R}_{\mathsf{CCCS}}$ *instance (structure-context tuple) is satisfied by an* $\mathcal{R}_{\mathsf{CCCS}}$ *witness if* $\mathsf{Commit}(\mathsf{pp}, \widetilde{\mathsf{wit}}) = C$ *and if for all* $\boldsymbol{x} \in \{0, 1\}^{s_x}$*,*

$$\sum_{i\in[q]} c_i \left(\prod_{j\in S_i} \left(\sum_{\boldsymbol{y}\in\{0,1\}^{s_y}} \widetilde{M}_j(\boldsymbol{x}, \boldsymbol{y}) \cdot \tilde{z}(\boldsymbol{y}) \right) \right) = 0, \tag{3}$$

where $\tilde{z}(\boldsymbol{y})$ *is an* s_y*-variate multilinear polynomial such that* $\tilde{z}(\boldsymbol{y}) = \widetilde{(\mathsf{wit}, 1, \mathsf{io})}$ *for all* $\boldsymbol{y} \in \{0, 1\}^{s_y}$*.*

3 Building Blocks

In this section, we describe several essential building blocks for the design of KiloNova. First, we extend the multi-folding scheme introduced in HyperNova [9] to support non-uniform circuit scenarios. The new model is called a *generic folding scheme*. Next, we instantiate a special sound protocol for proving the committed CCS relation in Section 2.6. To fold committed CCS instances with different structures (i.e., non-uniform circuits), we further propose a "relaxed" relation called *atomic CCS relation*. A special sound protocol of the new relation is instantiated as well.

3.1 Generic Folding Schemes

Recall that a folding scheme [KST22] for a relation $\mathcal{R}$ is a protocol between a prover and a verifier that reduces the task of checking two instances in $\mathcal{R}$ with the *same* structure $\mathcal{S}$ into the task of checking a single folded instance in $\mathcal{R}$ also with structure $\mathcal{S}$. Then in HyperNova, the authors introduce a generalization of folding schemes as multi-folding schemes, which can fold two collections of instances in relations $\mathcal{R}^{(1)}$ and $\mathcal{R}^{(2)}$ with the *same* structure $\mathcal{S}$ respectively.

This paper extends the multi-folding scheme to allow it to fold relations with *different* structures. Concretely, a *generic folding scheme* is defined with respect to a set of relations $\{\mathcal{R}^{(i)}\}_{i=1}^{\ell}$ with different structures $\{\mathcal{S}^{(i)}\}_{i=1}^{\ell}$ and size parameters $\{s^{(i)}\}_{i=1}^{\ell}$ (the number of repetition for each $\mathcal{S}^{(i)}$). It is an interactive protocol between a prover and a verifier that reduces the task of checking a collection of $s^{(i)}$ instances in $\mathcal{R}^{(i)}$ for all $i \in [\ell]$ ($\sum_{i\in[\ell]} s^{(i)}$ instances in total) into checking a single folded instance in $\mathcal{R}^*$ with structure $\mathcal{S}^*$. We formally define it below.

Definition 8 (Generic folding schemes). *Consider relations $\{\mathcal{R}^{(i)}\}_{i=1}^{\ell}$ over public parameters, structures, instance, and witness tuples such that each $\mathcal{R}^{(i)}$ has distinct structure $\mathcal{S}^{(i)}$. A generic folding scheme for $\{(\mathcal{R}^{(i)}, s^{(i)})\}_{i=1}^{\ell}$ consists of a PPT generator algorithm $\mathcal{G}$, a deterministic encoder algorithm $\mathcal{K}$, and a pair of PPT algorithms $\mathcal{P}$ and $\mathcal{V}$ denoting the prover and the verifier respectively, with the following interface:*

- $\mathcal{G}(1^\lambda) \to \mathsf{pp}$*: on input security parameter λ, samples public parameters* pp.
- $\mathcal{K}(\mathsf{pp}, \{\mathcal{S}^{(i)}\}_{i=1}^{\ell}) \to (\mathsf{pk}, \mathsf{vk})$*: on input* pp*, and common structures $\{\mathcal{S}^{(i)}\}_{i=1}^{\ell}$ among the instances to be folded, outputs a prover key* pk *and a verifier key* vk.
- $\mathcal{P}(\mathsf{pk}, \{\mathcal{S}^{(i)}, \mathbf{ctx}^{(i)}, \mathbf{wit}^{(i)}\}_{i=1}^{\ell}) \to (\mathcal{S}^*, \mathsf{ctx}^*, \mathsf{wit}^*)$*: on input ℓ vectors of contexts $\{\mathbf{ctx}^{(i)}\}_{i=1}^{\ell}$, where each vector $\mathbf{ctx}^{(i)}$ is in $\mathcal{R}^{(i)}$ with a distinct structure $\mathcal{S}^{(i)}$, and corresponding vector of witnesses $\mathbf{wit}^{(i)}$ for $i \in [\ell]$, outputs a folded context-witness pair* $(\mathsf{ctx}^*, \mathsf{wit}^*)$ *in a new relations $\mathcal{R}^*$ with structure $\mathcal{S}^*$.*
- $\mathcal{V}(\mathsf{vk}, \{\mathcal{S}^{(i)}, \mathbf{ctx}^{(i)}\}_{i=1}^{\ell}) \to (\mathcal{S}^*, \mathsf{ctx}^*)$*: on input ℓ vectors of contexts $\{\mathbf{ctx}^{(i)}\}_{i=1}^{\ell}$, outputs a folded context* ctx^* *in a new relations $\mathcal{R}^*$ with structure $\mathcal{S}^*$.*

Let $\mathit{\Pi}_{\mathsf{fold}}$ denote the interaction between $\mathcal{P}$ and $\mathcal{V}$. Then $\mathit{\Pi}_{\mathsf{fold}}$ is a function that takes as input $((\mathsf{pk}, \mathsf{vk}), \{(\mathcal{S}^{(i)}, \mathbf{\mathsf{ctx}}^{(i)}, \mathbf{\mathsf{wit}}^{(i)})\}_{i=1}^{\ell})$ and runs the interaction on prover input $(\mathsf{pk}, \{(\mathcal{S}^{(i)}, \mathbf{\mathsf{ctx}}^{(i)}, \mathbf{\mathsf{wit}}^{(i)})\}_{i=1}^{\ell})$ and verifier input $(\mathsf{vk}, \{\mathcal{S}^{(i)}, \mathbf{\mathsf{ctx}}^{(i)}\}_{i=1}^{\ell})$. At the end of interaction $\mathit{\Pi}_{\mathsf{fold}}$ outputs $(\mathsf{ctx}^*, \mathsf{wit}^*)$ where ctx^* is the verifier's output folded context, and wit^* is the prover's output folded witness.

We slightly abuse the vector-from denotation $(\mathsf{pp}, \mathcal{S}^{(i)}, \mathbf{\mathsf{ctx}}^{(i)}, \mathbf{\mathsf{wit}}^{(i)}) \in \mathcal{R}^{(i)}$ to represent that $(\mathsf{pp}, \mathcal{S}^{(i)}, \mathsf{ctx}_j^{(i)}, \mathsf{wit}_j^{(i)}) \in \mathcal{R}^{(i)}$ for all $j \in [s^{(i)}]$. A generic folding scheme for $\{\mathcal{R}^{(i)}\}_{i=1}^{\ell}$ satisfies the following requirements.

1. *Perfect Completeness:* For all PPT adversaries $\mathcal{A}$, we have that

$$\Pr\left[\begin{array}{c|l} & \mathsf{pp} \leftarrow \mathcal{G}(1^\lambda), \\ \{(\mathsf{pp}, \mathcal{S}^{(i)}, \mathbf{\mathsf{ctx}}^{(i)}, \mathbf{\mathsf{wit}}^{(i)}) \in \mathcal{R}^{(i)}\}_{i=1}^{\ell} & \{\mathcal{S}^{(i)}, \mathbf{\mathsf{ctx}}^{(i)}, \mathbf{\mathsf{wit}}^{(i)}\}_{i=1}^{\ell} \leftarrow \mathcal{A}(\mathsf{pp}), \\ \Downarrow & (\mathsf{pk}, \mathsf{vk}) \leftarrow \mathcal{K}(\mathsf{pp}, \{\mathcal{S}^{(i)}\}_{i=1}^{\ell}), \\ (\mathsf{pp}, S^*, \mathsf{ctx}^*, \mathsf{wit}^*) \in \mathcal{R}^* & (\mathcal{S}^*, \mathsf{ctx}^*, \mathsf{wit}^*) \\ & \leftarrow \mathit{\Pi}_{\mathsf{fold}}((\mathsf{pk}, \mathsf{vk}), \{\mathcal{S}^{(i)}, \mathbf{\mathsf{ctx}}^{(i)}, \mathbf{\mathsf{wit}}^{(i)}\}_{i=1}^{\ell}) \end{array}\right] = 1.$$

2. *Knowledge Soundness:* For any expected polynomial-time adversaries $\mathcal{A}$ and $\mathcal{P}^*$, $\mathit{\Pi}_{\mathsf{fold}}^*$ is run by $\mathcal{P}^*, \mathcal{V}$, there is an expected polynomial-time extractor Ext such that for all randomness ρ

$$\Pr\left[\begin{array}{c|l} & \mathsf{pp} \leftarrow \mathcal{G}(1^\lambda), \\ \{(\mathsf{pp}, \mathcal{S}^{(i)}, \mathbf{\mathsf{ctx}}^{(i)}, \mathbf{\mathsf{wit}}^{(i)}) \in \mathcal{R}^{(i)}\}_{i=1}^{\ell} & (\{\mathcal{S}^{(i)}, \mathbf{\mathsf{ctx}}^{(i)}\}_{i=1}^{\ell}, \mathsf{st}) \leftarrow \mathcal{A}(\mathsf{pp}, \rho), \\ & \{\mathbf{\mathsf{wit}}^{(i)}\}_{i \in [\ell]} \leftarrow \mathsf{Ext}(\mathsf{pp}, \rho) \end{array}\right] \approx$$

$$\Pr\left[\begin{array}{c|l} & \mathsf{pp} \leftarrow \mathcal{G}(1^\lambda), \\ (\mathsf{pp}, \mathcal{S}^*, \mathsf{ctx}^*, \mathsf{wit}^*) \in \mathcal{R}^* & (\{\mathcal{S}^{(i)}, \mathbf{\mathsf{ctx}}^{(i)}\}_{i=1}^{\ell}, \mathsf{st}) \leftarrow \mathcal{A}(\mathsf{pp}, \rho), \\ & (\mathsf{pk}, \mathsf{vk}) \leftarrow \mathcal{K}(\mathsf{pp}, \{\mathcal{S}^{(i)}\}_{i=1}^{\ell}), \\ & (\mathcal{S}^*, \mathsf{ctx}^*, \mathsf{wit}^*) \leftarrow \mathit{\Pi}_{\mathsf{fold}}^*((\mathsf{pk}, \mathsf{vk}), \{\mathcal{S}^{(i)}, \mathbf{\mathsf{ctx}}^{(i)}\}_{i=1}^{\ell}, \mathsf{st}) \end{array}\right].$$

3. *Efficiency:* The communication costs and $\mathcal{V}$'s computation are lower in the case where $\mathcal{V}$ participates in the generic folding scheme and then checks a witness sent by $\mathcal{P}$ for the folded instance than in the case where $\mathcal{V}$ checks witnesses sent by $\mathcal{P}$ for each of the original instances.

A generic folding scheme is secure in the random oracle model if the above requirements hold when all parties are provided access to a random oracle.

Definition 9 (Honest Verifier Zero-knowledge). *Let* $\mathsf{trace}(\mathit{\Pi}_{\mathsf{fold}}, \mathsf{input})$ *denote the non-deterministic function which takes as input an interaction function* $\mathit{\Pi}_{\mathsf{fold}}$ *and a prescribed input* input*, and produces an interaction transcript between* $\mathcal{P}$ *and* $\mathcal{V}$ *on* input*. A generic folding scheme* $(\mathcal{G}, \mathcal{K}, \mathcal{P}, \mathcal{V})$ *for* $\{R^{(i)}, s^{(i)}\}_{i=1}^{\ell}$ *satisfies honest verifier zero-knowledge if there exists a PPT simulator* Sim *such that for all PPT adversaries* $\mathcal{A}$*, the following distributions are (statistically/computationally)* *indistinguishable*

$$\left\{ (\mathsf{pp}, \{\mathcal{S}^{(i)}, \mathbf{ctx}^{(i)}\}_{i=1}^{\ell}, \mathsf{tr}) \;\middle|\; \begin{array}{l} \mathsf{pp} \leftarrow \mathcal{G}(1^{\lambda}), \\ (\{\mathcal{S}^{(i)}, \mathbf{ctx}^{(i)}, \mathbf{wit}^{(i)}\}_{i=1}^{\ell}) \leftarrow \mathcal{A}(\mathsf{pp}), \\ \{(\mathsf{pp}, \mathcal{S}^{(i)}, \mathbf{ctx}^{(i)}, \mathbf{wit}^{(i)}) \in \mathcal{R}^{(i)}\}_{i=1}^{\ell}, \\ (\mathsf{pk}, \mathsf{vk}) \leftarrow \mathcal{K}(\mathsf{pp}, \{\mathcal{S}^{(i)}\}_{i=1}^{\ell}), \\ \mathsf{tr} \leftarrow \mathsf{trace}(\Pi_{\mathsf{fold}}, ((\mathsf{pk}, \mathsf{vk}), \{\mathcal{S}^{(i)}, \mathbf{ctx}^{(i)}, \mathbf{wit}^{(i)}\}_{i=1}^{\ell}) \end{array} \right\}$$

and

$$\left\{ (\mathsf{pp}, \{\mathcal{S}^{(i)}, \mathbf{ctx}^{(i)}\}_{i=1}^{\ell}, \mathsf{tr}) \;\middle|\; \begin{array}{l} (\mathsf{pp}, \tau) \leftarrow \mathsf{Sim}(1^{\lambda}), \\ (\{\mathcal{S}^{(i)}, \mathbf{ctx}^{(i)}\}_{i=1}^{\ell}, \mathsf{st}) \leftarrow \mathcal{A}(\mathsf{pp}), \\ \{(\mathsf{pp}, \mathcal{S}^{(i)}, \mathbf{ctx}^{(i)}, \mathbf{wit}^{(i)}) \in \mathcal{R}^{(i)}\}_{i=1}^{\ell}, \\ \mathsf{tr} \leftarrow \mathsf{Sim}(\mathsf{pp}, \{\mathcal{S}^{(i)}, \mathbf{ctx}^{(i)}\}_{i=1}^{\ell}, \tau) \end{array} \right\}.$$

Definition 10 (Non-interactive). *A generic folding scheme $(\mathcal{G}, \mathcal{K}, \mathcal{P}, \mathcal{V})$ is non-interactive if the interaction between $\mathcal{P}$ and $\mathcal{V}$ consists of a single message from $\mathcal{P}$ to $\mathcal{V}$. This single message is denoted as $\mathcal{P}$'s output and as $\mathcal{V}$'s input.*

Definition 11 (Public coin). *A generic folding scheme $(\mathcal{G}, \mathcal{K}, \mathcal{P}, \mathcal{V})$ is called public coin if all the messages sent from $\mathcal{V}$ to $\mathcal{P}$ are sampled from a uniform distribution.*

3.2 Special Sound Protocol for Committed CCS

This part describes special sound protocols for the committed CCS relation $\mathcal{R}_{\mathsf{CCCS}}$. The basic idea is to commit the special sound protocol in SuperSpartan [26]. Different from the general-purpose protocol described in Protostar [10], the special sound protocol we used is specified for concrete CCS relations because the relation itself is already expressive enough. Note that Protostar also covers CCS relations with their special sound protocol to manifest expressiveness. While their protocols are not based on sum-check protocols. And the performance of the final scheme is still restricted by the accumulation scheme they proposed.

We instantiate a protocol with a series of interactions between two parties $(\mathcal{P}, \mathcal{V})$, checking the validity of relation $\mathcal{R}_{\mathsf{CCCS}}$ by running sum-check protocols on the target multi-variate polynomials. The running steps of the protocol are given in Π_{CCCS} below. Specifically, the prover wants to convince that Equation (3) holds for all $\boldsymbol{x} \in \{0,1\}^{s_x}$. To check this equation with sum-check protocol, a trick is introduced in Spartan [29]: if multiply each value with a corresponding term $\widetilde{eq}(\boldsymbol{\alpha}, \boldsymbol{x})$ with random chosen $\boldsymbol{\alpha}$, then their sum equals to zero only when all values equal to zero with high probability. Formally, denote $\widetilde{F}(\boldsymbol{x})$ as

$$\widetilde{F}(\boldsymbol{x}) = \sum_{i \in [q]} c_i \left(\prod_{j \in S_i} \left(\sum_{\boldsymbol{y} \in \{0,1\}^{s_y}} \widetilde{M}_j(\boldsymbol{x}, \boldsymbol{y}) \cdot \tilde{z}(\boldsymbol{y}) \right) \right),$$

denote $\widetilde{Q}(\boldsymbol{t})$ as

$$\widetilde{Q}(\boldsymbol{t}) = \sum_{\boldsymbol{x} \in \{0,1\}^{s_x}} \widetilde{F}(\boldsymbol{x}) \cdot \widetilde{eq}(\boldsymbol{t}, \boldsymbol{x}),$$

where $\widetilde{eq}(\boldsymbol{t}, \boldsymbol{x}) = \prod_{i=1}^{s_x}(t_i \cdot x_i + (1-t_i)\cdot(1-x_i))$. Note that $Q(\boldsymbol{t})$ is a multivariate polynomial evaluates to $\widetilde{F}(\boldsymbol{t})$ for all $\boldsymbol{t} \in \{0,1\}^{s_x}$. Therefore, $Q(\boldsymbol{t})$ is a zero-polynomial if and only if $\widetilde{F}(\boldsymbol{x})$ evaluates to zero everywhere on $\boldsymbol{x} \in \{0,1\}^{s_x}$ (that is, the CCS relation is satisfied). To check whether $Q(\boldsymbol{t})$ is a zero-polynomial, it is sufficient to query its value on a random input $\boldsymbol{t} = \boldsymbol{\alpha}$ with an acceptable soundness error.

Lemma 3. $\Pr_{\boldsymbol{\alpha}}\{Q(\alpha) = 0 \mid \exists \boldsymbol{x} \in \{0,1\}^{s_x} \text{ s.t. } \widetilde{F}(\boldsymbol{x}) \neq 0\} \leq \log m/|\mathbb{F}|$.

Proof. Refers to the proof of Lemma 4.3 in Spartan [29]. □

Special Sound Protocol $\mathit{\Pi}_{\mathsf{CCCS}} = (\mathcal{P}, \mathcal{V})$ for relation $\mathcal{R}_{\mathsf{CCCS}}$

1. $\mathcal{V}$: Sample $\boldsymbol{\alpha} \leftarrow\$\ \mathbb{F}^{s_x}$ and send to $\mathcal{P}$.
2. $\mathcal{V}$: Sample $\boldsymbol{r}_x \leftarrow\$\ \mathbb{F}^{s_x}$.
3. $\mathcal{P}$: Compute $\tilde{z}(\boldsymbol{y}) = (\widetilde{\mathsf{wit}, 1, \mathsf{io}})$.
4. **Sum-check#1.** Run $c_x \leftarrow \mathit{\Pi}_{\mathsf{sc}}(f, s_x, d+1, \mathrm{sum}_x, \boldsymbol{r}_x)$ where:

$$f(\boldsymbol{x}) = \widetilde{eq}(\boldsymbol{\alpha}, \boldsymbol{x}) \cdot \left(\sum_{i\in[q]} c_i \cdot \prod_{j\in S_i} \left(\sum_{\boldsymbol{y}\in\{0,1\}^{s_y}} \widetilde{M}_j(\boldsymbol{x}, \boldsymbol{y}) \cdot \tilde{z}(\boldsymbol{y})\right)\right).$$

5. $\mathcal{P}$: Compute $\{\sigma_j\}_{j\in[t]}$ and send to $\mathcal{V}$, where for all $j \in [t]$:

$$\sigma_j = \sum_{\boldsymbol{y}\in\{0,1\}^{s_y}} \widetilde{M}_j(\boldsymbol{r}_x, \boldsymbol{y}) \cdot \tilde{z}(\boldsymbol{y}).$$

6. $\mathcal{V}$: Compute $e \leftarrow \widetilde{eq}(\boldsymbol{\alpha}, \boldsymbol{r}_x)$, and abort if:

$$c_x \neq e \cdot \sum_{i\in[q]} c_i \cdot \prod_{j\in S_i} \sigma_j.$$

7. $\mathcal{V}$: Sample $\delta \leftarrow\$\ \mathbb{F}$, and send to $\mathcal{P}$.
8. $\mathcal{V}$: Sample $\boldsymbol{r}_y \leftarrow\$\ \mathbb{F}^{s_y}$.
9. **Sum-check#2.** Run $c_y \leftarrow \mathit{\Pi}_{\mathsf{sc}}(g, s_y, 2, \mathrm{sum}_y, \boldsymbol{r}_y)$ where:

$$g(\boldsymbol{y}) = \sum_{j\in[t]} \delta^j \cdot \widetilde{M}_j(\boldsymbol{r}_x, \boldsymbol{y}) \cdot \tilde{z}(\boldsymbol{y}).$$

10. $\mathcal{P}$: Compute $\epsilon, \{\theta_j\}_{j\in[t]}$ and send to $\mathcal{V}$, where for all $j \in [t]$:

$$\epsilon = \tilde{z}(\boldsymbol{r}_y),\ \theta_j = \widetilde{M}_j(\boldsymbol{r}_x, \boldsymbol{r}_y).$$

11. $\mathcal{V}$: Abort if:

$$c_y \neq \sum_{j\in[t]} \delta^j \cdot \theta_j \cdot \epsilon.$$

12. $\mathcal{P}$: Open the witness $\widetilde{\mathsf{wit}}$.
13. $\mathcal{V}$: Check that

$$(1)\ \mathsf{Commit}(\mathsf{pp}, \widetilde{\mathsf{wit}}) = C$$

$$(2)\ \epsilon = \tilde{z}(\boldsymbol{r}_y),\ \theta_j = \widetilde{M}_j(\boldsymbol{r}_x, \boldsymbol{r}_y).$$

As a result, the prover runs the first sum-check protocol at step 4 on the polynomial $f(\boldsymbol{x})$ with the randomness $\boldsymbol{\alpha}, \boldsymbol{r}_x$ given by the verifier at step 1 and 2, where $\text{sum}_x = 0$.

To evaluate $f(\boldsymbol{r}_x)$, the verifier needs to know the value of $\sum_{\boldsymbol{y}\in\{0,1\}^{s_y}} \widetilde{M}_j(\boldsymbol{x}, \boldsymbol{y})\cdot \tilde{z}(\boldsymbol{y})$ for all $j \in [t]$, which can be reduced to another sum-check problem. Thus, the prover makes t separate claims to the sums on $\boldsymbol{y} \in \{0,1\}^{s_y}$ to the verifier at step 5. The verifier checks two facts accordingly: (1) c_x in sum-check#1 is consistent with the above claims (step 6), and (2) the t claims are valid.

The first fact is verified directly at step 6. For the second fact, a naive approach is to run t more times the sum-check protocol in parallel for t claims. A more elegant solution aggregates these claims by linear combination with weights $[\delta^1, ..., \delta^t]$ generated from a random δ. Consequently, the prover and verifier can run the sum-check protocol only once on the aggregated multi-variate polynomial $g(\boldsymbol{y})$ at step 9 with the randomness $\boldsymbol{r}_y$ and $\text{sum}_y = \sum_{j\in[t]} \delta^j \cdot \sigma_j$. Likewisely, the prover makes claims to t evaluations $\{M_j(\boldsymbol{r}_x, \boldsymbol{r}_y)\}_{j=1}^t$ and one evaluation $\tilde{z}(\boldsymbol{r}_y)$ at steps 10. The verifier checks accordingly by running step 11 and computing the evaluations at step 13.

The security properties of the protocol $\mathit{\Pi}_{\mathsf{CCCS}}$ are guaranteed as follows:

- Completeness. $\mathit{\Pi}_{\mathsf{CCCS}}$ satisfies perfect completeness.
- Knowledge Soundness. $\mathit{\Pi}_{\mathsf{CCCS}}$ is a knowledge sound protocol for $\mathcal{R}_{\mathsf{CCCS}}$ if the commitment scheme $\mathsf{Commit}()$ satisfies the binding property. To prove it, let $\mathsf{Ext}_{\mathsf{CCCS}}$ be the PPT extractor for the protocol $\mathit{\Pi}_{\mathsf{CCCS}}$. By rewinding the malicious prover $\mathcal{P}^*$ twice with different challenges ρ, ρ', $\mathsf{Ext}_{\mathsf{CCCS}}$ can compute a witness wit' satisfying: (1) $\mathsf{Commit}(\mathsf{pp}, \widetilde{\mathsf{wit}}') = C$ guaranteed by the binding property of commitment scheme and (2) the CCS relation guaranteed by the soundness of sum-check protocol [32] and Schwartz-Zippel lemma. By applying the union bound, we claim that the soundness error of $\mathit{\Pi}_{\mathsf{CCCS}}$ is at most $O(d \cdot \log m + t + \log n)/|\mathbb{F}|$.

3.3 Atomic CCS Relations

In this part, we first introduce a relaxed CCS relation called *atomic CCS* that is amenable to constructing folding schemes for multiple instances with different structures. Different from the committed CCS relations or linearized committed CCS in [9], this new variant is satisfied with linear constraints on matrices and context-witness pairs, respectively. Therefore, folding multiple atomic CCS instances under different matrices does not produce any cross terms.

Definition 12 (Atomic CCS). *Let* $\mathsf{PC} = (\mathsf{Gen}, \mathsf{Commit}, \mathsf{Open}, \mathsf{Eval})$ *denote an additively-homomorphic polynomial commitment scheme for multilinear polynomials over a finite field* $\mathbb{F}$. *Denote the public parameters of size bounds as* $m, n, N, l, t \in \mathbb{N}$ *where* $n = 2 \cdot (l + 1)$ *and* $\mathsf{pp} \leftarrow \mathsf{Gen}(1^\lambda, s_y - 1)$. *The atomic customizable constraint system (ACCS) relation* $\mathcal{R}_{\mathsf{ACCS}}$ *is defined as follows.*

- *An* $\mathcal{R}_{\mathsf{ACCS}}$ *structure* $\mathcal{S}$ *consists of a sequence of sparse multilinear polynomials in* $s_x + s_y$ *variables* $\{\widetilde{M}_j\}_{j\in[t]}$ *such that they evaluate to a non-zero value in at most* $N = \mathit{\Omega}(m)$ *locations over the Boolean hypercube* $\{0,1\}^{s_x} \times \{0,1\}^{s_y}$.

- *An $\mathcal{R}_{\mathsf{ACCS}}$ context is $(C, v_0, \mathsf{io}, \boldsymbol{r}_x, \boldsymbol{r}_y, v_1, ..., v_t, v_z)$ where $v_0 \in \mathbb{F}, \mathsf{io} \in \mathbb{F}^l, \boldsymbol{r}_x \in \mathbb{F}^{s_x}, \boldsymbol{r}_y \in \mathbb{F}^{s_y}, v_z \in \mathbb{F}, v_j \in \mathbb{F}$ for all $j \in [t]$, and C is a commitment to a multilinear polynomial in $s_y - 1$ variables.*
- *An $\mathcal{R}_{\mathsf{ACCS}}$ witness consists of a multilinear polynomial $\widetilde{\mathsf{wit}}$ in $s_y - 1$ variables.*

An $\mathcal{R}_{\mathsf{ACCS}}$ instance (structure-context tuple) is satisfied by an $\mathcal{R}_{\mathsf{ACCS}}$ witness if $\mathsf{Commit}(\mathsf{pp}, \widetilde{\mathsf{wit}}) = C$, $v_z = \tilde{z}(\boldsymbol{r}_y)$ and if for all $j \in [t]$, the equation $v_j = \widetilde{M_j}(\boldsymbol{r}_x, \boldsymbol{r}_y)$ holds, where $\widetilde{M_j}(\boldsymbol{x}, \boldsymbol{y})$ is an $(s_x + s_y)$-variate multilinear polynomial, $\tilde{z}(\boldsymbol{y})$ is an s_y-variate multilinear polynomial such that $\tilde{z}(\boldsymbol{y}) = \widetilde{(\mathsf{wit}, v_0, \mathsf{io})}$ for all $\boldsymbol{y} \in \{0,1\}^{s_y}$.

The special sound protocol Π_{CCCS} in Section 3.2 proves the validity of committed CCS relations based on sum-check protocols. Moreover, this primitive also provides an approach to rewrite any committed CCS instance into the atomic CCS instance with this primitive. Nevertheless, we are not ready to build a folding scheme because it is infeasible to fold multiple atomic instances with contexts of different random vectors $\boldsymbol{r}_x, \boldsymbol{r}_y$. To amend this, we further devise another special sound protocol for atomic committed relations. Our idea is motivated by HyperNova [9], which runs a sum-check protocol to substitute the random vectors of linearized committed CCS instances for new ones. We illustrate this with a simple example: assuming a claim (constraint) $\widetilde{f}(\boldsymbol{r}_x) = v$, the prover writes a new polynomial as $\widetilde{g}(\boldsymbol{x}) = \widetilde{eq}(\boldsymbol{r}_x, \boldsymbol{x}) \cdot \widetilde{f}(\boldsymbol{x})$, and engages in a sum-check protocol with the verifier to show $\sum_{\boldsymbol{x} \in \{0,1\}^{s_x}} \widetilde{g}(\boldsymbol{x}) = v$ with randomness $\boldsymbol{r}'_x$. This equation holds because the sum of $\widetilde{g}(\boldsymbol{x})$ can be regarded as an MLE of $\widetilde{f}(\cdot)$ as $\widetilde{f}(\boldsymbol{e}) = \sum_{\boldsymbol{x} \in \{0,1\}^{s_x}} \widetilde{eq}(\boldsymbol{e}, \boldsymbol{x}) \cdot \widetilde{f}(\boldsymbol{x})$ according to Lemma 1. By evaluating $\widetilde{f}(\boldsymbol{e})$ on $\boldsymbol{e} = \boldsymbol{r}_x$, we obtain

$$v = \widetilde{f}(\boldsymbol{r}_x) = \sum_{\boldsymbol{x} \in \{0,1\}^{s_x}} \widetilde{eq}(\boldsymbol{r}_x, \boldsymbol{x}) \cdot \widetilde{f}(\boldsymbol{x}) = \sum_{\boldsymbol{x} \in \{0,1\}^{s_x}} \widetilde{g}(\boldsymbol{x}).$$

As a result, the prover produces a new claim as $\widetilde{g}(\boldsymbol{r}'_y) = v'$. By checking that $v' = e \cdot v$, where $e = \widetilde{eq}(\boldsymbol{r}_y, \boldsymbol{r}'_y)$, the validity of the original claim can be guaranteed by the soundness of the sum-check protocol.

Based on the observations above, we build a special sound protocol Π_{ACCS} friendly for the folding schemes. The prover and verifier run a series of interactions in the protocol to substitute the random vectors $\boldsymbol{r}_x, \boldsymbol{r}_y$ in the atomic CCS instance. First, the prover rewrites each $\widetilde{M_j}(\boldsymbol{r}_x, \boldsymbol{r}_y), j \in [t]$ as

$$M_j(\boldsymbol{x}) = \widetilde{eq}(\boldsymbol{r}_x, \boldsymbol{x}) \cdot \left(\sum_{\boldsymbol{y} \in \{0,1\}^{s_y}} \widetilde{eq}(\boldsymbol{r}_y, \boldsymbol{y}) \cdot \widetilde{M_j}(\boldsymbol{x}, \boldsymbol{y}) \right)$$

Then the prover and verifier run the first sum-check protocol on $\boldsymbol{x}$ for each $M_j(\boldsymbol{x})$. With the random challenge γ sampled by the verifier at step 1, the prover constructs an aggregated polynomial $f(\boldsymbol{x})$ as the linear combinations of each $M_j(\boldsymbol{x})$. Then the prover and verifier run the sum-check#1 on $f(\boldsymbol{x})$ at step

4 with the random vector $\boldsymbol{r}'_x$, where the sum_x equals to $\sum_{j\in[t]} \gamma^j \cdot v_j$. If the sum-check#1 is correctly executed, the claims on matrices are updated to

$$\sigma_j = \sum_{\boldsymbol{y}\in\{0,1\}^{s_y}} \widetilde{eq}(\boldsymbol{r}_y, \boldsymbol{y}) \cdot \widetilde{M}_j(\boldsymbol{r}'_x, \boldsymbol{y})$$

for $j \in [t]$ with the new random vector $\boldsymbol{r}'_x$ at step 5.

Special Sound Protocol $\mathit{\Pi}_{\mathsf{ACCS}} = (\mathcal{P}, \mathcal{V})$ for relation $\mathcal{R}_{\mathsf{ACCS}}$

1. $\mathcal{V}$: Sample $\gamma \gets\!\!\$\ \mathbb{F}$, and send to $\mathcal{P}$.
2. $\mathcal{V}$: Sample $\boldsymbol{r}'_x \gets\!\!\$\ \mathbb{F}^{s_x}$.
3. $\mathcal{P}$: Compute $\tilde{z}(\boldsymbol{y}) = (\widetilde{\mathsf{wit}}, 1, \mathsf{io})$.
4. **Sum-check#1.** $c_x \leftarrow \mathit{\Pi}_{\mathsf{sc}}(f, s_x, 2, \mathrm{sum}_x)$ with random $\boldsymbol{r}'_x$ where:
$$f(\boldsymbol{x}) = \textstyle\sum_{j\in[t]} \gamma^j \cdot \widetilde{eq}(\boldsymbol{r}_x, \boldsymbol{x}) \cdot \Big(\sum_{\boldsymbol{y}\in\{0,1\}^{s_y}} \widetilde{eq}(\boldsymbol{r}_y, \boldsymbol{y}) \cdot \widetilde{M}_j(\boldsymbol{x}, \boldsymbol{y})\Big).$$
5. $\mathcal{P}$: Compute $\{\sigma_j\}_{j\in[t]}$ and send to $\mathcal{V}$, where:
$$\sigma_j = \textstyle\sum_{\boldsymbol{y}\in\{0,1\}^{s_y}} \widetilde{eq}(\boldsymbol{r}_y, \boldsymbol{y}) \cdot \widetilde{M}_j(\boldsymbol{r}'_x, \boldsymbol{y}), \text{ for all } j \in [t].$$
6. $\mathcal{V}$: Compute $e_1 \leftarrow \widetilde{eq}(\boldsymbol{r}_x, \boldsymbol{r}'_x)$, and abort if:
$$c_x \neq e_1 \cdot \textstyle\sum_{j\in[t]} \gamma^j \cdot \sigma_j.$$
7. $\mathcal{V}$: Sample $\delta \gets\!\!\$\ \mathbb{F}$, and send to $\mathcal{P}$.
8. $\mathcal{V}$: Sample $\boldsymbol{r}'_y \gets\!\!\$\ \mathbb{F}^{s_y}$.
9. **Sum-check#2.** $c_y \leftarrow \mathit{\Pi}_{\mathsf{sc}}(g, s_y, 2, \mathrm{sum}_y)$ with random $\boldsymbol{r}'_y$ where:
$$g(\boldsymbol{y}) = \textstyle\sum_{j\in[t]} \delta^j \cdot \widetilde{eq}(\boldsymbol{r}_y, \boldsymbol{y}) \cdot \widetilde{M}_j(\boldsymbol{r}'_x, \boldsymbol{y}) + \delta^{t+1} \cdot \widetilde{eq}(\boldsymbol{r}_y, \boldsymbol{y}) \cdot \tilde{z}(\boldsymbol{y}).$$
10. $\mathcal{P}$: Compute $\epsilon, \{\theta_j\}_{j\in[t]}$ and send to $\mathcal{V}$, where for all $j \in [t]$:
$$\epsilon = \tilde{z}(\boldsymbol{r}'_y),\ \theta_j = \widetilde{M}_j(\boldsymbol{r}'_x, \boldsymbol{r}'_y).$$
11. $\mathcal{V}$: Compute $e_2 \leftarrow \widetilde{eq}(\boldsymbol{r}_y, \boldsymbol{r}'_y)$, and abort if:
$$c_y \neq e_2 \cdot (\textstyle\sum_{j\in[t]} \delta^j \cdot \theta_j + \delta^{t+1} \cdot \epsilon).$$
12. $\mathcal{P}$: Open the witness $\widetilde{\mathsf{wit}}$.
13. $\mathcal{V}$: Check that
$$(1)\ \mathsf{Commit}(\mathsf{pp}, \widetilde{\mathsf{wit}}) = C \qquad ,$$
$$(2)\ \epsilon = \tilde{z}(\boldsymbol{r}'_y),\ \theta_j = \widetilde{M}_j(\boldsymbol{r}'_x, \boldsymbol{r}'_y).$$

Next, the prover rewrites for $\widetilde{z}(\boldsymbol{r}_y)$ and each $\widetilde{M}_j(\boldsymbol{r}_x, \boldsymbol{r}_y)$ as

$$\begin{aligned} M_j(\boldsymbol{y}) &= \widetilde{eq}(\boldsymbol{r}_y, \boldsymbol{y}) \cdot \widetilde{M}_j(\boldsymbol{r}'_x, \boldsymbol{y}) \\ z(\boldsymbol{y}) &= \widetilde{eq}(\boldsymbol{r}_y, \boldsymbol{y}) \cdot \widetilde{z}(\boldsymbol{y}) \end{aligned}$$

the prover and verifier run the sum-check#2 on the aggregated $g(\boldsymbol{y})$ at step 9 with the random vector $\boldsymbol{r}'_y$, where $\mathrm{sum_y} = \sum_{j\in[t]} \delta^j \cdot \sigma_j + \delta^{t+1} \cdot v_z$. The remaining process runs similarly. Finally, the protocol transforms the original atomic CCS into a new one on $\boldsymbol{r}'_x, \boldsymbol{r}'_y$.

For the security of the proposed Π_{ACCS}, it retains completeness and soundness as the Π_{CCCS} does. We omit the security proof since it can be covered by the proofs for our generic folding scheme in Appendix A.

4 Generic Folding Scheme for CCS

4.1 High-level Ideas

This section describes a multi-folding scheme for committed CCS or atomic CCS instances with different structures, i.e., a generic folding scheme. Its aim is to fold the input instances into one atomic CCS instance. For better understanding, one can first imagine running the special sound protocol for each instance independently and then applying aggregation techniques for their intermediate steps, e.g., sum-check protocols and polynomial evaluations.

We use Figure 2 to further illustrate our idea. Given polynomials $\{f^{(i)}(\boldsymbol{x})\}_{i=1}^n$ derived from the input committed CCS or atomic CCS instances, the prover can aggregate them into one polynomial $f(\boldsymbol{x})$ by a challenge value γ given by the verifier. Then they can run the first sum-check protocol for $f(\boldsymbol{x})$ on the random vector $\boldsymbol{r}_x$. By fixing the input $\boldsymbol{x}$ as value $\boldsymbol{r}_x$, we further derive polynomials $\{g^{(i)}(\boldsymbol{y})\}_{i=1}^n$. Then, the prover and verifier run the same aggregation with challenge δ and the sum-check protocol on $\boldsymbol{r}_y$. Finally, the prover sends the claim to the evaluations of polynomials $\{\widetilde{M}_j^{(i)}\}_{j\in[t]}$ and $\tilde{z}^{(i)}$ for each i-th instance. A folding operation is executed on all the claims above to obtain a folded atomic CCS instance.

To build such a folding scheme, our starting point is to consider the simplest case of folding two instances. Practically, there are three combinations of input instances to be folded:

- two committed CCS instances;
- one committed CCS instance and one atomic CCS instance;
- two atomic CCS instances.

Note that if two instances share the same relation form (case 1 or 3), their special sound protocols are the same, which implies a straightforward aggregation for sum-check protocols and yields a folding scheme naturally. Therefore, we only need to focus on the second case.

In particular, we provide a folding scheme for two instances in specific relations $\mathcal{R}$ and $\mathcal{R}'$, where $\mathcal{R}$ and $\mathcal{R}'$ are $\mathcal{R}_{\mathsf{ACCS}}$ and $\mathcal{R}_{\mathsf{CCCS}}$ relations with different structures $\mathcal{S}$ and $\mathcal{S}'$ respectively. Specifically, $\mathcal{S}$ and $\mathcal{S}'$ share the same size bounds m, n, N, l, t, but contain different multilinear polynomials, $\{\widetilde{M}_j\}_{j\in[t]}$ and $\{\widetilde{M}'_j\}_{j\in[t]}$, respectively. The folding scheme for this trivial case is presented

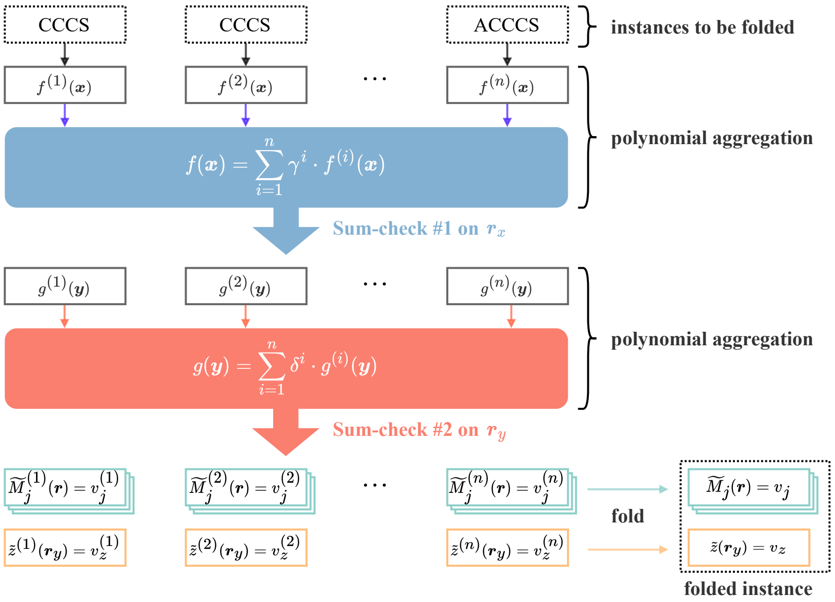
CCCS
CCCS
ACCCS
instances to be folded
$f^{(1)}(\boldsymbol{x})$
$f^{(2)}(\boldsymbol{x})$
$\cdots$
$f^{(n)}(\boldsymbol{x})$
polynomial aggregation
$f(\boldsymbol{x}) = \sum_{i=1}^{n} \gamma^i \cdot f^{(i)}(\boldsymbol{x})$
Sum-check #1 on $\boldsymbol{r}_x$
$g^{(1)}(\boldsymbol{y})$
$g^{(2)}(\boldsymbol{y})$
$\cdots$
$g^{(n)}(\boldsymbol{y})$
polynomial aggregation
$g(\boldsymbol{y}) = \sum_{i=1}^{n} \delta^i \cdot g^{(i)}(\boldsymbol{y})$
Sum-check #2 on $\boldsymbol{r}_y$
$\widetilde{M}_j^{(1)}(\boldsymbol{r}) = v_j^{(1)}$
$\widetilde{M}_j^{(2)}(\boldsymbol{r}) = v_j^{(2)}$
$\cdots$
$\widetilde{M}_j^{(n)}(\boldsymbol{r}) = v_j^{(n)}$
$\widetilde{M}_j(\boldsymbol{r}) = v_j$
fold
$\tilde{z}^{(1)}(\boldsymbol{r}_y) = v_z^{(1)}$
$\tilde{z}^{(2)}(\boldsymbol{r}_y) = v_z^{(2)}$
$\tilde{z}^{(n)}(\boldsymbol{r}_y) = v_z^{(n)}$
$\tilde{z}(\boldsymbol{r}_y) = v_z$
folded instance

Fig. 2: Generic Folding Scheme

in the Section 4.2, which satisfies completeness and knowledge soundness. We further introduce how to realize zero-knowledge in Section 4.3.

Besides, it is straightforward to adopt this folding scheme in other cases. To build a generic folding scheme for multiple instances, the prover and verifier run the same process for each instance at first and fold all claims at the final steps. The atomic CCS structure allows us to fold multiple instances without cross terms since $\mathcal{R}_{\mathsf{ACCS}}$ has linear constraints on matrices and context-witness pairs. For structures with different size bounds, we can use a simple padding scheme to ensure the same size. Since all these techniques are quite straightforward, we omit the detailed description of the generic folding for simplicity. Instead, we present a non-interactive generic folding scheme with zero knowledge in Section 4.4 to put everything together.

4.2 Main Protocol

Construction 1 (Folding scheme for committed CCS)**.** Let $\mathsf{PC} = (\mathsf{Gen}, \mathsf{Commit}, \mathsf{Open}, \mathsf{Eval})$ denote an additively homomorphic polynomial commitment scheme for multilinear polynomials. The generator and the encoder are defined as follows.

```
$\mathcal{G}(1^\lambda \to \mathsf{pp})$ :
Sample size bounds $m, n, N, l, t, q, d \in \mathbb{N}$ with $n = 2 \cdot (l + 1)$.
$\mathsf{pp}_{\mathsf{PC}} \leftarrow \mathsf{Gen}(1^\lambda, \log n - 1)$.
Output $(m, n, N, l, t, q, d, \mathsf{pp}_{\mathsf{PC}})$.
```

$\mathcal{K}(\mathsf{pp}, \mathcal{S}, \mathcal{S}') \to (\mathsf{pk}, \mathsf{vk})$:

```
Parse $\mathcal{S}$ to obtain $\{\widetilde{M}_j\}_{j \in [t]}$.
Parse $\mathcal{S}'$ to obtain $\{\widetilde{M}'_j\}_{j \in [t]}, \{S'_i\}_{i \in [q]}, \{c'_i\}_{i \in [q]}$.
$\mathsf{pk} \leftarrow (\mathsf{pp}, (\{\widetilde{M}_j\}_{j \in [t]}, \{\widetilde{M}'_j\}_{j \in [t]}, \{S'_i\}_{i \in [q]}, \{c'_i\}_{i \in [q]}))$.
$\mathsf{vk} \leftarrow \perp$.
Output $(\mathsf{pk}, \mathsf{vk})$.
```

To distinguish, we mark the parts corresponding to the committed CCS instance in blue text. The verifier $\mathcal{V}$ takes as inputs an atomic CCS context $(C, v_0, \mathsf{io}, \boldsymbol{r}_x, \boldsymbol{r}_y, \{v_j\}_{j \in [t]}, v_z)$ and a committed CCS context (C', io'). The prover $\mathcal{P}$, in addition to the two contexts, takes witnesses $\widetilde{\mathsf{wit}}$ and $\widetilde{\mathsf{wit}}'$. Let $s_x = \log m$, $s_y = \log n$, $\tilde{z} = (\widetilde{\mathsf{wit}}, v_0, \mathsf{io})$, and $\tilde{z}' = (\widetilde{\mathsf{wit}}', 1, \mathsf{io}')$. The prover and the verifier proceed as follows.

1. $\mathcal{V} \to \mathcal{P}$: $\mathcal{V}$ samples $\gamma \leftarrow\!\!\$\ \mathbb{F}$, $\boldsymbol{\alpha} \leftarrow\!\!\$\ \mathbb{F}^{s_x}$, and sends them to $\mathcal{P}$.
2. $\mathcal{V}$: Sample $\boldsymbol{r}'_x \leftarrow\!\!\$\ \mathbb{F}^{s_x}$.
3. $\mathcal{P}$: Compute $\tilde{z}(\boldsymbol{y}) = (\widetilde{\mathsf{wit}}, v_0, \mathsf{io})$, $\tilde{z}'(\boldsymbol{y}) = (\widetilde{\mathsf{wit}}', 1, \mathsf{io}')$.
4. $\mathcal{V} \leftrightarrow \mathcal{P}$: Run the sum-check protocol#1 $c_x \leftarrow \mathit{\Pi}_{\mathrm{sc}}(f, s_x, d+1, \mathsf{sum}_x, \boldsymbol{r}'_x)$, where:

$$\mathsf{sum}_x := \sum_{j \in [t]} \gamma^j \cdot v_j,$$

$$f(\boldsymbol{x}) := \left(\sum_{j \in [t]} \gamma^j \cdot L_j(\boldsymbol{x}) \right) + \gamma^t \cdot Q(\boldsymbol{x}),$$

$$L_j(\boldsymbol{x}) := \widetilde{eq}(\boldsymbol{r}_x, \boldsymbol{x}) \cdot \left(\sum_{\boldsymbol{y} \in \{0,1\}^{s_y}} \widetilde{eq}(\boldsymbol{r}_y, \boldsymbol{y}) \cdot \widetilde{M}_j(\boldsymbol{x}, \boldsymbol{y}) \right), j \in [t],$$

$$Q(\boldsymbol{x}) := \widetilde{eq}(\boldsymbol{\alpha}, \boldsymbol{x}) \cdot \left(\sum_{i \in [q]} c'_i \cdot \prod_{j \in S_i} \left(\sum_{\boldsymbol{y} \in \{0,1\}^{s_y}} \widetilde{M}'_j(\boldsymbol{x}, \boldsymbol{y}) \cdot \tilde{z}'(\boldsymbol{y}) \right) \right).$$

5. $\mathcal{P} \to \mathcal{V}$: $(\{\sigma_j\}_{j \in [t]}, \{\sigma'_j\}_{j \in [t]})$, where:

$$\sigma_j = \sum_{\boldsymbol{y} \in \{0,1\}^{s_y}} \widetilde{eq}(\boldsymbol{r}_y, \boldsymbol{y}) \cdot \widetilde{M}_j(\boldsymbol{r}'_x, \boldsymbol{y}), \forall j \in [t],$$

$$\sigma'_j = \sum_{\boldsymbol{y} \in \{0,1\}^{s_y}} \widetilde{M}'_j(\boldsymbol{r}'_x, \boldsymbol{y}) \cdot \tilde{z}'(\boldsymbol{y}), \forall j \in [t].$$

6. $\mathcal{V}$: Compute $e_1 \leftarrow \widetilde{eq}(\boldsymbol{r}_x, \boldsymbol{r}'_x)$ and $e_2 \leftarrow \widetilde{eq}(\boldsymbol{\alpha}, \boldsymbol{r}'_x)$, and abort if:

$$c_x \neq \left(\sum_{j \in [t]} \gamma^j \cdot e_1 \cdot \sigma_j \right) + \left(\gamma^t \cdot e_2 \cdot \sum_{i \in [q]} c'_i \cdot \prod_{j \in S_i} \sigma'_j \right).$$

7. $\mathcal{V} \to \mathcal{P}$: $\mathcal{V}$ samples $\delta \leftarrow\!\!\$\ \mathbb{F}$, and sends it to $\mathcal{P}$.
8. $\mathcal{V}$: Sample $\boldsymbol{r}'_y \leftarrow\!\!\$\ \mathbb{F}^{s_y}$.
9. $\mathcal{V} \leftrightarrow \mathcal{P}$: Run the sum-check protocol#2 $c_y \leftarrow \mathit{\Pi}_{\mathrm{sc}}(g, s_y, 2, \mathsf{sum}_y, \boldsymbol{r}'_y)$, where:

$$\begin{aligned}
\mathsf{sum}_y &:= \sum_{j\in[t]} \delta^j \cdot \sigma_j + \delta^{t+1} \cdot v_z + \delta^{t+1} \cdot \sum_{j\in[t]} \delta^j \cdot \sigma'_j, \\
g(\boldsymbol{y}) &:= \sum_{j\in[t]} \delta^j \cdot R_j(\boldsymbol{y}) + \delta^{t+1} \cdot S(\boldsymbol{y}) + \delta^{t+1} \cdot \sum_{j\in[t]} \delta^j \cdot T_j(\boldsymbol{y}), \\
R_j(\boldsymbol{y}) &= \widetilde{eq}(\boldsymbol{r}_y, \boldsymbol{y}) \cdot \widetilde{M}_j(\boldsymbol{r}'_x, \boldsymbol{y}), \forall j \in [t], \\
S(\boldsymbol{y}) &= \widetilde{eq}(\boldsymbol{r}_y, \boldsymbol{y}) \cdot \tilde{z}(\boldsymbol{y}), \\
T_j(\boldsymbol{y}) &= \widetilde{M}'_j(\boldsymbol{r}'_x, \boldsymbol{y}) \cdot \tilde{z}'(\boldsymbol{y}), \forall j \in [t].
\end{aligned}$$

10. $\mathcal{P} \to \mathcal{V}$: $(\epsilon, \epsilon', \{\theta_j\}_{j\in[t]}, \{\theta'_j\}_{j\in[t]})$, where:

$$\begin{aligned}
\epsilon &= \tilde{z}(\boldsymbol{r}'_y), \\
\epsilon' &= \tilde{z}'(\boldsymbol{r}'_y), \\
\theta_j &= \widetilde{M}_j(\boldsymbol{r}'_x, \boldsymbol{r}'_y), \forall j \in [t], \\
\theta'_j &= \widetilde{M}'_j(\boldsymbol{r}'_x, \boldsymbol{r}'_y), \forall j \in [t].
\end{aligned}$$

11. $\mathcal{V} \to \mathcal{P}$: $\mathcal{V}$ samples $\eta \leftarrow\!\!\$\ \mathbb{F}$ and sends it to $\mathcal{P}$.
12. $\mathcal{V}$: Compute $e_3 \leftarrow \widetilde{eq}(\boldsymbol{r}_y, \boldsymbol{r}'_y)$, and abort if:

$$c_y \neq \sum_{j\in[t]} \delta^j \cdot e_3 \cdot \theta_j + \delta^{t+1} \cdot e_3 \cdot \epsilon + \delta^{t+1} \cdot \sum_{j\in[t]} \delta^j \cdot \theta'_j \cdot \epsilon'$$

13. $\mathcal{V}, \mathcal{P}$: Output the folded atomic CCS structure $\mathcal{S}^*$ containing

$$M^*_j = M_j + \eta \cdot M'_j$$

for all $j \in [t]$ and its context $(C^*, v^*_0, \mathsf{io}^*, \boldsymbol{r}^*_x, \boldsymbol{r}^*_y, \{v^*_j\}_{j\in[t]}, v^*_z)$, where $\boldsymbol{r}^*_x = \boldsymbol{r}'_x, \boldsymbol{r}^*_y = \boldsymbol{r}'_y$, and for all $j \in [t]$:

$$\begin{aligned}
C^* &\leftarrow C + \eta \cdot C', \\
v^*_0 &\leftarrow v_0 + \eta \cdot 1, \\
\mathsf{io}^* &\leftarrow \mathsf{io} + \eta \cdot \mathsf{io}', \\
v^*_j &\leftarrow \theta_j + \eta \cdot \theta'_j, \\
v^*_z &\leftarrow \epsilon + \eta \cdot \epsilon'.
\end{aligned}$$

14. $\mathcal{P}$: Output the folded witness $\mathsf{wit} + \eta \cdot \mathsf{wit}'$.

Theorem 2. *(Folding scheme for committed CCS). Construction 1 is a public coin folding scheme for $(\mathcal{R}, \mathcal{R}')$ with perfect completeness and knowledge soundness.*

The proof of Theorem 1 is given in Appendix A.

4.3 Adding zero-knowledge

The above construction is proven to satisfy completeness and knowledge soundness properties. In this part, we further discuss adding zero knowledge to it. A straightforward idea is directly adding a masking value for the witness wit. That is, the prover runs the protocol with $\rho \cdot \mathsf{wit} + \boldsymbol{w}$ instead of wit, where $\boldsymbol{w} \in \mathbb{F}^{n-l-1}$. Although this technique can prove the validity of folding without leaking any information about the witness, the prover needs to do the extra computation, especially when folding committed CCS instances. We illustrate this point with the following example. Assume the prover wants to run the sum-check#1 on a committed CCS instance with a masking vector $\boldsymbol{w}$. The target polynomial is written as:

$$f(\boldsymbol{x}) = \widetilde{eq}(\boldsymbol{\alpha}, \boldsymbol{x}) \cdot \left(\sum_{i \in [q]} c_i \cdot \prod_{j \in S_i} \left(\sum_{\boldsymbol{y} \in \{0,1\}^{s_y}} \widetilde{M}_j(\boldsymbol{x}, \boldsymbol{y}) \cdot \widetilde{Z}(\boldsymbol{y}) \right) \right),$$

where $\widetilde{Z}(\boldsymbol{y}) = (\rho \cdot \widetilde{\mathsf{wit} + \boldsymbol{w}}, 1, \mathsf{io})$. Since the masking vector $\boldsymbol{w}$ is sampled randomly, the above target polynomial no longer equals zero. To ensure the equation holds, the prover has to compute the sum of the new polynomial $f(\boldsymbol{x})$, which takes $O(N + tm + qmd \log^2 d)$ $\mathbb{F}$-ops dominated by the computation of non-linear part with degree d. This seems not a serious problem when the folding input includes only one committed CCS instance. However, when dealing with multiple committed CCS instances, the prover must execute the above computation for each independently.

To alleviate the prover's computation, we propose a more efficient approach for our scheme by separating the zero-knowledge problem into three parts and solving them independently.

(1) Zero-knowledge for prover claims. We first consider shielding the witness wit among the verification of the claims. The verifier is expected to learn no information about wit except its validity to the CCS instance from steps 5, 6, and 10-12. As mentioned above, the non-linearity part of the CCS relation prevents us from directly masking the witness as $\rho \cdot \mathsf{wit} + \boldsymbol{w}$. Here, we utilize an approach in [28] by Bootle et al. to randomize the claims, which will not introduce extra costs for non-linear parts. Generally speaking, the claims at steps 5 and 10 can be regarded as linear combinations of the values in wit. According to the result given by Bootle et al., if we pad the witness with as many non-zero random values as the number of combinations it receives, then all the responses will be uniformly random and leak no information.

Again, we take the committed CCS relation in Equation (2) as an example. With a randomly sampled vector $\boldsymbol{r} = [\boldsymbol{r}_j]_{j=1}^{t}, \boldsymbol{r}_j \in \mathbb{F}^2$ added to the vector $\boldsymbol{r}$, the equation can be rewritten as

$$\sum_{i \in [q]} c_i \cdot \bigcirc_{j \in S_i} \begin{bmatrix} M_j & O \\ O & I_j \end{bmatrix} \cdot (\boldsymbol{z}, \boldsymbol{r}) = (\boldsymbol{0}, \sum_{i \in [q]} c_i \cdot \bigcirc_{j \in S_i} \boldsymbol{r}_j),$$

where O denotes zero matrix, I_j is a $2 \times 2t$ matrix with an 2×2 identity matrix I in the j-th position, i.e.,

$$I_j = [\underbrace{O, ..., O}_{j-1}, I, O, ..., O].$$

So far, the zero knowledge of the committed CCS instance is retained against $2t$ queries of the linear combination of wit. To accommodate it to our folding scheme, we denote extra two sets of variables as $\boldsymbol{a} \in \{0,1\}, \boldsymbol{b} \in \{0,1\}^{\log(2t)}$ for representing $I_j, \boldsymbol{r}_j$. The above equation can be further written into the multilinear polynomial form as follows [3]:

$$\sum_{i \in [q]} c_i \prod_{j \in S_i} \left(\sum_{\boldsymbol{y} \in \{0,1\}^{s_y}, \boldsymbol{b} \in \{0,1\}^{\log(2t)}} (\widetilde{M}_j(\boldsymbol{x}, \boldsymbol{y}) + \widetilde{I}_j(\boldsymbol{a}, \boldsymbol{b})) \cdot (\widetilde{z}(\boldsymbol{y}) + \widetilde{r}(\boldsymbol{b})) \right)$$
$$= \sum_{\boldsymbol{b} \in \{0,1\}^{\log(2t)}} \left(\sum_{i \in [q]} c_i \cdot \bigcirc_{j \in S_i} \widetilde{r}_j(\boldsymbol{b}) \right),$$

for all $\boldsymbol{x} \in \{0,1\}^{s_x}, \boldsymbol{a} \in \{0,1\}$, where $\widetilde{I}_j(\boldsymbol{a}, \boldsymbol{b})), \widetilde{r}_j(\boldsymbol{b})$ are the MLE's for I_j, r_j on $\boldsymbol{a}, \boldsymbol{b}$ and $\widetilde{r}(\boldsymbol{b}) = \sum_{j=1}^{t} \widetilde{r}_j(\boldsymbol{b})$.

(2) Zero-knowledge for sum-check protocols. Preserving the privacy of wit among the claims sent by the prover is insufficient to ensure zero knowledge of the whole folding scheme. Note that the sum-check protocols#1 and #2 at steps 4 and 9 are not shielded. Thus, the transcripts in the protocol also leak the information of wit. To amend this problem, we refer to a previous work by Chiesa [27], which presents a zero-knowledge sum-check protocol to mask the coefficients of the target polynomial with a random polynomial of the same size. Briefly, we can first sample a random vector $f_r(\boldsymbol{x})$ with the same variables and individual degrees of $f(\boldsymbol{x})$, and run the sum-check protocol with $\rho \cdot f(\boldsymbol{x}) + f_r(\boldsymbol{x})$ accordingly, where $\rho \in \mathbb{F}$ is a challenge. The following equation holds for sum-check#1 at step 4,

$$\rho \cdot \mathsf{sum}_x + \Delta\mathsf{sum}_x = \sum_{\boldsymbol{x} \in \{0,1\}^{s_x+2}} (\rho \cdot f(\boldsymbol{x}) + f_r(\boldsymbol{x})),$$

where $\Delta\mathsf{sum}_x$ is computed as $\sum_{\boldsymbol{x} \in \{0,1\}^{s_x+2}} f_r(\boldsymbol{x})$ and sent to the verifier at the beginning. For simplicity, we denote the zero-knowledge sum-check protocol as $c \leftarrow \Pi_{\mathsf{zksc}}(f, n, d, \mathsf{sum}, \boldsymbol{r})$. The security is guaranteed accordingly by the results in [27].

[3] It is more efficient to write polynomials on $\boldsymbol{x} \in \{0,1\}^{\log(m+2)}, \boldsymbol{y} \in \{0,1\}^{\log(n+2t)}$, we omit this expression for simplicity. In fact, the extra variables may even be unnecessary in real-world implementation since most vector $\boldsymbol{z}$ ends with a number of zeros, which can be replaced with randomness.

(3) Zero-knowledge for folded instance. The previous two techniques can guarantee zero knowledge of most processes in Construction 1 except the final folding operation at step 14. The prover outputs the folded witness $\mathsf{wit} + \eta \cdot \mathsf{wit}'$ without randomization. To amend this, the masking value $\boldsymbol{w} \in \mathbb{F}^{n-l-1}$ mentioned at the beginning has to be introduced. Differently, only one $\boldsymbol{w}$ is needed this time because the privacy in the previous steps is already preserved. Therefore, the prover takes the masking value as another committed CCS instance, i.e., masking instance, with empty structure $\mathcal{S}$, empty io and commitment $C = \mathsf{Commit}(\mathsf{pp}, \widetilde{\boldsymbol{w}})$, and runs the folding scheme accordingly, where O denote $m \times n$ zero matrix.

Besides, we also want to mention a trick for saving the computation of the sum of $\widetilde{\boldsymbol{w}}$. The polynomial $f(\boldsymbol{x})$ aggregates the masking value with all other instances. According to the proving algorithm for the sum-check protocol proposed in [36], it is handy to acquire the sum of $f(\boldsymbol{x})$ from the bookkeeping table. Thus, we can obtain the sum of $\widetilde{\boldsymbol{w}}$ by subtracting sums of other instances, i.e., $\sum_{j\in[t]} \gamma^j \cdot v_j \cdot v_z$ for atomic CCS instance and 0 for committed CCS instance, from $\sum_{\boldsymbol{x}\in\{0,1\}^{s_x}}$ without actually computing the concrete evaluations.

Due to page limitations, we do not present the complete zero-knowledge version scheme. The corresponding security proof of the zero-knowledge version of construction 1 is given in Appendix A Moreover, the techniques mentioned above will be applied in the non-interactive version in the following subsection.

4.4 Putting Everything Together

Applying the Fiat-Shamir transformation to the above protocol makes obtaining a non-interactive generic folding scheme in the random oracle model feasible while preserving zero knowledge. In the construction below, we present a non-interactive generic folding scheme with input as multiple committed CCS or atomic CCS instances. Zero knowledge is achieved as well by applying the techniques mentioned above. Guaranteed by the security of construction 1, it is not difficult to argue that construction 2 also satisfies completeness, knowledge soundness, and zero-knowledge. At the end of this part, we give a comprehensive evaluation of the performance, including the prover cost, verifier cost, and communication complexity.

Construction 2 (Zero-knowledge non-interactive generic folding scheme)**.** We construct a zero-knowledge non-interactive generic folding scheme as $\mathsf{zk\text{-}NIFS}$, which consists of 4 PPT algorithms $(\mathcal{G}, \mathcal{K}, \mathcal{P}, \mathcal{V})$. Let $\mathbb{H}$ be the random oracle, $\mathsf{PC} = (\mathsf{Gen}, \mathsf{Commit}, \mathsf{Open}, \mathsf{Eval})$ denote an additively-homomorphic polynomial commitment scheme for multilinear polynomials. For generality, we assume the scheme takes input as multiple CCS instances, including

- $\sum_{i\in[\ell_1]} s^{(i)}$ atomic CCS instances in a set of relations $\{\mathcal{R}^{(i)}\}_{i\in[\ell_1]}$ with different structures $\{\mathcal{S}^{(i)}\}_{i\in[\ell_1]}$, where each relation $\mathcal{R}^{(i)}$ corresponds to $s^{(i)}$ instances. Let $s_1 = \sum_{i\in[\ell_1]} s^{(i)}$, each atomic CCS instance consists of $(S^{(k)}, \mathsf{ctx}^{(k)}, \mathsf{wit}^{(k)})$, where $\mathsf{ctx}^{(k)} = (C^{(k)}, v_0^{(k)}, \mathsf{io}^{(k)}, \boldsymbol{r}_x^{(k)}, \boldsymbol{r}_y^{(k)}, \{v_j^{(k)}\}_{j\in[t]}, v_z^{(k)})$ for all $k = 1, ..., s_1$.

- $\sum_{i=\ell_1+1}^{\ell_1+\ell_2} s^{(i)}$ committed CCS instances in a set of relations $\{\mathcal{R}^{(i)}\}_{i=\ell_1+1}^{\ell_1+\ell_2}$ with different structures $\{\mathcal{S}^{(i)}\}_{i=\ell_1+1}^{\ell_1+\ell_2}$, where each relation $\mathcal{R}^{(i)}$ corresponds to $s^{(i)}$ instances. Let $s_2 = \sum_{i=\ell_1+1}^{\ell_1+\ell_2} s^{(i)}$, each committed CCS instance is consists of $(S^{(k)}, \mathsf{ctx}^{(k)}, \mathsf{wit}^{(k)})$, where $\mathsf{ctx}^{(k)} = (C^{(k)}, \mathsf{io}^{(k)})$ for all $k = s_1 + 1, ..., s_1 + s_2$

Denote $s = s_1 + s_2$, we use i, k to index the relations (structures) $\{\mathcal{R}^{(i)}\}_{i\in[\ell]}$ and contexts $\{\mathsf{ctx}^{(k)}\}_{k\in[s]}$ respectively. By applying Fiat-Shamir transformation to the zero-knowledge sum-check protocol mentioned in Section 4.3, we obtain the proving algorithm $\mathrm{FS}[\Pi_{\mathsf{zksc}}].\mathcal{P}$ with output transcript as $\mathsf{ts} = (c, \Delta\mathsf{sum}, \{m_j\}_{j\in[s]}, \boldsymbol{r})$ and the verifying algorithm $\mathrm{FS}[\Pi_{\mathsf{zksc}}].\mathcal{V}$ with output 1 for validity.

We only present the concrete algorithms for $\mathsf{zk\text{-}NIFS}.\mathcal{P}, \mathsf{zk\text{-}NIFS}.\mathcal{V}$ below to highlight the differences compared to Section 4.2.

```
$\mathsf{zk\text{-}NIFS}.\mathcal{P}((\mathsf{pk}, \mathsf{vk}), \{S^{(i)}\}_{i\in[\ell]}, \{\mathsf{ctx}^{(k)}\}_{k\in[s]}, \{\mathsf{wit}^{(k)}\}_{k\in[s]})$ :
Randomly sample $\{\boldsymbol{r}^{(k)} \in \mathbb{F}^{2t}\}_{k\in[s]}, \boldsymbol{w} \in \mathbb{F}^{n-l-1}$.
Generate instance $\mathsf{ctx}^{(0)}$ with $\boldsymbol{w}$.
Pad each $\boldsymbol{z^{(k)}}$ with $\boldsymbol{r}^{(k)}$, update $\mathsf{sum}_x$.
Generate claims on sums of $\widetilde{\boldsymbol{w}}(\boldsymbol{y}), \{\widetilde{\boldsymbol{r}}^{(k)}(\boldsymbol{y})\}_{k=0}^{s}$.
Pad each $M_j^{(i)}$ with $I_j$.
$\gamma, \boldsymbol{\alpha} \leftarrow \mathbb{H}(\{\mathsf{ctx}^{(k)}\}_{k=0}^{s})$, construct polynomial $f$.
Run sum-check#1 as $\mathsf{ts}_x \leftarrow \mathrm{FS}[\Pi_{\mathsf{zksc}}].\mathcal{P}(f, s_x, d+1, \mathsf{sum}_x)$.
Generate claims on $\{\sigma_j^{(k)}\}_{k=0,j=1}^{s,t}$.
$\delta \leftarrow \mathbb{H}(\mathsf{ts}_x, \{\sigma_j^{(k)}\}_{k=0,j=1}^{s,t})$, construct polynomial $g$..
Run sum-check#2 as $\mathsf{ts}_y \leftarrow \mathrm{FS}[\Pi_{\mathsf{zksc}}].\mathcal{P}(g, s_y, d+1, \mathsf{sum}_y)$.
Generate claims on $\{\epsilon^{(k)}\}_{k=0}^{s}, \{\theta_j^{(k)}\}_{k=0,j=1}^{s,t}$.
$\eta \leftarrow \mathbb{H}(\mathsf{ts}_y, \{\epsilon^{(k)}\}_{k=0}^{s}, \{\theta_j^{(k)}\}_{k=0,j=1}^{s,t})$.
Set vectors for each instance $\mathsf{ctx}^{(k)}, k = 0, ..., s$ as
    $\mathbf{v}^{(k)} := (\{M_j^{(k)}\}_{j\in[t]}, C^{(k)}, v_0^{(k)}, \mathsf{io}^{(k)}, \{\theta_j^{(k)}\}_{j\in[t]}, \epsilon^{(k)}, \mathsf{wit}^{(k)})$.
$\mathbf{v}^* := \sum_{k=0}^{s} \eta^k \cdot \mathbf{v}^{(k)}$, $M_j^* := \sum_{i\in[\ell]} \eta^i \cdot M_j^{(i)}, \forall j \in [t]$.
Set folding proof as
    $\mathsf{pf} := (\mathsf{ctx}^{(0)}, \gamma, \boldsymbol{\alpha}, \boldsymbol{r}'_x, \mathsf{ts}_x, \{\sigma_j^{(k)}\}_{k=0,j=1}^{s,t}, \delta, \boldsymbol{r}'_y, \mathsf{ts}_y, \{\epsilon^{(k)}\}_{k=0}^{s}, \{\theta_j^{(k)}\}_{k=0,j=1}^{s,t})$.
Output $(\{M_j^*\}_{j\in[t]}, \mathbf{v}^*, \mathsf{pf})$.
```

```
$\mathsf{zk\text{-}NIFS}.\mathcal{V}((\mathsf{pk},\mathsf{vk}), \{S^{(i)}\}_{i\in[\ell]}, \{\mathsf{ctx}^{(k)}\}_{k\in[s]}, \mathsf{pf})$ :
1 :  Check the validity of $\mathsf{sum}_x$ with claims on $\widetilde{\boldsymbol{w}}(\boldsymbol{y}), \{\widetilde{\boldsymbol{r}}^{(k)}(\boldsymbol{y})\}_{k=0}^{s}$.
2 :  Pad each $M_j^{(i)}$ with $I_j$.
3 :  $\gamma, \boldsymbol{\alpha} \leftarrow \mathbb{H}(\{\mathsf{ctx}^{(k)}\}_{k=0}^{s})$.
4 :  Check sum-check#1 as $1 \overset{?}{=} \mathrm{FS}[\Pi_{\mathsf{zksc}}].\mathcal{V}(\mathsf{ts}_x, s_x, d+1, \mathsf{sum}_x)$.
5 :  Check claims on $\{\sigma_j^{(k)}\}_{k=0,j=1}^{s,t}$ with $c_x, \gamma, \boldsymbol{\alpha}$.
6 :  $\delta \leftarrow \mathbb{H}(\mathsf{ts}_x, \{\sigma_j^{(k)}\}_{k=0,j=1}^{s,t})$.
7 :  Check sum-check#2 as $1 \overset{?}{=} \mathrm{FS}[\Pi_{\mathsf{zksc}}].\mathcal{V}(\mathsf{ts}_y, s_y, d+1, \mathsf{sum}_y)$.
8 :  Check claims of $\{\epsilon^{(k)}\}_{k=0}^{s}, \{\theta_j^{(k)}\}_{k=0,j=1}^{s,t}$ with $c_y, \delta$.
9 :  $\eta \leftarrow \mathbb{H}(\mathsf{ts}_y, \{\epsilon^{(k)}\}_{k=0}^{s}, \{\theta_j^{(k)}\}_{k=0,j=1}^{s,t})$.
10 : Set vectors for each instance $\mathsf{ctx}^{(k)}, k = 0, ..., s$ as
         $$\mathbf{v}^{(k)} := (\{M_j^{(k)}\}_{j\in[t]}, C^{(k)}, v_0^{(k)}, \mathsf{io}^{(k)}, \{\theta_j^{(k)}\}_{j\in[t]}, \epsilon^{(k)}, \mathsf{wit}^{(k)}).$$
11 : Check $\mathbf{v}^* := \sum_{k=0}^{s} \eta^k \cdot \mathbf{v}^{(k)}$, $M_j^* := \sum_{i\in[\ell]} \eta^i \cdot M_j^{(i)}, \forall j \in [t]$.
```

Complexity. Denote the random oracle for sum-check protocol as $\mathbb{H}_{\mathsf{sc}}$.

The folding scheme prover

- asks $\log m + \log n$ queries to $\mathbb{H}_{\mathsf{sc}}$ and $\log m + 2$ queries to $\mathbb{H}$;
- computes sum-check protocols (steps 7,8,10,11 in $\mathsf{zk\text{-}NIFS}.\mathcal{P}$) with $O(s_1(N + tm) + s_2(N + tm + qmd \log^2 d))$ $\mathbb{F}$-ops for all s_1 atomic CCS instances and s_2 committed CCS instances where
  - for each atomic CCS instance, runs $O(N + tm)$ $\mathbb{F}$-ops according to the standard linear-time-sum-check techniques [36],
  - for each committed CCS instance, runs $O(N + tm + qmd \log^2 d)$ $\mathbb{F}$-ops according to the technique in SuperSpartan [26];
- performs s $\mathbb{G}$-ops to combine $\{C^{(k)}\}_{k=0}^{s}$;
- performs $O(\ell N + s(n+t))$ $\mathbb{F}$-ops to combine $\{M_j^{(i)}\}_{i,j=1}^{\ell,t}$ and field elements in $\{\mathbf{v}^{(k)}\}_{k=0}^{s}$.

The folding scheme verifier

- asks $\log m + \log n$ queries to $\mathbb{H}_{\mathsf{sc}}$ and $\log m + 2$ queries to $\mathbb{H}$;
- checks sum-check protocols (steps 4,5,7,8 in $\mathsf{zk\text{-}NIFS}.\mathcal{V}$) with $O(s_1(d \log m + \log n) + s_2(dq + d \log m + \log n))$ $\mathbb{F}$-ops for all s_1 atomic CCS instances and s_2 committed CCS instances;
- performs s $\mathbb{G}$-ops to combine $\{C^{(k)}\}_{k=0}^{s}$;
- performs $O(\ell N + s(n+t))$ $\mathbb{F}$-ops to combine $\{M_j^{(i)}\}_{i,j=1}^{\ell,t}$ and field elements in $\{\mathbf{v}^{(k)}\}_{k=0}^{s}$.

With an efficient technique for matrix aggregation introduced in the next section, the prover communication cost can be reduced. For input two instances, the prover time of NIFS is dominated by $O(s(N+tm+qmd\log^2 d))$ $\mathbb{F}$-ops, $O(s)$ $\mathbb{G}$-ops and $O(\log m + 2\log n)$ RO queries, the verification time is dominated by $O(s(dq+d\log m+\log n))$ $\mathbb{F}$-ops, $O(s)$ $\mathbb{G}$-ops and $O(\log m + 2\log n)$ RO queries. The communication complexity can be reduced from $O(d\log m + \log n + tN)$ to $O(d\log m + \log n + N)$.

5 KiloNova: Non-uniform zk-PCD Scheme

This section explains how to build a non-uniform zk-PCD scheme from the abovementioned generic folding scheme. To begin with, we discuss the optimization technique used to reduce the overhead for handling structure folds in the non-uniform PCD scheme in Section 5.1. Based on this technique, we further construct a zk-PCD scheme from the zero-knowledge non-interactive generic folding scheme in Section 5.2.

5.1 Optimization Techniques

First, we explain the complexity problem caused by structure folds and highlight the necessity of applying our optimization techniques. In real-world implementations, the folding scheme verifier does not directly compute the linear combination of matrices as described in step 11 of $\mathsf{zk\text{-}NIFS}.\mathcal{V}$. Instead, commitments on each matrix are used according to Protostar and Protogalaxy [7, 10], which incurs a large number of group scalar multiplications in the recursive circuits when handling multiple non-uniform instances.

Take our generic foldings scheme as an example, the verifier $\mathsf{zk\text{-}NIFS}.\mathcal{V}$ needs to compute the linear combination of matrix commitments $\{\mathsf{Commit}(M_j^{(i)})\}_{i\in[\ell]}$ for all $j = [t]$, leading to $O(\ell \cdot t)$ $\mathbb{G}$ operations in total. This raises complexity concerns for PCD system, especially when the prover needs to fold multiple instances with different structures among mutually distrustful nodes. On the one hand, the verification logic should be written into the recursive circuit, increasing the prover cost. On the other hand, the folded matrix commitments should be sent to the next node in PCD system, incurring a high communication cost. To alleviate this problem, we introduce two observations and propose the optimization techniques in the following.

Observation 1. We can reduce the size of the folded instance with a linear combination of matrix claims. Assume an output atomic CCS instance contains matrices $\{M_j^*\}_{j\in[t]}$ and corresponding claim values $\{v_j^*\}_{j=1}^t$ in the generic folding scheme. Essentially, instead of checking each claim independently, we can check one claim with random linear combinations of the matrices and values. Taking a randomly sampled challenge ζ, the prover computes $M^* = \sum_{j=1}^t \zeta^j \cdot M_j^*$ and

$v^* = \sum_{j=1}^{t} \zeta^j \cdot v_j^*$ checked by the following equation

$$\widetilde{M}^*(\boldsymbol{r}_x, \boldsymbol{r}_y) = \sum_{j=1}^{t} \zeta^j \cdot \widetilde{M}_j^*(\boldsymbol{r}_x, \boldsymbol{r}_y) = \sum_{j=1}^{t} \zeta^j \cdot v_j = v^*.$$

Therefore, the t matrices and values of an atomic CCS instance can be aggregated into one matrix and one value, respectively. The matrix number of the folded instance sent to the next node is reduced from t to 1.

The security of the folding scheme still holds. Here, we only give sketch proof of the knowledge soundness. Assume the original folding scheme $\mathsf{zk\text{-}NIFS}$ satisfies knowledge soundness. Thus, there exists an extractor Ext runs in polynomial time for $\mathsf{zk\text{-}NIFS}$, which succeeds in extracting witness with non-negligible probability ϵ. For the new aggregated folding scheme, an extractor Ext' can also be constructed by calling Ext. The extractor Ext' invokes Ext to obtain t transcripts, each with different challenge $\zeta^{(i)}, i = 1, .., t$. By interpolating, the extractor can compute the matrices $\{M_j^*\}_{j \in [t]}$ and values $\{v_j^*\}_{j=1}^{t}$ from the linear combined M^* and v^*. It is naive to argue that Ext' runs in polynomial time. For the advantage Ext' succeeds with probability $(\epsilon - \text{negl}(\lambda)) \cdot (1 - \text{negl}(\lambda))$. This is because given that Ext does not abort, the probability that different challenges are sampled with less than $\sqrt[d+1]{|\mathbb{F}|}$ rewinds, i.e., $\zeta^{(1)} \neq \cdots \neq \zeta^{(t)}$, is $(1 - O(1)/\sqrt[d+1]{|\mathbb{F}|}) \cdot \epsilon \cdot (1 - \sqrt[d+1]{|\mathbb{F}|}^d / |\mathbb{F}|)$.

Observation 2. The verification of structure folds can be decoupled from the generic folding scheme. In our folding scheme, the prover folds the structures by computing the linear combinations of public matrices, and the verifier repeats the same process. We observe that this subprotocol is independent of other steps in the folding scheme. Thus, the original prover can prove the structure folds by delegating to a third party (delegated prover) [4]. This is because the matrices are selected from a public list, e.g., an instruction set. Such delegation is extremely useful for optimizing the performance of PCD system because the delegated computation is an IVC executed by only one prover, which avoids communication between different nodes.

Although this result is kind of counter-intuitive, we argue that this modification does not contradict the security definitions. According to the knowledge soundness defined for the generic folding scheme in Section 3.1, the malicious prover is only required to output a valid folded instance of $(\mathcal{S}^*, \mathsf{ctx}^*, \mathsf{wit}^*)$ satisfying the atomic CCS relations. The extractor needs to check the correctness of the claims on M_j^*, z^* with v_j, v_z for $j \in [t]$, and then extract witnesses. Therefore, the knowledge soundness does not include the correctness of structure folds. In other words, if the extractor can extract witnesses $\mathsf{wit}^{(i)}$ satisfying the structure $S^{(i)}$ for all i, then there must be a folded structure S^{**} satisfied by the correctly folded witness wit^*. The consistency of S^{**} and S^* can be verified by anyone with access to public structures $S^{(i)}, i \in [t]$ and the challenge η for linear combinations.

[4] It is also practical to delegate this task directly to the final verifier (or decider) if it is not very frequent to run the verifier in the PCD systems.

As a result, we can safely remove the verification of structure folds from the original prover. This observation also applies to other non-uniform folding schemes such as Protostar and Protogalaxy [7,10]. Now, our task becomes constructing another efficient scheme for checking the structure folds. Our solution is to run another IVC system for this computation on the delegated prover. According to observation 1, the already linear combined matrices $\{M_j^*\}_{j=1}^t$ can be further aggregated with an extra linear combination as $M^* = \sum_{j=1}^t \zeta^j \cdot M_j^*$. Therefore, the main function of the candidate IVC system is iteratively folding matrices (or their commitments alternatively). Meanwhile, the IVC does not need to be zero-knowledge because the matrices are all public. We believe an IVC system instantiated from Nova [8] is sufficient for this task. One thing left is to ensure the delegated prover uses the same challenge η as the original prover. To achieve this, we can instantiate an accumulator with the binding property in both parties. A naive approach is computing an extra function as $z_i = \mathsf{Hash}(z_{i-1}, \eta_i)$ at each step. Other existing accumulators, such as [37], can also be utilized. Since these accumulators are also incremental computations, it is natural to add this computation in the original function F in the IVC system. Although this approach does not reduce the overall recursion overhead (including both provers) in a significant way, it can shift part of the computations from the nodes in PCD system to a powerful third party with more computational resources.

Technically, the extra IVC for structure folds runs parallelly with PCD system, taking challenges from the node and computing the linear combination of matrices accordingly. The function for each incremental computation is represented as the minimum operation as $z_{i+1} = F(z_i, \rho, M) = \rho \cdot z_i + M$, where ρ is the weight for linear combination (i.e., the challenge in PCD), M is the matrix to be folded. Next, we describe the concrete construction of PCD.

5.2 Construting zk-PCD from a generic folding scheme

This part constructs zk-PCD from our zero-knowledge non-interactive generic folding scheme. If zero knowledge is not considered, one can directly adapt the scheme in [21] to build a PCD from the folding scheme. However, a technical gap exists when we try to achieve zero-knowledge for PCD scheme. According to the conclusion in [14], a zk-PCD is built from an accumulation scheme (folding schemes in our paper) and an argument system (SNARK in [15] or NARK in [15]) for proving the recursive circuit, both of which are required to satisfy zero-knowledge. Unfortunately, in the construction of [21], PCD prover only computes and outputs a committed CCS instance satisfying the recursive circuit instead of proof from the argument system. Adding zero-knowledge to the committed CCS instance will introduce extra prover cost, as discussed in Section 4.3.

Therefore, we must find another efficient approach to realize the zk-PCD. The general idea is to redesign the original construction for PCD scheme in [21] by modifying the recursive circuit. To state the problem clearly, we describe the predicates represented by the recursive circuit as follows

1. Check that the compliance predicate $\varphi(z, z_{\mathsf{loc}}, z_1, ..., z_s)$ satisfies.
2. Check that the hash values for all input instances with non-empty $z_k, k \in [s]$ are valid.
3. Run the $\mathsf{zk\text{-}NIFS}.\mathcal{V}$ algorithm to check the validity of the folded instance.
4. Compute the hash value for the folded instance.

Note that predicates 3 and 4 will not leak information about $z_{\mathsf{loc}}, z_1, ..., z_s$ since the generic folding scheme already achieves zero knowledge. Thus, we only need to preserve the privacy of the witness $z_{\mathsf{loc}}, z_1, ..., z_s$ for predicates 1 and 2 (z is public). Thankfully, predicates 1 and 2 do not require the output of folding scheme $\mathsf{zk\text{-}NIFS}.\mathcal{P}$, which means that they can be checked before the prover runs $\mathsf{zk\text{-}NIFS}.\mathcal{P}$. We can split the circuit for the predicate into two parts as $\mathcal{R}_0, \mathcal{R}_1$ and handle them respectively in different steps. As a result, the prover first computes the instance $(\mathsf{ctx}_0, \mathsf{wit}_0)$ for $\mathcal{R}_0$, then folds it with other input instances. In the circuit $\mathcal{R}_1$, the validity of the folded instances CTX is checked. And the prover computes another instance for $\mathcal{R}_1$. The idea is illustrated in the figure below.

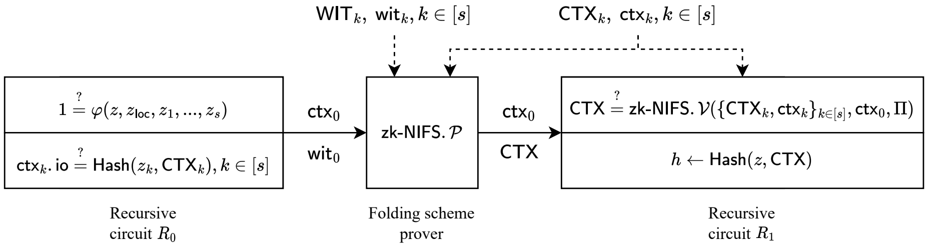
$\mathsf{WIT}_k,\ \mathsf{wit}_k, k \in [s]$
$\mathsf{CTX}_k,\ \mathsf{ctx}_k, k \in [s]$
$1 \overset{?}{=} \varphi(z, z_{\mathsf{loc}}, z_1, ..., z_s)$
$\mathsf{ctx}_k.\mathsf{io} \overset{?}{=} \mathsf{Hash}(z_k, \mathsf{CTX}_k), k \in [s]$
ctx_0
wit_0
$\mathsf{zk\text{-}NIFS}.\mathcal{P}$
ctx_0
CTX
$\mathsf{CTX} \overset{?}{=} \mathsf{zk\text{-}NIFS}.\mathcal{V}(\{\mathsf{CTX}_k, \mathsf{ctx}_k\}_{k\in[s]}, \mathsf{ctx}_0, \Pi)$
$h \leftarrow \mathsf{Hash}(z, \mathsf{CTX})$
Recursive circuit R_0
Folding scheme prover
Recursive circuit R_1

Fig. 3: Modified Recursive Cricuit for zk-PCD.

Compared to the original construction, PCD prover only needs to run the folding scheme with one more instance for $\mathcal{R}_0$, which adds negligible cost to its asymptotic complexity. We also show that this modification does not contradict the securities of PCD scheme in Appendix B. Moreover, the same modification can be applied to the construction of ZK-IVC as long as they are built from a multi-folding scheme.

Construction 3 (A PCD from Generic Folding Schemes)**.** Let $\mathsf{zk\text{-}NIFS}$ be the zero-knowledge non-interactive generic folding scheme for committed CCS and atomic CCS relations $\{\mathcal{R}^{(i)}\}_{i\in[\ell]}$. Let $(\mathsf{ctx}_\perp, \mathsf{wit}_\perp)$ be a default trivially satisfying atomic CCS context-witness pair for any structure and public parameters. According to the definition of PCD, we can construct a scheme consisting of polynomial-time algorithms $\mathsf{PCD} = (\mathcal{G}, \mathcal{K}, \mathcal{P}, \mathcal{V})$ for a class of compliance predicates F. Besides, we assume all the structures used below are valid, which is guaranteed by the extra IVC for proving the structure folds.

Denote a compliance predicate φ with a cryptographic hash function Hash, we first define the circuits R_0 and R_1 realizing the recursion on s inputs of $\{z_k, \mathsf{CTX}_k, \mathsf{ctx}_k\}_{k=1}^{s}$.

$0/1 \leftarrow R_0(h; (z, z_{\mathsf{loc}}, \{z_k, \mathsf{CTX}_k, \mathsf{ctx}_k\}_{k\in[s]}, \mathsf{vk}'))$:

```
Check that the compliance predicate $\varphi(z, z_{\mathsf{loc}}, z_1, ..., z_s)$ accepts.
For all $k \in [s]$ such that $z_k \neq \bot$, check that $\mathsf{ctx}_k.\mathsf{io} = \mathsf{Hash}(\mathsf{vk}', z_k, \mathsf{CTX}_k)$,
    where $\mathsf{ctx}_k.\mathsf{io}$ is the public IO of $\mathsf{ctx}_k$.
If the above checks hold, output 1; otherwise, output 0.
```

Since R_0 can be computed in polynomial time, it can be represented as a $\mathcal{R}_{\mathsf{CCCS}}$ structure s_0. Let

$$(\mathsf{s}_0, \mathsf{ctx}_0, \mathsf{wit}_0) \leftarrow \mathsf{trace}(R_0, (h, (z, z_{\mathsf{loc}}, \{z_k, \mathsf{CTX}_k, \mathsf{ctx}_k\}_{k\in[s]}, \mathsf{vk}'))$$

denote the satisfied $\mathcal{R}_{\mathsf{CCCS}}$ instance for the execution of the circuit R_0 on input $(h, (z, z_{\mathsf{loc}}, \{z_k, \mathsf{CTX}_k, \mathsf{ctx}_k\}_{k\in[s]}, \mathsf{vk}'))$.

$0/1 \leftarrow R_1(h; (\{\mathsf{CTX}_k, \mathsf{ctx}_k\}_{k\in[s]}, \mathsf{ctx}_0, \mathsf{vk}', \mathsf{CTX}, \mathit{\Pi}))$:

```
If $z_k = \bot$ for all $k \in [s]$, check that $h = \mathsf{Hash}(\mathsf{vk}', z, \mathsf{CTX})$ and $\mathsf{ctx}_0 = \mathsf{CTX}$, else check that
    (a) $\mathsf{CTX} = \mathsf{zk\text{-}NIFS}.\mathcal{V}'(\mathsf{vk}', \{\mathsf{CTX}_k, \mathsf{ctx}_k\}_{k\in[s]}, \mathsf{ctx}_0, \mathit{\Pi})$.
    (b) $h = \mathsf{Hash}(\mathsf{vk}', z, \mathsf{CTX})$.
If the above checks hold, output 1; otherwise, output 0.
```

Since R_1 can be computed in polynomial time, it can be represented as a $\mathcal{R}_{\mathsf{CCCS}}$ structure s. Let

$$(\mathsf{s}, \mathsf{ctx}, \mathsf{wit}) \leftarrow \mathsf{trace}(R_0, (h, (\{\mathsf{CTX}_k, \mathsf{ctx}_k\}_{k\in[s]}, \mathsf{ctx}_0, \mathsf{vk}', \mathsf{CTX}, \mathit{\Pi}))$$

denote the satisfied $\mathcal{R}_{\mathsf{CCCS}}$ instance for the execution of the circuit R_1 on input $(h, (\{\mathsf{CTX}_k, \mathsf{ctx}_k\}_{k\in[s]}, \mathsf{ctx}_0, \mathsf{vk}', \mathsf{CTX}, \mathit{\Pi}))$.

Now, we can define the algorithms $(\mathcal{G}, \mathcal{K}, \mathcal{P}, \mathcal{V})$ for PCD scheme.

$\mathsf{pp} \leftarrow \mathcal{G}(1^\lambda)$:

```
Compute and output $\mathsf{pp}' \leftarrow \mathsf{zk\text{-}NIFS}.\mathcal{G}(1^\lambda)$.
```

$(\mathsf{pk}, \mathsf{vk}) \leftarrow \mathcal{K}(\mathsf{pp}, \varphi)$:

```
Compute $(\mathsf{pk_{fs}}', \mathsf{vk_{fs}}') \leftarrow \mathsf{zk\text{-}NIFS}.\mathcal{K}'(\mathsf{pp}', R_\varphi)$.
Output $(\mathsf{pk}, \mathsf{vk}) \leftarrow ((\varphi, \mathsf{pk_{fs}}'), (\varphi, \mathsf{vk_{fs}}'))$.
```

```
$\mathit{\Pi} \leftarrow \mathcal{P}(\mathsf{pk}, z, z_{\mathsf{loc}}, \{z_k, \mathit{\Pi}_k\}_{k=1}^{s})$:
For $k \in [s]$, parse $\mathit{\Pi}_k$ as satisfied atomic CCS instance $(\mathsf{S}_k, \mathsf{CTX}_k, \mathsf{WIT}_k)$ and satisfied committed CCS instance $(\mathsf{s}_k, \mathsf{ctx}_k, \mathsf{wit}_k))$.
$(\mathsf{s}_0, \mathsf{ctx}_0, \mathsf{wit}_0) \leftarrow \mathsf{trace}(R_0, (h, (z, z_{\mathsf{loc}}, \{z_k, \mathsf{CTX}_k, \mathsf{ctx}_k\}_{k\in[s]}, \mathsf{vk}'))$
If $z_k = \perp$ for all $k \in [s]$, then set $(\mathsf{CTX}, \mathsf{WIT}, \mathsf{pf}) := (\mathsf{ctx}_0, \mathsf{wit}_0, \perp)$, else compute $(\mathsf{S}, \mathsf{CTX}, \mathsf{WIT}, \mathsf{pf}) \leftarrow \mathsf{zk\text{-}NIFS}.\mathcal{P}'(\mathsf{pk}_{\mathsf{fs}}, \{\mathsf{S}_k, \mathsf{CTX}_k, \mathsf{WIT}_k\}_{k\in[s]}, \{\mathsf{s}_k, \mathsf{ctx}_k, \mathsf{wit}_k\}_{k=0}^{s})$.
Compute $h \leftarrow \mathsf{Hash}({\mathsf{vk}_{\mathsf{fs}}}', z, \mathsf{CTX})$.
$(\mathsf{s}, \mathsf{ctx}, \mathsf{wit}) \leftarrow \mathsf{trace}(R_1, (h, (\{\mathsf{CTX}_k, \mathsf{ctx}_k\}_{k\in[s]}, \mathsf{ctx}_0, {\mathsf{vk}_{\mathsf{fs}}}', \mathsf{CTX}, \mathsf{pf})))$.
Output $\mathit{\Pi} := ((\mathsf{S}, \mathsf{CTX}, \mathsf{WIT}), (\mathsf{s}, \mathsf{ctx}, \mathsf{wit}))$.
```

```
$0/1 \leftarrow \mathcal{V}(\mathsf{vk}, z, \mathit{\Pi})$:
Parse $\mathit{\Pi}$ as $((\mathsf{S}, \mathsf{CTX}, \mathsf{WIT}), (\mathsf{s}, \mathsf{ctx}, \mathsf{wit}))$.
Check that $\mathsf{ctx.io} = \mathsf{Hash}({\mathsf{vk}_{\mathsf{fs}}}', z, \mathsf{CTX})$.
Check that $\mathsf{WIT}$ is a satisfied $\mathcal{R}_{\mathsf{ACCS}}$ witness to $\mathsf{CTX}$ and $\mathsf{wit}$ is a satisfied $\mathcal{R}_{\mathsf{CCCS}}$ witness to $\mathsf{ctx}$.
If the above checks hold, output 1; otherwise, output 0.
```

Theorem 3. *PCD scheme* $\mathsf{PCD} = (\mathcal{G}, \mathcal{K}, \mathcal{P}, \mathcal{V})$ *in Construction 3 for a class of compliance predicates* F *with constant depth in definition XX satisfies the perfect completeness, knowledge soundness, and zero knowledge in the random oracle.*

The proof of Theorem 3 is presented in Appendix B. We present the evaluation of complexity below.

Complexity. Denote the random oracle for sum-check protocol as $\mathbb{H}_{\mathsf{sc}}$.

The recursive cost at each step contains

- computing the compliance predicate φ;
- computing r times pf hash function Hash;
- invoking $\mathsf{zk\text{-}NIFS}.\mathcal{V}'$ with $2\log m + \log n$ random oracle queries, $O(s(dq + d\log m + \log n))$ $\mathbb{F}$-ops and s $\mathbb{G}$-ops.

The native prover at each step cost contains

- invoking $\mathsf{zk\text{-}NIFS}.\mathcal{P}'$ with $2\log m + \log n$ random oracle queries, $O(s(N + tm + qmd\log^2 d))$ $\mathbb{F}$-ops and s $\mathbb{G}$-ops;
- computing 1 time of hash function Hash;
- computing two satisfying committed CCS instances for the execution of $\mathcal{R}_0, \mathcal{R}_1$, which is dominated by computing the commitment of witness $\mathsf{wit}_0, \mathsf{wit}$ with $O(n)$ $\mathbb{G}$-ops.

The proof $\mathit{\Pi}$ consists of an atomic CCS instance $(\mathsf{S}, \mathsf{CTX}, \mathsf{WIT})$ and a committed CCS instance $(\mathsf{s}, \mathsf{ctx}, \mathsf{wit})$, which are linear in the size of $\mathcal{R}_{\varphi}$. While according to previous work in Nova [8] and HyperNova [9], we can fold these two instances with $\mathsf{zk\text{-}NIFS}.\mathcal{P}$ and apply a general SNARK (the folded instance is already zero-knowledge) to prove their validity. For example, instantiating a polynomial IOP based on Bulletproofs polynomial commitment schemes [24] for $(\mathsf{S}, \mathsf{CTX}, \mathsf{WIT})$ and $(\mathsf{s}, \mathsf{ctx}, \mathsf{wit})$ can reduce the proof size to $O(\log m)$.

PCD verifier cost contains

- invoking $\mathsf{zk\text{-}NIFS}.\mathcal{V}$ for two instances with $2\log m + \log n$ random oracle queries, $O((dq + d\log m + \log n))$ $\mathbb{F}$-ops and 2 $\mathbb{G}$-ops;
- verification of polynomial commitments with $O(\log n)$ random oracle queries and $O(n)$ $\mathbb{G}$-ops;
- verification of the IVC system for structure folds with $O(mn)$ $\mathbb{F}$-ops.

Acknowledgement

We would like to thank Binyi Chen and Benedikt Bünz from Espresso Systems for the inspiring discussions about the technical details of ProtoStar. We would also like to thank Yingfei from the Secbit zero-knowledge discussion group, Josh Beal from Yale University and Yanpei Guo from Beihang University for helping proofread our paper. This research has received partial support from HK RGC GRF under Grants PolyU 15216721, 15207522, 15202123, and NSFC Youth 62302418.

References

1. P. Valiant, "Incrementally verifiable computation or proofs of knowledge imply time/space efficiency," in *Theory of Cryptography: Fifth Theory of Cryptography Conference, TCC 2008, New York, USA, March 19-21, 2008. Proceedings 5*. Springer, 2008, pp. 1–18.
2. N. Bitansky, R. Canetti, A. Chiesa, and E. Tromer, "Recursive composition and bootstrapping for snarks and proof-carrying data," in *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*, 2013, pp. 111–120.
3. A. Chiesa, E. Tromer, and M. Virza, "Cluster computing in zero knowledge," in *Advances in Cryptology-EUROCRYPT 2015: 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part II 34*. Springer, 2015, pp. 371–403.
4. J. Bonneau, I. Meckler, V. Rao, and E. Shapiro, "Mina: Decentralized cryptocurrency at scale," *New York Univ. O (1) Labs, New York, NY, USA, Whitepaper*, pp. 1–47, 2020.
5. J. Bonneau, I. Meckler, and V. Rao, "Coda: Decentralized cryptocurrency at scale," *Cryptology ePrint Archive*, 2020.
6. J. Beal and B. Fisch, "Derecho: Privacy pools with proof-carrying disclosures," *Cryptology ePrint Archive*, 2023.
7. L. Eagen and A. Gabizon, "Protogalaxy: Efficient protostar-style folding of multiple instances," *Cryptology ePrint Archive*, 2023.
8. A. Kothapalli, S. Setty, and I. Tzialla, "Nova: Recursive zero-knowledge arguments from folding schemes," in *Annual International Cryptology Conference*. Springer, 2022, pp. 359–388.
9. A. Kothapalli and S. Setty, "Hypernova: Recursive arguments for customizable constraint systems," *Cryptology ePrint Archive*, 2023.
10. B. Bünz and B. Chen, "Protostar: Generic efficient accumulation/folding for special sound protocols," *Cryptology ePrint Archive*, 2023.
11. V. Buterin, "The different types of zk evm," https://vitalik.ca/general/2022/08/04/zkevm.html, 2022.
12. S. Bowe, J. Grigg, and D. Hopwood, "Recursive proof composition without a trusted setup," *Cryptology ePrint Archive*, 2019.
13. D. Boneh, J. Drake, B. Fisch, and A. Gabizon, "Halo infinite: Proof-carrying data from additive polynomial commitments," in *Advances in Cryptology–CRYPTO 2021: 41st Annual International Cryptology Conference, CRYPTO 2021, Virtual Event, August 16–20, 2021, Proceedings, Part I 41*. Springer, 2021, pp. 649–680.
14. B. Bünz, A. Chiesa, P. Mishra, and N. Spooner, "Proof-carrying data from accumulation schemes," *Cryptology ePrint Archive*, 2020.
15. B. Bünz, A. Chiesa, W. Lin, P. Mishra, and N. Spooner, "Proof-carrying data without succinct arguments," in *Advances in Cryptology–CRYPTO 2021: 41st Annual International Cryptology Conference, CRYPTO 2021, Virtual Event, August 16–20, 2021, Proceedings, Part I 41*. Springer, 2021, pp. 681–710.
16. M. Bellare, J. A. Garay, and T. Rabin, "Fast batch verification for modular exponentiation and digital signatures," in *Advances in Cryptology—EUROCRYPT'98: International Conference on the Theory and Application of Cryptographic Techniques Espoo, Finland, May 31–June 4, 1998 Proceedings 17*. Springer, 1998, pp. 236–250.
17. E. Ben-Sasson, A. Chiesa, and N. Spooner, "Interactive oracle proofs," in *Theory of Cryptography: 14th International Conference, TCC 2016-B, Beijing, China, October 31-November 3, 2016, Proceedings, Part II 14*. Springer, 2016, pp. 31–60.

18. S. Bowe, J. Grigg, and D. Hopwood, "Halo2," https://github. com/zcash/halo2, 2020.
19. M. Maller, S. Bowe, M. Kohlweiss, and S. Meiklejohn, "Sonic: Zero-knowledge snarks from linear-size universal and updatable structured reference strings," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, 2019, pp. 2111–2128.
20. A. Gabizon, Z. J. Williamson, and O. Ciobotaru, "Plonk: Permutations over lagrange-bases for oecumenical noninteractive arguments of knowledge," *Cryptology ePrint Archive*, 2019.
21. Z. Zhou, Z. Zhang, and J. Dong, "Proof-carrying data from multi-folding schemes," *Cryptology ePrint Archive*, 2023.
22. A. Kothapalli and S. Setty, "Supernova: Proving universal machine executions without universal circuits," *Cryptology ePrint Archive*, 2022.
23. T. Zheng, S. Gao, Y. Song, and B. Xiao, "Leaking arbitrarily many secrets: Any-out-of-many proofs and applications to ringct protocols," in *2023 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2023, pp. 2533–2550.
24. B. Bünz, J. Bootle, D. Boneh, A. Poelstra, P. Wuille, and G. Maxwell, "Bulletproofs: Short proofs for confidential transactions and more," in *2018 IEEE symposium on security and privacy (SP)*. IEEE, 2018, pp. 315–334.
25. T. Xie, J. Zhang, Z. Cheng, F. Zhang, Y. Zhang, Y. Jia, D. Boneh, and D. Song, "zkbridge: Trustless cross-chain bridges made practical," in *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, 2022, pp. 3003–3017.
26. S. Setty, J. Thaler, and R. Wahby, "Customizable constraint systems for succinct arguments," *Cryptology ePrint Archive*, 2023.
27. A. Chiesa, M. A. Forbes, and N. Spooner, "A zero knowledge sumcheck and its applications," *arXiv preprint arXiv:1704.02086*, 2017.
28. J. Bootle, A. Chiesa, and S. Liu, "Zero-knowledge iops with linear-time prover and polylogarithmic-time verifier," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2022, pp. 275–304.
29. S. Setty, "Spartan: Efficient and general-purpose zksnarks without trusted setup," in *Annual International Cryptology Conference*. Springer, 2020, pp. 704–737.
30. J. Thaler *et al.*, "Proofs, arguments, and zero-knowledge," *Foundations and Trends® in Privacy and Security*, vol. 4, no. 2–4, pp. 117–660, 2022.
31. J. T. Schwartz, "Fast probabilistic algorithms for verification of polynomial identities," *Journal of the ACM (JACM)*, vol. 27, no. 4, pp. 701–717, 1980.
32. C. Lund, L. Fortnow, H. Karloff, and N. Nisan, "Algebraic methods for interactive proof systems," *Journal of the ACM (JACM)*, vol. 39, no. 4, pp. 859–868, 1992.
33. T. Xie, Y. Zhang, and D. Song, "Orion: Zero knowledge proof with linear prover time," in *Annual International Cryptology Conference*. Springer, 2022, pp. 299–328.
34. B. Bünz, B. Fisch, and A. Szepieniec, "Transparent snarks from dark compilers," in *Advances in Cryptology–EUROCRYPT 2020: 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, May 10–14, 2020, Proceedings, Part I 39*. Springer, 2020, pp. 677–706.
35. A. Kate, G. M. Zaverucha, and I. Goldberg, "Constant-size commitments to polynomials and their applications," in *Advances in Cryptology-ASIACRYPT 2010: 16th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 5-9, 2010. Proceedings 16*. Springer, 2010, pp. 177–194.

36. J. Thaler, “Time-optimal interactive proofs for circuit evaluation,” in *Annual Cryptology Conference*. Springer, 2013, pp. 71–89.
37. D. Boneh, B. Bünz, and B. Fisch, “Batching techniques for accumulators with applications to iops and stateless blockchains,” in *Advances in Cryptology–CRYPTO 2019: 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18–22, 2019, Proceedings, Part I 39*. Springer, 2019, pp. 561–586.

A Security Proofs of Folding Scheme

In this section, we present the formal security proofs of our generic foldings scheme, including perfect completeness, knowledge soundness, and honest verifier zero-knowledge. For the former two properties, we mainly refer to the proof of the HyperNova [9].

Lemma 4. *(Perfect Completeness). Construction 1 satisfies perfect completeness.*

Proof. Consider public parameters $\mathsf{pp} = (m, n, N, l, t, q, d, \mathsf{pp}_{\mathsf{PC}}) \leftarrow \mathcal{G}(1^\lambda)$ and let $s_x = \log m$ and $s_y = \log n$. Consider arbitrary structures

$$\begin{aligned}
&\mathcal{S} = \{\widetilde{M_j}\}_{j\in[t]} \leftarrow \mathcal{A}(\mathsf{pp}),\\
&\mathcal{S}' = \{\widetilde{M'_j}\}_{j\in[t]}, \{S'_i\}_{i\in[q]}, \{c'_i\}_{i\in[q]} \leftarrow \mathcal{A}(\mathsf{pp}).
\end{aligned}$$

Consider prover and verifier key $(\mathsf{pk}, \mathsf{vk}) \leftarrow \mathcal{K}(\mathsf{pp}, \mathcal{S}, \mathcal{S}')$. Suppose the prover and verifier are provided an atomic CCS context

$$(C, v_0, \mathsf{io}, \boldsymbol{r}_x, \boldsymbol{r}_y, \{v_j\}_{j\in[t]}, v_z),$$

and a committed CCS context

$$(C', \mathsf{io}').$$

1. Sum-check protocol#1: suppose the prover is additionally provided the corresponding satisfying witnesses $\widetilde{\mathsf{wit}}$ and $\widetilde{\mathsf{wit}}'$. Since the input atomic CCS context-witness pair is satisfying, we have, for $\tilde{z} = (\widetilde{\mathsf{wit}, v_0}, \mathsf{io})$, that

$$\begin{aligned}
v_j &= \sum_{\boldsymbol{y}\in\{0,1\}^{s_y}} \widetilde{eq}(\boldsymbol{r}_y, \boldsymbol{y}) \cdot \widetilde{M_j}(\boldsymbol{r}_x, \boldsymbol{y})\\
&= \sum_{\boldsymbol{x}\in\{0,1\}^{s_x}} \widetilde{eq}(\boldsymbol{r}_x, \boldsymbol{x}) \cdot \left(\sum_{\boldsymbol{y}\in\{0,1\}^{s_y}} \widetilde{eq}(\boldsymbol{r}_y, \boldsymbol{y}) \cdot \widetilde{M_j}(\boldsymbol{x}, \boldsymbol{y}) \right)\\
&= \sum_{\boldsymbol{x}\in\{0,1\}^{s_x}} L_j(\boldsymbol{x}), \forall j \in [t].
\end{aligned}$$

Moreover, since the input committed CCS context-witness pair is satisfying, we have, for $\tilde{z}'(\boldsymbol{y}) = (\widetilde{\mathsf{wit}', 1}, \mathsf{io}')(\boldsymbol{y})$, that

$$0 = \sum_{i\in[q]} c'_i \cdot \prod_{j\in S'_i} \left(\sum_{y\in\{0,1\}^{s_y}} \widetilde{M'_j}(\boldsymbol{x}, \boldsymbol{y}) \cdot \tilde{z}'(\boldsymbol{y}) \right), \forall \boldsymbol{x} \in \{0,1\}^{s_x}.$$

Treating the right-hand side of the above equation as a polynomial in $\boldsymbol{x}$, because it is multilinear and vanishes on all $\boldsymbol{x} \in \{0,1\}^{s_x}$, we have that it must be the zero polynomial. Therefore, we have, for $\boldsymbol{\alpha}$ sampled by the verifier, that

$$\begin{aligned}
0 &= \sum_{i \in [q]} c'_i \cdot \prod_{j \in S'_i} \left(\sum_{\boldsymbol{y} \in \{0,1\}^{s_y}} \widetilde{M}'_j(\boldsymbol{\alpha}, \boldsymbol{y}) \cdot \tilde{z}'(\boldsymbol{y}) \right) \\
&= \sum_{\boldsymbol{x} \in \{0,1\}^{s_x}} \widetilde{eq}(\boldsymbol{\alpha}, \boldsymbol{x}) \cdot \left(\sum_{i \in [q]} c'_i \prod_{j \in S'_i} \left(\sum_{\boldsymbol{y} \in \{0,1\}^{s_y}} \widetilde{M}'_j(\boldsymbol{x}, \boldsymbol{y}) \cdot \tilde{z}'(\boldsymbol{y}) \right) \right) \\
&= \sum_{\boldsymbol{x} \in \{0,1\}^{s_x}} Q(\boldsymbol{x}).
\end{aligned}$$

For γ sampled by the verifier, by linearity, we have that

$$\begin{aligned}
\sum_{j \in [t]} \gamma^j \cdot v_j &= \sum_{\boldsymbol{x} \in \{0,1\}^{s_x}} \left(\left(\sum_{j \in [t]} \gamma^j \cdot L_j(\boldsymbol{x}) \right) + \gamma^{t+1} \cdot Q(\boldsymbol{x}) \right) \\
&= \sum_{\boldsymbol{x} \in \{0,1\}^{s_x}} f(\boldsymbol{x}).
\end{aligned}$$

Therefore, by the perfect completeness of the sum-check protocol, we have for $e_1 = \widetilde{eq}(\boldsymbol{r}_x, \boldsymbol{r}'_x), e_2 = \widetilde{eq}(\boldsymbol{\alpha}, \boldsymbol{r}'_x)$ and

$$\begin{aligned}
\sigma_j &= \sum_{\boldsymbol{y} \in \{0,1\}^{s_y}} \widetilde{eq}(\boldsymbol{r}_y, \boldsymbol{y}) \cdot \widetilde{M}_j(\boldsymbol{r}'_x, \boldsymbol{y}), \forall j \in [t], \\
\sigma'_j &= \sum_{\boldsymbol{y} \in \{0,1\}^{s_y}} \widetilde{M}'_j(\boldsymbol{r}'_x, \boldsymbol{y}) \cdot \tilde{z}'(\boldsymbol{y}), \forall j \in [t],
\end{aligned}$$

that

$$\begin{aligned}
c_x &= f(\boldsymbol{r}'_x) \\
&= \left(\sum_{j \in [t]} \gamma^j \cdot L_j(\boldsymbol{r}'_x) \right) + \gamma^{t+1} \cdot Q(\boldsymbol{r}'_x) \\
&= \left(\sum_{j \in [t]} \gamma^j \cdot e_1 \cdot \sigma_j \right) + \gamma^{t+1} \cdot e_2 \cdot \sum_{i \in [q]} c'_i \cdot \prod_{j \in S'_i} \sigma'_j.
\end{aligned}$$

2. Sum-check protocol#2: According to the results of sum-check protocol#1, for $\tilde{z}(\boldsymbol{x}), \tilde{z}'(\boldsymbol{x})$ and sampled $\boldsymbol{r}'_x$, we have

$$\begin{aligned}
\sigma_j &= \sum_{\boldsymbol{y}\in\{0,1\}^{s_y}} \widetilde{eq}(\boldsymbol{r}_y, \boldsymbol{y}) \cdot \widetilde{M}_j(\boldsymbol{r}'_x, \boldsymbol{y}) \\
&= \sum_{\boldsymbol{y}\in\{0,1\}^{s_y}} R_j(\boldsymbol{y}), \forall j \in [t], \\
v_z &= \sum_{\boldsymbol{y}\in\{0,1\}^{s_y}} \widetilde{eq}(\boldsymbol{r}_y, \boldsymbol{y}) \cdot \tilde{z}(\boldsymbol{y}) \\
\sigma'_j &= \sum_{\boldsymbol{y}\in\{0,1\}^{s_y}} \widetilde{M}'_j(\boldsymbol{r}'_x, \boldsymbol{y}) \cdot \tilde{z}'(\boldsymbol{y}), \\
&= \sum_{\boldsymbol{y}\in\{0,1\}^{s_y}} T_j(\boldsymbol{y}), \forall j \in [t].
\end{aligned}$$

For δ sampled by the verifier, by linearity, we have that

$$\begin{aligned}
&\sum_{j\in[t]} \delta^j \cdot \sigma_j + \delta^{t+1} \cdot v_z + \delta^{t+1} \cdot \sum_{j\in[t]} \delta^j \cdot \sigma'_j \\
=& \sum_{j\in[t]} \delta^j \cdot R_j(\boldsymbol{y}) + \delta^{t+1} \cdot S(\boldsymbol{y}) + \delta^{t+1} \cdot \sum_{j\in[t]} \delta^j \cdot T_j(\boldsymbol{y}) \\
=& \sum_{j\in[t]} g(\boldsymbol{y}).
\end{aligned}$$

Therefore, by the perfect completeness of the sum-check protocol, we have for

$$\begin{aligned}
&\epsilon = \tilde{z}(\boldsymbol{r}'_y), \\
&\epsilon' = \tilde{z}'(\boldsymbol{r}'_y), \\
&\theta_j = \widetilde{M}_j(\boldsymbol{r}'_x, \boldsymbol{r}'_y), \forall j \in [t], \\
&\theta'_j = \widetilde{M}'_j(\boldsymbol{r}'_x, \boldsymbol{r}'_y), \forall j \in [t],
\end{aligned}$$

that

$$\begin{aligned}
c_y &= g(\boldsymbol{r}'_y) \\
&= \sum_{j\in[t]} \delta^j \cdot R_j(\boldsymbol{r}'_y) + \delta^{t+1} \cdot S(\boldsymbol{y}) + \delta^{t+1} \cdot \sum_{j\in[t]} \delta^j \cdot T_j(\boldsymbol{r}'_y) \\
&= \sum_{j\in[t]} \delta^j \cdot e_3 \cdot \theta_j + \delta^{t+1} \cdot \epsilon + \delta^{t+1} \cdot \sum_{j\in[t]} \delta^j \cdot \theta_j \cdot \epsilon'.
\end{aligned}$$

The above two steps imply that the verifier will not abort. Now, consider the atomic CCS context obtained from $\mathcal{R}'$ as

$$(C', 1, \mathsf{io}', \boldsymbol{r}'_x, \boldsymbol{r}'_y, \{\theta'_j\}_{j\in[t]}, \epsilon').$$

By the precondition that the committed CCS context (C', io') is satisfied by $\widetilde{\mathsf{wit}}'$ and by the definition of $\{\theta_j'\}_{j\in[t]}, \epsilon'$ we have that this new atomic CCS context is satisfied by the witness $\widetilde{\mathsf{wit}}'$.

Therefore, for random η sampled by the verifier, and folded structure $\mathcal{S}^*$ with $\{M_j^* = M_j + \eta \cdot M_j'\}_{j=1}^{t}$, folded context $C^* = C + \eta \cdot C'$, $v_0^* = v_0 + \eta \cdot 1$, $\mathsf{io}^* = \mathsf{io} + \eta \cdot \mathsf{io}'$, $v_j^* = \theta_j + \eta \cdot \theta_j'$, $v_z^* = \epsilon_j + \eta \cdot \epsilon_j'$, we have that the output folded atomic CCS context

$$(C^*, v_0^*, \mathsf{io}^*, \boldsymbol{r}_x', \boldsymbol{r}_y', \{v_j^*\}_{j\in[t]}, v_z^*).$$

is satisfied by the witness $\widetilde{\mathsf{wit}}^* \leftarrow \widetilde{\mathsf{wit}} + \eta \cdot \widetilde{\mathsf{wit}}'$ under the structure $\mathcal{S}^*$ by the linearity and the additive homomorphism property of the commitment scheme. □

Lemma 5. *(Knowledge Soundness). Construction 1 satisfies knowledge soundness.*

Proof. Consider an adversary $\mathcal{A}$ that adaptively picks the structures and contexts, and a malicious prover $\mathcal{P}^*$ that succeeds with probability ϵ. Let $\mathsf{pp} \leftarrow \mathcal{G}(1^\lambda)$. Suppose on input pp and random tape η, the adversary $\mathcal{A}$ picks two structures

$$\begin{aligned}&\mathcal{S} = \{\widetilde{M_j}\}_{j\in[t]},\\ &\mathcal{S}' = \{\widetilde{M_j'}\}_{j\in[t]}, \{S_i'\}_{i\in[q]}, \{c_i'\}_{i\in[q]}.\end{aligned}$$

a new committed CCS context

$$\mathsf{ctx} = (C, v_0, \mathsf{io}, \boldsymbol{r}_x, \boldsymbol{r}_y, \{v_j\}_{j\in[t]}, v_z),$$

and committed CCS context

$$\mathsf{ctx}' = (C', \mathsf{io}'),$$

and some auxiliary state st.

1. Extraction Algorithm: we construct an expected-polynomial time extractor Ext that succeeds with probability $\epsilon - \mathrm{negl}(\lambda)$ in obtaining satisfying witnesses for the original contexts as follows.

```
$\mathsf{Ext}(\mathsf{pp}, \rho)$:
1 : Obtain the output tuple from $\mathcal{A}$:
        $(\mathcal{S}, \mathcal{S}', \mathsf{ctx}, \mathsf{ctx}', \mathsf{st}) \leftarrow \mathcal{A}(\mathsf{pp}, \rho).$
2 : Compute $(\mathsf{pk}, \mathsf{vk}) \leftarrow \mathcal{K}(\mathsf{pp}, \mathcal{S}, \mathcal{S}')$.
3 : Run the folding interaction#1
        $(\mathcal{S}_1^*, \mathsf{ctx}_1^*, \widetilde{\mathsf{wit}}_1^*) \leftarrow \langle \mathcal{P}^*, \mathcal{V} \rangle((\mathsf{pk}, \mathsf{vk}), \mathcal{S}, \mathcal{S}', \mathsf{ctx}, \mathsf{ctx}', \mathsf{st})$
    *once* with the final verifier challenge $\eta_1 \leftarrow\!\!\$\ \mathbb{F}$.
4 : Abort if $(\mathsf{pp}, \mathcal{S}_1^*, \mathsf{ctx}_1^*, \widetilde{\mathsf{wit}}_1^*) \notin \mathcal{R}_{\mathsf{ACCS}}$.
5 : Run the folding interaction#2
        $(\mathcal{S}_2^*, \mathsf{ctx}_2^*, \widetilde{\mathsf{wit}}_2^*) \leftarrow \langle \mathcal{P}^*, \mathcal{V} \rangle((\mathsf{pk}, \mathsf{vk}), \mathcal{S}, \mathcal{S}', \mathsf{ctx}, \mathsf{ctx}', \mathsf{st}))$
    with a different verifier's final challenge $\eta_2 \leftarrow\!\!\$\ \mathbb{F}$ while maintaining the same
    prior randomness. Keep doing so until $(\mathsf{pp}, \mathcal{S}_2^*, \mathsf{ctx}_2^*, \widetilde{\mathsf{wit}}_2^*) \in \mathcal{R}_{\mathsf{ACCS}}$.
6 : Abort if $\eta_1 = \eta_2$ or $\mathcal{S}_1^* \neq \mathcal{S}_2^*$.
7 : Interpolating points $(\eta_1, \widetilde{\mathsf{wit}}_1^*)$ and $(\eta_2, \widetilde{\mathsf{wit}}_2^*)$, retrieve the witness polynomials
    $\widetilde{\mathsf{wit}}$ and $\widetilde{\mathsf{wit}}'$ such that for $i \in \{1, 2\}$
        $\widetilde{\mathsf{wit}} + \eta_i \cdot \widetilde{\mathsf{wit}}' = \widetilde{\mathsf{wit}}_i^*.$
8 : Output $(\widetilde{\mathsf{wit}}, \widetilde{\mathsf{wit}}')$.
```

We first demonstrate that the extractor Ext runs in expected polynomial time. Observe that Ext runs the folding interaction#1 once, and if it does not abort, keeps rerunning the folding interaction#2 until $\mathcal{P}^*$ succeeds. Let W denote the event that the extractor does not abort at step 4, and $\bar{W}$ denotes that the event W does not happen. Define the number of folding interactions Ext runs in total as a variable X (i.e., number of rewinds). We can calculate its expectation as

$$\mathbb{E}[X] = \Pr[W] \cdot (1 + \frac{1}{\Pr[\langle \mathcal{P}^*, \mathcal{V} \rangle \text{ succeeds}]}) + \Pr[\bar{W}] \cdot 1 = \epsilon \cdot (1 + \frac{1}{\epsilon}) + (1 - \epsilon) \cdot 1 = 2.$$

Therefore, we have that the extractor runs in the expected polynomial time.

2. Advantage Analysis: We now analyze Ext's success probability. We must demonstrate that Ext succeeds in producing $\widetilde{\mathsf{wit}}$ and $\widetilde{\mathsf{wit}}'$ such that

$$(\mathsf{pp}, \mathcal{S}, \mathsf{ctx}, \widetilde{\mathsf{wit}}) \in \mathcal{R}_{\mathsf{ACCS}} \text{ and } (\mathsf{pp}, \mathcal{S}', \mathsf{ctx}', \widetilde{\mathsf{wit}}') \in \mathcal{R}_{\mathsf{CCCS}}$$

, with probability $\epsilon - \mathrm{negl}(\lambda)$.

To do so, we first show that the extractor successfully produces *some* output (i.e., does not abort) in less than $\sqrt[3]{|\mathbb{F}|}$ rewinding steps with probability $\epsilon - \mathrm{negl}(\lambda)$. Indeed, by the malicious prover's success probability, we have that the extractor does not abort at step (4) with probability ϵ. Given that the extractor does not abort at step (4), by Markov's inequality, we have that the extractor rewinds more than $\sqrt[3]{|\mathbb{F}|}$ times with probability

$$\Pr[X \geq \sqrt[3]{|\mathbb{F}|}] \leq \frac{\mathbb{E}[X]}{\sqrt[3]{|\mathbb{F}|}} = \frac{2}{\sqrt[3]{|\mathbb{F}|}},$$

where X is the random variable of the number of running folding interactions. Thus, the probability that the extractor does not abort at step (4) and requires less than $\sqrt[3]{|\mathbb{F}|}$ rewinds is $(1 - 2/\sqrt[3]{|\mathbb{F}|}) \cdot \epsilon$.

Now, suppose that the extractor does not abort at step (4) and requires less than $\sqrt[3]{|\mathbb{F}|}$ rewinds. This ensures that the extractor tests at most $\sqrt[3]{|\mathbb{F}|}$ values for η. Since the challenges are sampled uniformly in random form $|\mathbb{F}|$, the probability that $\rho^{(1)} \neq \rho^{(2)}$ is $1 - \sqrt[3]{|\mathbb{F}|}^2/|\mathbb{F}|$. Therefore, assuming $\sqrt[3]{|\mathbb{F}|}^2 \geq 2$, we have that the probability the extractor successfully produces some output under $\sqrt[3]{|\mathbb{F}|}$ rewinding steps is

$$\begin{aligned}
\Pr[X < \sqrt[3]{|\mathbb{F}|}] \cdot \Pr[\rho^{(1)} \neq \rho^{(2)}] &= (1 - \frac{2}{\sqrt[3]{|\mathbb{F}|}}) \cdot \epsilon \cdot (1 - \frac{\sqrt[3]{|\mathbb{F}|}^2}{|\mathbb{F}|}) \\
&= (1 - \frac{2}{\sqrt[3]{|\mathbb{F}|}} - \frac{\sqrt[3]{|\mathbb{F}|}^2}{|\mathbb{F}|} + \frac{2}{|\mathbb{F}|}) \\
&= \epsilon - \mathrm{negl}(\lambda).
\end{aligned}$$

Next, if the extractor does not abort, we show that the extractor succeeds in producing satisfying witnesses with probability $1 - \mathrm{negl}(\lambda)$. This brings the overall extractor success probability to $(\epsilon - \mathrm{negl}(\lambda)) \cdot (1 - \mathrm{negl}(\lambda))$.

For $i \in \{1, 2\}$, let $\mathsf{ctx}_i^* = (C_i^*, v_{0,i}^*, \mathsf{io}_i^*, r_{x,i}^*, v_{1,i}^*, ..., v_{t,i}^*, v_{z,i}^*)$. We first show that the retrieved polynomials are valid openings to the corresponding commitments in the instance. For $i \in \{1, 2\}$, since $\widetilde{\mathsf{wit}}_i^*$ is a satisfying witness, by construction,

$$\begin{aligned}
&\mathsf{Commit}(\mathsf{pp}, \widetilde{\mathsf{wit}}) + \eta_i \cdot \mathsf{Commit}(\mathsf{pp}, \widetilde{\mathsf{wit}}') \\
&= \mathsf{Commit}(\mathsf{pp}, \widetilde{\mathsf{wit}} + \eta_i \cdot \widetilde{\mathsf{wit}}') \\
&= \mathsf{Commit}(\mathsf{pp}, \widetilde{\mathsf{wit}}_i^*) \\
&= C_i^* \\
&= C + \eta_i \cdot C'.
\end{aligned}$$

Interpolating, we have that

$$\mathsf{Commit}(\mathsf{pp}, \widetilde{\mathsf{wit}}) = C, \tag{4}$$

$$\mathsf{Commit}(\mathsf{pp}, \widetilde{\mathsf{wit}}') = C'. \tag{5}$$

Next, we must argue that $\widetilde{\mathsf{wit}}$ and $\widetilde{\mathsf{wit}}'$ satisfy the remainder of the instances $(\mathcal{S}, \mathsf{ctx})$ and $(\mathcal{S}', \varphi')$ respectively.

Consider $\{\theta_j\}_{j\in[t]}, \{\theta'_j\}_{j\in[t]}$ and ϵ, ϵ' sent by the prover which by the extractor's construction are identical across all executions of the interaction. By the verifier's computation we have that for $i \in \{1,2\}$ and all $j \in [t]$

$$v_{j,i} = \theta_j + \eta_i \cdot \theta'_j, \tag{6}$$

$$v_{z,i} = \epsilon + \eta_i \cdot \epsilon'. \tag{7}$$

Now, because $\widetilde{\mathsf{wit}}^*_i$ is a satisfying witness, for $i \in \{1,2\}$ we have for all $j \in [t]$ that

$$\begin{aligned} v_{j,i} &= \widetilde{M}^*_{j,i}(\boldsymbol{r}'_x, \boldsymbol{r}'_y), \\ v_{z,i} &= \tilde{z}^*_i(\boldsymbol{r}'_y), \end{aligned}$$

where $\widetilde{M}^*_{j,i} = \widetilde{M}_j + \eta_i \cdot \widetilde{M}_j$, $\tilde{z}^*_i = \widetilde{(\mathsf{wit}^*_i, v^*_{0,i}, \mathsf{io}^*_i)} = \tilde{z} + \eta \cdot \tilde{z}'$.

Meanwhile, according to equations (6) and (7), for $i \in \{1,2\}$ and $j \in [t]$, we have

$$\begin{aligned} \theta_j + \eta_i \cdot \theta'_j = v_{j,i} &= \widetilde{M}_j(\boldsymbol{r}'_x, \boldsymbol{r}'_y) + \eta_i \cdot \widetilde{M}'_j(\boldsymbol{r}'_x, \boldsymbol{r}'_y), \\ \epsilon + \eta_i \cdot \epsilon' = v_{z,i} &= \tilde{z}(\boldsymbol{r}'_y) + \eta_i \cdot \tilde{z}'(\boldsymbol{r}'_y), \end{aligned}$$

where $\tilde{z} = \widetilde{(\mathsf{wit}_i, v_{0,i}, \mathsf{io}_i)}$ and $\tilde{z}' = \widetilde{(\mathsf{wit}', 1, \mathsf{io}')}$. Interpolating, we have that, for all $j \in [t]$

$$\begin{aligned} \theta_j &= \widetilde{M}_j(\boldsymbol{r}'_x, \boldsymbol{r}'_y), \\ \theta'_j &= \widetilde{M}'_j(\boldsymbol{r}'_x, \boldsymbol{r}'_y), \\ \epsilon &= \tilde{z}(\boldsymbol{r}'_y), \\ \epsilon' &= \tilde{z}'(\boldsymbol{r}'_y). \end{aligned}$$

Thus, because the verifier does not abort at step 11, we have that

$$\begin{aligned} c_y &= \sum_{j\in[t]} \delta^j \cdot e_3 \cdot \theta_j + \delta^{t+1} \cdot e_3 \cdot \epsilon + \delta^{t+1} \cdot \sum_{j\in[t]} \delta^j \cdot \theta'_j \cdot \epsilon' \\ &= \sum_{j\in[t]} \delta^j \cdot \widetilde{eq}(\boldsymbol{r}_y, \boldsymbol{r}'_y) \cdot \theta_j + \delta^{t+1} \cdot \widetilde{eq}(\boldsymbol{r}_y, \boldsymbol{r}'_y) \cdot \epsilon + \delta^{t+1} \cdot \sum_{j\in[t]} \delta^j \cdot \theta'_j \cdot \epsilon' \\ &= \sum_{j\in[t]} \delta^j \cdot \widetilde{eq}(\boldsymbol{r}_y, \boldsymbol{r}'_y) \cdot \widetilde{M}_j(\boldsymbol{r}'_x, \boldsymbol{r}'_y) + \delta^{t+1} \cdot \widetilde{eq}(\boldsymbol{r}_y, \boldsymbol{r}'_y) \cdot \tilde{z}(\boldsymbol{r}'_y) + \delta^{t+1} \cdot \sum_{j\in[t]} \delta^j \cdot \widetilde{M}'_j(\boldsymbol{r}'_x, \boldsymbol{r}'_y) \cdot \tilde{z}'(\boldsymbol{r}'_y) \\ &= \sum_{j\in[t]} \delta^j \cdot R_j(\boldsymbol{r}'_y) + \delta^{t+1} \cdot S(\boldsymbol{r}'_y) + \delta^{t+1} \cdot \sum_{j\in[t]} \delta^j \cdot T_j(\boldsymbol{r}'_y) \\ &= g(\boldsymbol{r}'_y), \end{aligned}$$

by the soundness of the sum-check protocol#2, this implies that with probability $1 - O(d \cdot s_y)/|\mathbb{F}| = 1 - \mathrm{negl}(\lambda)$ over the choice of $\boldsymbol{r}'_y$,

$$\begin{aligned}
&\sum_{j \in [t]} \delta^j \cdot \sigma_j + \delta^{t+1} \cdot v_z + \delta^{t+1} \cdot \sum_{j \in [t]} \delta^j \cdot \sigma'_j \\
= &\sum_{\boldsymbol{y} \in \{0,1\}^{s_y}} g(\boldsymbol{y}) \\
= &\sum_{\boldsymbol{y} \in \{0,1\}^{s_y}} \left(\sum_{j \in [t]} \delta^j \cdot R_j(\boldsymbol{y}) + \delta^{t+1} \cdot S(\boldsymbol{y}) + \delta^{t+1} \cdot \sum_{j \in [t]} \delta^j \cdot T_j(\boldsymbol{y}) \right) \\
= &\sum_{j \in [t]} \delta^j \cdot \sum_{\boldsymbol{y} \in \{0,1\}^{s_y}} R_j(\boldsymbol{y}) + \delta^{t+1} \cdot \sum_{\boldsymbol{y} \in \{0,1\}^{s_y}} S(\boldsymbol{y}) + \delta^{t+1} \cdot \sum_{j \in [t]} \delta^j \cdot \sum_{\boldsymbol{y} \in \{0,1\}^{s_y}} T_j(\boldsymbol{y}).
\end{aligned}$$

By the Schwartz-Zippel lemma [Sch80], this implies that with probability $1 - O(t)/|\mathbb{F}| = 1 - \mathrm{negl}(\lambda)$ over the choice of δ, for all $j \in [t]$, we have

$$\begin{aligned}
\sigma_j &= \sum_{\boldsymbol{y} \in \{0,1\}^{s_y}} R_j(\boldsymbol{y}) = \sum_{\boldsymbol{y} \in \{0,1\}^{s_y}} \widetilde{eq}(\boldsymbol{r}_y, \boldsymbol{y}) \cdot \widetilde{M}_j(\boldsymbol{r}'_x, \boldsymbol{y}), \\
v_z &= \sum_{\boldsymbol{y} \in \{0,1\}^{s_y}} S(\boldsymbol{y}) = \sum_{\boldsymbol{y} \in \{0,1\}^{s_y}} \widetilde{eq}(\boldsymbol{r}_y, \boldsymbol{y}) \cdot \tilde{z}(\boldsymbol{y}), \\
\sigma'_j &= \sum_{\boldsymbol{y} \in \{0,1\}^{s_y}} T_j(\boldsymbol{y}) = \sum_{\boldsymbol{y} \in \{0,1\}^{s_y}} \widetilde{M}'_j(\boldsymbol{r}'_x, \boldsymbol{y}) \cdot \tilde{z}'(\boldsymbol{y}).
\end{aligned}$$

Thus, because the verifier does not abort at step 5, we have that

$$\begin{aligned}
c_x &= \left(\sum_{j \in [t]} \gamma^j \cdot e_1 \cdot \sigma_j \right) + \left(\gamma^{t+1} \cdot e_2 \cdot \sum_{i \in [q]} c'_i \cdot \prod_{j \in S_i} \sigma_j \right) \\
&= \left(\sum_{j \in [t]} \gamma^j \cdot \widetilde{eq}(\boldsymbol{r}_x, \boldsymbol{r}'_x) \cdot \sigma_j \right) + \left(\gamma^{t+1} \cdot \widetilde{eq}(\boldsymbol{\alpha}, \boldsymbol{r}'_x) \cdot \sum_{i \in [q]} c'_i \cdot \prod_{j \in S_i} \theta_j \right) \\
&= \left(\sum_{j \in [t]} \gamma^j \cdot \widetilde{eq}(\boldsymbol{r}_x, \boldsymbol{r}'_x) \cdot \sum_{\boldsymbol{y} \in \{0,1\}^{s_y}} \widetilde{eq}(\boldsymbol{r}_y, \boldsymbol{y}) \cdot \widetilde{M}_j(\boldsymbol{r}'_x, \boldsymbol{y}) \right) \\
&+ \left(\gamma^{t+1} \cdot \widetilde{eq}(\boldsymbol{\alpha}, \boldsymbol{r}'_x) \cdot \sum_{i \in [q]} c_i \cdot \prod_{j \in S_i} \sum_{\boldsymbol{y} \in \{0,1\}^{s_y}} \widetilde{M}'_j(\boldsymbol{r}'_x, \boldsymbol{y}) \cdot \tilde{z}'(\boldsymbol{y}) \right) \\
&= \sum_{j \in [t]} \gamma^j \cdot L_j(\boldsymbol{r}'_x) + \gamma^{t+1} \cdot Q(\boldsymbol{r}'_x) \\
&= f(\boldsymbol{r}'_x),
\end{aligned}$$

by the soundness of the sum-check protocol#1, this implies that with probability $1 - O(d \cdot s_x)/|\mathbb{F}| = 1 - \mathrm{negl}(\lambda)$ over the choice of $\boldsymbol{r}'_x$,

$$\begin{aligned}
\sum_{j\in[t]} \gamma^j \cdot v_j + \gamma^{t+1} \cdot 0 &= \sum_{\boldsymbol{x}\in\{0,1\}^{s_x}} f(x) \\
&= \sum_{\boldsymbol{x}\in\{0,1\}^{s_x}} \left(\left(\sum_{j\in[t]} \gamma^j \cdot L_j(\boldsymbol{x}) \right) + \gamma^{t+1} \cdot Q(\boldsymbol{x}) \right) \\
&= \sum_{\boldsymbol{x}\in\{0,1\}^{s_x}} \gamma^j \cdot \left(\sum_{j\in[t]} L_j(\boldsymbol{x}) \right) + \gamma^{t+1} \cdot \sum_{\boldsymbol{x}\in\{0,1\}^{s_x}} Q(x).
\end{aligned}$$

By the Schwartz-Zippel lemma [Sch80], this implies that with probability $1 - O(t)/|\mathbb{F}| = 1 - \mathrm{negl}(\lambda)$ over the choice of γ, for all $j \in [t]$, we have

$$\begin{aligned}
v_j &= \sum_{\boldsymbol{x}\in\{0,1\}^{s_x}} L_j(\boldsymbol{x}), \\
0 &= \sum_{\boldsymbol{x}\in\{0,1\}^{s_x}} Q(x).
\end{aligned}$$

Therefore,

$$\begin{aligned}
v_j &= \sum_{\boldsymbol{x}\in\{0,1\}^{s_x}} L_j(\boldsymbol{x}) \\
&= \sum_{\boldsymbol{x}\in\{0,1\}^{s_x}} \widetilde{eq}(\boldsymbol{r}_x, \boldsymbol{x}) \cdot \left(\sum_{\boldsymbol{y}\in\{0,1\}^{s}} \widetilde{eq}(\boldsymbol{r}_y, \boldsymbol{y}) \cdot \widetilde{M}_j(\boldsymbol{x}, \boldsymbol{y}) \right) \\
&= \widetilde{M}_j(\boldsymbol{r}_x, \boldsymbol{r}_y).
\end{aligned}$$

This implies that $\widetilde{\mathsf{wit}}$ is a satisfying witness to $(\mathcal{S}, \mathsf{ctx})$. Finally, we have that

$$\begin{aligned}
0 &= \sum_{\boldsymbol{x}\in\{0,1\}^{s_x}} Q(x) \\
&= \sum_{\boldsymbol{x}\in\{0,1\}^{s_x}} \widetilde{eq}(\boldsymbol{\alpha}, \boldsymbol{x}) \cdot \left(\sum_{i\in[q]} c'_i \cdot \prod_{j\in S_i} \left(\sum_{\boldsymbol{y}\in\{0,1\}^{s_y}} \widetilde{M}'_j(\boldsymbol{x}, \boldsymbol{y}) \cdot \tilde{z}'(\boldsymbol{y}) \right) \right) \\
&= \sum_{i\in[q]} c'_i \cdot \prod_{j\in S_i} \left(\sum_{\boldsymbol{y}\in\{0,1\}^{s_y}} \widetilde{M}'_j(\boldsymbol{\alpha}, \boldsymbol{y}) \cdot \tilde{z}'(\boldsymbol{y}) \right).
\end{aligned}$$

By the Schwartz-Zippel lemma, this implies that with probability $1 - s_x/|\mathbb{F}| = 1 - \mathrm{negl}(\lambda)$ over the choice of $\boldsymbol{\alpha}$, we have that for all $\boldsymbol{x} \in \{0,1\}^{s_x}$

$$0 = \sum_{i\in[q]} c'_i \cdot \prod_{j\in S_i} \left(\sum_{\boldsymbol{y}\in\{0,1\}^{s_y}} \widetilde{M}'_j(\boldsymbol{x}, \boldsymbol{y}) \cdot \tilde{z}'(\boldsymbol{y}) \right).$$

This implies that $\widetilde{\mathsf{wit}}'$ is a satisfying witness to $(\mathcal{S}', \mathsf{ctx}')$. Thus, if the extractor does not abort, it succeeds in producing satisfying witness $\widetilde{\mathsf{wit}}, \widetilde{\mathsf{wit}}'$ with probability $1 - \mathrm{negl}(\lambda)$. □

Lemma 6. *(Honest Verifier Zero Knowledge). Construction 1 satisfies honest verifier zero knowledge.*

Proof. Consider an adversary $\mathcal{A}$ that adaptively picks the structures and contexts. Let $\mathsf{pp} \leftarrow \mathcal{G}(1^\lambda)$. Suppose on input pp, the adversary $\mathcal{A}$ picks two structures

$$\begin{aligned}\mathcal{S} &= \{\widetilde{M_j}\}_{j\in[t]},\\ \mathcal{S}' &= \{\widetilde{M_j'}\}_{j\in[t]}, \{S_i'\}_{i\in[q]}, \{c_i'\}_{i\in[q]},\end{aligned}$$

a new committed CCS context-witness pair

$$(\mathsf{ctx}, \mathsf{wit}) = (C, v_0, \mathsf{io}, \boldsymbol{r}_x, \boldsymbol{r}_y, \{v_j\}_{j\in[t]}, v_z, \mathsf{wit}),$$

and committed CCS context-witness pair

$$(\mathsf{ctx}', \mathsf{wit}') = (C', \mathsf{io}'.\mathsf{wit}).$$

With the keys generated by $(\mathsf{pk}, \mathsf{vk}) \leftarrow \mathcal{K}(\mathsf{pp}, \mathcal{S}, \mathcal{S}')$, the non-deterministic function trace produces an interaction transcript tr between honest $\mathcal{P}$ and $\mathcal{V}$ of Π_{fold} on input $((\mathsf{pk}, \mathsf{vk}), (\mathsf{ctx}, \mathsf{ctx}'), (\mathsf{wit}, \mathsf{wit}'))$.

Next, we construct a PPT simulator Sim producing the trace $\hat{\mathsf{tr}}$ with indistinguishable distribution from tr with the input of $(\mathsf{pp}, \{(\mathcal{S}, \mathcal{S}'), (\mathsf{ctx}, \mathsf{ctx}'), \rho)$.

To begin with, the simulator inputs a random challenge $\hat{\eta}$ to aggregate the structures and contexts accordingly to obtain the folded structure $\hat{S}^*$ containing

$$\hat{M_j}^* = \hat{\eta} \cdot M_j + \hat{\eta}^2 \cdot M_j'$$

for all $j \in [t]$, and part of the folded context $\hat{\mathsf{ctx}}^*$ containing

$$\begin{aligned}&\hat{v_0}^* \leftarrow \hat{\eta} \cdot v_0 + \hat{\eta}^2 \cdot 1,\\ &\hat{\mathsf{io}}^* \leftarrow \hat{\eta} \cdot \mathsf{io} + \hat{\eta}^2 \cdot \mathsf{io}'.\\ &\hat{v_j}^* \leftarrow \hat{\eta} \cdot v_j + \hat{\eta}^2 \cdot v_j' \ \forall j \in [t],\end{aligned}$$

To simulate the trace $\hat{\mathsf{tr}}$, the simulator samples a random vector in $\mathbb{F}n - l - 1$ as $\hat{\mathsf{wit}}^*$, and compute the commitment on the random witness $\boldsymbol{w}$ in the masking instance as

$$\hat{C}'' = \mathsf{Commit}(\mathsf{pp}, \hat{\mathsf{wit}}^*) - \hat{\eta} \cdot C - \hat{\eta}^2 \cdot C'.$$

The commitment $\hat{C}^* = \mathsf{Commit}(\mathsf{pp}, \hat{\mathsf{wit}}^*)$ is added to the context $\hat{\mathsf{ctx}}^*$.

By sampling another random value as $\hat{v_z}^*$, the simulator computes the value $\hat{\epsilon}''$ for the claim on $\hat{v_z}''$ of the masking instance as

$$\hat{\epsilon}'' = \hat{v_z}^* - \hat{\eta} \cdot \hat{\epsilon} - \hat{\eta}^2 \cdot \hat{\epsilon}',$$

where $\hat{\epsilon} = v_z, \hat{\epsilon}' = v'_z$. The $\hat{v_z}^*$ is then added to the context $\hat{\mathsf{ctx}}^*$.

Denote $\hat{\theta_j} = v_j, \hat{\theta_j}' = v'_j, \hat{\theta_j}'' = \perp$ Now, we have obtained the claims on matrices and context-witness pairs for three instances as follows

$$\begin{aligned}
&\hat{\epsilon} = \tilde{z}(\boldsymbol{r}'_y), \\
&\hat{\epsilon}' = \tilde{z}'(\boldsymbol{r}'_y), \\
&\hat{\epsilon}'' = \tilde{z}'(\boldsymbol{r}'_y), \\
&\hat{\theta_j} = \widetilde{M}_j(\boldsymbol{r}'_x, \boldsymbol{r}'_y), \forall j \in [t], \\
&\hat{\theta_j}' = \widetilde{M}'_j(\boldsymbol{r}'_x, \boldsymbol{r}'_y), \forall j \in [t].
\end{aligned}$$

Note that the matrices for making instance can be set equal to either $\{M_j\}_{j\in[t]}$ or $\{M'_j\}_{j\in[t]}$. The above values are indistinguishable from those in tr.

According to the conclusion given by Chiesa et al. in [27], the Sim can invoke another efficient simulator $\mathsf{Sim}_{\mathsf{sc}}$ to simulate an indistinguishable trace tr_2 for sum-check#2 based on the claims above.

By running the similar process as above, the Sim can simulate another indistinguishable trace tr_1 for sum-check#1 based on the claims given in tr_2.

Finally, the Sim outputs a valid trace $\hat{\mathsf{tr}}$ constructed from $\hat{\epsilon}, \hat{\epsilon}', \hat{\epsilon}'', \hat{\theta_j}, \hat{\theta_j}', \hat{\theta_j}''$ and $\mathsf{tr}_1, \mathsf{tr}_2$. Obviously, the Sim can be executed in polynomial time. □

B Security proofs of PCD scheme

We refer to the security proofs of completeness and knowledge soundness to [21].

Lemma 7 (Perfect Completeness). *Construction 3 satisfies perfect completeness.*

Proof. For public parameter pp, consider arbitrary adversarially chosen messages $(\varphi, z, z_{\mathsf{loc}}, \{z_i, \Pi_i\}_{k\in[s]})$ satisfying

$$\begin{aligned}
&\varphi \in \mathsf{F}; \varphi(z, z_{\mathsf{loc}}, \{z_k\}_{k\in[s]}) = 1; \\
&\forall k \in [s], z_k = \perp \text{ or } \mathcal{V}(\mathsf{vk}, z_k, \Pi_k) = 1,
\end{aligned}$$

such that the perfect completeness precondition is satisfied. We show that given $\Pi \leftarrow \mathcal{P}(\mathsf{pk}, z, z_{\mathsf{loc}}, \{z_k, \Pi_k\}_{k\in[s]})$, the verifier algorithm passes, i.e., $\mathcal{V}(\mathsf{vk}, z, \Pi) = 1$ with probability 1.

Specifically, there are two cases:

- If $z_k = \perp$ for all $k \in [s]$, the prover runs the algorithm honestly, and the compliance $\varphi(z, z_{\mathsf{loc}}, z_1, ..., z_s)$ holds by the preconditions. The circuit $\mathcal{R}_0$ can be constructed accordingly and a satisfied $\mathcal{R}_{\mathsf{CCCS}}$ instance is as

$$(\mathsf{s}_0, \mathsf{ctx}_0, \mathsf{wit}_0) \leftarrow \mathsf{trace}(R_0, (h, (z, z_{\mathsf{loc}}, \{z_k, \mathsf{CTX}_k, \mathsf{ctx}_k\}_{k\in[s]}, \mathsf{vk}'))$$

Then the prover sets $(\mathsf{CTX}, \mathsf{WIT}, \mathsf{pf})$ accordingly to $(\mathsf{ctx}_0, \mathsf{wit}_0, \perp)$ and computes $h = \mathsf{Hash}(\mathsf{vk}'_{mathsffs}, z, \mathsf{CTX})$. And the circuit $\mathcal{R}_1$ can be constructed accordingly and a satisfied $\mathcal{R}_{\mathsf{CCCS}}$ instance is as

$$(\mathsf{s}, \mathsf{ctx}, \mathsf{wit}) \leftarrow \mathsf{trace}(R_1, (h, (\{\mathsf{CTX}_k, \mathsf{ctx}_k\}_{k\in[s]}, \mathsf{ctx}_0, \mathsf{vk}', \mathsf{CTX}, \mathit{\Pi})).$$

Besides, $\mathsf{ctx.io} = \mathsf{H}(\mathsf{vk}', z, \mathsf{CTX})$. As a result, $\mathcal{V}(\mathsf{vk}, z, \mathit{\Pi}) = 1$ with probability 1.

– If $\exists k \in [s]$ such that $z_k \neq \perp$, by the perfect completeness precondition, $\{\mathsf{CTX}_k, \mathsf{WIT}_k\}_{k\in[s]}$ are satisfied $\mathcal{R}_{\mathsf{ACCS}}$ context-witness pairs, $\{\mathsf{ctx}_k, \mathsf{wit}_k\}_{k\in[s]}$ are satisfied $\mathcal{R}_{\mathsf{CCCS}}$ context-witness pairs, and $\mathsf{ctx}_k.\mathsf{io} = \mathsf{H}(\mathsf{vk}', z_k, \mathsf{CTX}_k)$. The prover runs the algorithm honestly, and the compliance $\varphi(z, z_{\mathsf{loc}}, z_1, ..., z_s)$ holds by the preconditions. The circuit $\mathcal{R}_0$ can be constructed accordingly and a satisfied $\mathcal{R}_{\mathsf{CCCS}}$ instance is as

$$(\mathsf{s}_0, \mathsf{ctx}_0, \mathsf{wit}_0) \leftarrow \mathsf{trace}(R_0, (h, (z, z_{\mathsf{loc}}, \{z_k, \mathsf{CTX}_k, \mathsf{ctx}_k\}_{k\in[s]}, \mathsf{vk}'))$$

Then, the prover runs the generic foldings scheme for $\{\mathsf{S}_k, \mathsf{CTX}_k, \mathsf{WIT}_k\}_{k\in[s]}$, $\{\mathsf{s}_k, \mathsf{ctx}_k, \mathsf{wit}_k\}_{k=0}^{s}$. By the perfect completeness of the generic folding scheme, we have that $(\mathsf{CTX}, \mathsf{WIT})$ is a satisfied $\mathcal{R}_{\mathsf{CCCS}}$ context-witness pair. The circuit $\mathcal{R}_1$ can be constructed accordingly and a satisfied $\mathcal{R}_{\mathsf{CCCS}}$ instance is as

$$(\mathsf{s}, \mathsf{ctx}, \mathsf{wit}) \leftarrow \mathsf{trace}(R_1, (h, (\{z_k, \mathsf{CTX}_k, \mathsf{ctx}_k\}_{k\in[s]}, \mathsf{ctx}_0, \mathsf{vk}', \mathsf{CTX}, \mathit{\Pi})).$$

Besides, $\mathsf{ctx.io} = \mathsf{H}(\mathsf{vk}', z, \mathsf{CTX})$. As a result, $\mathcal{V}(\mathsf{vk}, z, \mathit{\Pi}) = 1$ with probability 1.

In conclusion, we show that Construction 3 has perfect completeness. □

Lemma 8 (Knowledge Soundness). *Construction 3 satisfies knowledge soundness.*

Proof. Given a fixed set Z, $\mathsf{pp} \leftarrow \mathcal{G}(1^\lambda)$ and auxiliary input $\mathsf{ai} \leftarrow \mathcal{D}(\mathsf{pp})$, the polynomial time adversary $\mathcal{P}^*$ succeeds in producing valid transcript $(\varphi, \circ, \mathit{\Pi}, \mathsf{ao})$ with non-negligible probability ϵ. We aim to show that it is feasible to construct an extractor $\mathsf{Ext}_{\mathcal{P}^*}$ on input $(\mathsf{pp}, \mathsf{ai})$, succeeds in outputting $(\varphi, \mathsf{T}, \mathsf{ao})$ with probability $\epsilon - \mathrm{negl}(\lambda)$, where $\varphi \in \mathsf{F}$, $(\mathsf{pp}, \mathsf{ai}, \varphi, \mathsf{ao}) \in Z$ and T is φ-compliant.

According to [15], it is convenient to assume the transcript T as a d-depth tree, where d is the depth of φ. Among the tree T, each node u with local data $z(u)_{\mathsf{loc}}$ has a unique outgoing edge labelled with $z^{(u)}$ and a proof $\mathit{\Pi}^{(u)}$ for the correctness of $z^{(u)}$. The extractor $\mathsf{Ext}_{\mathcal{P}^*}$ is constructed inductively by constructing a sequence of extractors $\mathsf{Ext}_0, ..., \mathsf{Ext}_d$. For $i \in 0, ..., d$, Ext_i outputs a $(i+1)$-depth tree T_i. Basically, we define $\mathsf{Ext}_0(\mathsf{pp}, \mathsf{ai})$ runs $(\varphi, \circ, \mathit{\Pi}, \mathsf{ao}) \leftarrow \mathcal{P}^*(\mathsf{pp}, \mathsf{ai})$ and outputs $(\varphi, \mathsf{T}_0, \mathsf{ao})$, where T_0 contains only one node labeled with $(\circ, \mathit{\Pi})$.

Then assume we already have extractor Ext_{i-1}. To construct Ext_i, an adversary $\mathcal{P}^*_{i-1}$ for the zero-knowledge non-interactive generic folding scheme needs to be constructed first.

$\mathcal{P}^*_{i-1}(\mathsf{pp}, \mathsf{ai}, \rho)$:

```
Compute $(\varphi, \mathsf{T}_{i-1}, \mathsf{ao}) \leftarrow \mathsf{Ext}_{i-1}(\mathsf{pp}, \mathsf{ai})$. If $\mathsf{T}_{i-1}$ is not a tree of depth $i$, abort.
For each node $u \in L_{\mathsf{T}_{i-1}}(i)$, denote its label as $(z^{(u)}, \Pi_{(u)})$.
Parse $\Pi^{(u)}$ as $((\mathsf{CTX}^{(u)}, \mathsf{WIT}^{(u)}), (\mathsf{ctx}^{(u)}, \mathsf{wit}^{(u)}))$.
Obtain $(\{\mathsf{CTX}_k^{(u)}, \mathsf{ctx}_k^{(u)}, z_j^{(u)}\}_{k\in[s]}, \mathsf{ctx}_0, \mathsf{pf}^{(u)})$ from $\mathsf{wit}^{(u)}$.
Let $L_{i-1} := \{u \in L_{\mathsf{T}_{i-1}}(i) \mid \exists k \in [s], z_k^{(u)} \neq \bot\}$.
Output $\left(\left\{\{\mathsf{CTX}_k^{(u)}, \mathsf{ctx}_k^{(u)}\}_{k\in[s]}, \mathsf{ctx}_0^{(u)}, \mathsf{CTX}^{(u)}, \mathsf{WIT}^{(u)}, \mathsf{pf}^{(u)}\right\}_{u\in L_{i-1}}, (\varphi, \mathsf{T}_{i-1}, \mathsf{ao})\right)$.
```

where $L_{\mathsf{T}_{i-1}}(i)$ denotes the set of nodes of T at depth i. According to the knowledge soundness of the generic folding scheme, we can construct another extractor $\mathsf{Ext}_{\mathcal{P}^*_{i-1}}$. On input $v \in L_{i-1}$, $\mathsf{Ext}_{\mathcal{P}^*_{i-1}}$ outputs $\{\mathsf{WIT}_k^{(u)}, \mathsf{wit}_k^{(u)}\}_{k\in[s]}$ and $\mathsf{wit}_0^{(u)}$ with non-negligible probability, where $\{\mathsf{CTX}_k^{(u)}, \mathsf{WIT}_k^{(u)}\}_{k\in[s]}$ are satisfied atomic CCS context-witness pairs and $\{\mathsf{ctx}_k^{(u)}, \mathsf{wit}_k^{(u)}\}_{k=0}^{s}$ are satisfied committed CCS context-witness pairs.

Based on $\mathcal{P}^*_{i-1}, \mathsf{Ext}_{\mathcal{P}^*_{i-1}}$, we can further construct Ext_i as follows.

$(\varphi, \mathsf{T}_i, \mathsf{ao}) \leftarrow \mathsf{Ext}_i(\mathsf{pp}, \mathsf{ai})$:

```
Compute $\left(\left\{\{\mathsf{CTX}_k^{(u)}, \mathsf{WIT}_k^{(u)}\}_{k\in[s]}, \{\mathsf{ctx}_k^{(u)}, \mathsf{wit}_k^{(u)}\}_{k=0}^{s}\right\}_{u\in L_{i-1}}, (\varphi, \mathsf{T}_{i-1}, \mathsf{ao})\right)$
    $\leftarrow \mathsf{Ext}_{\mathcal{P}^*_{i-1}}(\mathsf{pp}, \mathsf{ai}, \rho)$. If $\mathsf{T}_{i-1}$ is not a tree of depth $i$, abort.
Retrieve $\{\mathsf{wit}^{(u)}\}_{u\in L_{\mathsf{T}_{i-1}}(i)}$ from the internal state of $\mathcal{P}^*_{i-1}$ and obtain $z_{\mathsf{loc}}^{(u)}, \{z_k^{(u)}\}_{k\in[s]}$
    from $\mathsf{wit}^{(u)}$.
Append $z_{\mathsf{loc}}^{(u)}$ to the label of $u \in L_{\mathsf{T}_{i-1}(i)}$.
For each node $u \in L_{i-1}$, let $L_u := \{k \in [s] \mid z_k^{(u)} \neq \bot\}$. Construct $\mathsf{T}_i$ of depth $i+1$
    from $\mathsf{T}_{i-1}$ by adding, for each node $u \in L_{i-1}$, $(z_k^{(u)}, \Pi_k^{(u)})$ to the label of its child $k \in L_u$,
    where $\Pi_k^{(u)} = \left((\mathsf{CTX}_k^{(u)}, \mathsf{WIT}_k^{(u)}), (\mathsf{ctx}_k^{(u)}, \mathsf{wit}_k^{(u)})\right)$.
Output $(\varphi, \mathsf{T}_i, \mathsf{ao})$.
```

We claim that for $i \in \{0, 1, ..., d\}$, the extractor $\mathsf{Ext}_i(\mathsf{pp}, \mathsf{ai})$ outputs $(\varphi, \mathsf{T}_i, \mathsf{ao})$ in expected polynomial time such that with probability $\epsilon - \mathrm{negl}(\lambda)$, the following conditions hold

- $\varphi \in \mathsf{F}$, $(\mathsf{pp}, \mathsf{ai}, \varphi, \circ(\mathsf{T}_i), \mathsf{ao}) \in Z$;
- T_i is φ-compliant up to depth i;
- for all $u \in L_{\mathsf{T}_i}(i+1)$, $\mathcal{V}(\mathsf{vk}, z^{(u)}, \Pi^{(u)}) = 1$.

The correctness of the above claim can be proved by induction.

- (Base case.) Since $\mathsf{Ext}_0(\mathsf{pp}, \mathsf{ai})$ runs $(\varphi, \circ, \Pi, \mathsf{ao}) \leftarrow \mathcal{P}^*(\mathsf{pp}, \mathsf{ai})$, it satisfies the conditions above.

- (Inductive hypothesis.) Assume that the extractor Ext_{i-1} satisfies the abovementioned conditions.
- (Inductive step.) Based on the hypothesis, we show that Ext_i also satisfies the conditions by the following discussion.

The inductive hypothesis ensures that Ext_{i-1} satisfies with probability $\epsilon-\mathrm{negl}(\lambda)$, that $\varphi \in \mathsf{F}$, $(\mathsf{pp}, \mathsf{ai}, \varphi, \circ(\mathsf{T}_{i-1}), \mathsf{ao}) \in Z$, T_{i-1} is φ-compliant up to the depth $i-1$, and for all $u \in L_{\mathsf{T}_{i-1}}(i)$, $\mathcal{V}(\mathsf{vk}, z^{(u)}, \Pi^{(u)}) = 1$. By the correctness of algorithm $\mathcal{V}$, we have

- (1) $\{(\mathsf{CTX}^{(u)}, \mathsf{WIT}^{(u)}), (\mathsf{ctx}^{(u)}, \mathsf{wit}^{(u)})\}_{u \in L_{\mathsf{T}_{i-1}(i)}}$ are satisfied context-witness pairs.

Since T_{i-1} is φ-compliant, by the construction of $\mathcal{R}_0, \mathcal{R}_1$ and hash function Hash, we have

- (2) for $u \in L_{\mathsf{T}_{i-1}}(i), \varphi(z^{(u)}, z^{(u)}_{\mathsf{loc}}, z^{(u)}_1, ..., z^{(u)}_s)$ accepts;
- (3) for $u \in L_{i-1}$, $\mathsf{CTX}^{(u)} = \text{zk-NIFS}.\,\mathcal{V}'(\mathsf{vk}', \{\mathsf{CTX}^{(u)}_k\}_{k\in[s]}, \{\mathsf{ctx}^{(u)}_k\}^s_{k=0}, \mathsf{pf}^{(u)})$;
- (4) for $u \in L_{i-1}$, $\mathsf{ctx}^{(u)}_k.\mathsf{io} = \mathsf{Hash}(\mathsf{vk}', z^{(u)}_k, \mathsf{CTX}^{(u)}_k)\ \forall k \in [s]$.

(2) implies that T_i is φ-compliant up to depth i and $\varphi \in \mathsf{F}, (\mathsf{pp}, \mathsf{ai}, \varphi, \circ(\mathsf{T}_i), \mathsf{ao}) \in Z$. (1) and (3) imply that there exists efficient construction of $\mathcal{P}^*_{i-1}$ that succeeds in producing folded pairs $\{\mathsf{CTX}^{(u)}, \mathsf{WIT}^{(u)}\}_{u\in L_{i-1}}$ with probability $\epsilon - \mathrm{negl}(\lambda)$. Then there exists an efficient extractor $\mathsf{Ext}_{\mathcal{P}^*_{i-1}}$ outputting $\{\{\mathsf{WIT}^{(u)}_k\}_{k\in[s]}, \{\mathsf{wit}^{(u)}_k\}^s_{k=0}\}_{u\in L_{i-1}}$ guaranteed by the knowledge soundness of generic foldings scheme. (1)-(4) imply that $\mathcal{V}(\mathsf{vk}, z^{(u)}, \Pi^{(u)}) = 1$ holds for all $u \in L_{(T)_i}(i+1)$. Therefore, the hypothesis for Ext_i also holds.

In conclusion, we prove that Construction 3 is knowledge-sound. □

Lemma 9 (Zero Knowledge). *Construction 3 satisfies zero knowledge.*

Proof. We prove that PCD scheme is zero-knowledge by constructing a probabilistic polynomial-time simulator Sim as

```
Sim(1^λ):
Compute $(\mathsf{pp}_{\mathsf{fs}}, \tau_{\mathsf{fs}}) \leftarrow \mathsf{Sim}_{\mathsf{fs}}(1^\lambda)$.
Output $(\mathsf{pp} = \mathsf{pp}_{\mathsf{fs}}, \tau = \tau_{\mathsf{fs}})$.
```

```
Sim(pp, φ, z, τ):
Obtain $\{\mathsf{S}_k, \mathsf{CTX}_k\}_{k\in[s]}$, $\{\mathsf{s}_k, \mathsf{ctx}_k\}^s_{k=0}$ from public $\mathcal{R}_1$.
Compute $(\mathsf{S}, \mathsf{CTX}, \mathsf{WIT}, \mathsf{pf}) \leftarrow \mathsf{Sim}_{\mathsf{fs}}(\mathsf{pp}_{\mathsf{fs}}, \{\mathsf{S}^{(k)}, \mathsf{CTX}^{(k)}\}_{k\in[s]}, \{\mathsf{s}^{(k)}, \mathsf{ctx}^{(k)}\}^s_{k=0}, \tau)$.
Compute $h \leftarrow \mathsf{Hash}(\mathsf{vk}'_{\mathsf{fs}}, z, \mathsf{CTX})$.
Output $(\mathsf{s}, \mathsf{ctx}, \mathsf{wit}) \leftarrow \mathsf{trace}(\mathcal{R}_1, (h, (\{\mathsf{CTX}_k, \mathsf{ctx}_k\}_{k\in[s]}, \mathsf{ctx}_0, \mathsf{vk}'_{\mathsf{fs}}, \mathsf{CTX}, \mathsf{pf})))$.
```

□