Nova: Recursive Zero-Knowledge Arguments from Folding Schemes

Abhiram Kothapalli[⋆†] Srinath Setty[⋆] Ioanna Tzialla[‡]

[⋆]Microsoft Research [†]Carnegie Mellon University [‡]New York University

Abstract. We introduce *folding schemes* for NP, an interactive protocol between a prover and a verifier to combine two N-sized NP instances over a finite field $\mathbb{F}$ into a single N-sized instance such that the folded instance is satisfiable only if the original instances are satisfiable. In particular, we devise a folding scheme for *relaxed R1CS*, a characterization of NP that is especially amenable to folding. The verifier's cost and the communication in the folding scheme are both $O_\lambda(1)$, where λ is the security parameter, assuming any additively-homomorphic commitment scheme that provides $O_\lambda(1)$-sized commitments to N-sized vectors over $\mathbb{F}$. Additionally, the protocol is honest-verifier zero-knowledge and public coin, so it can be made non-interactive in the ROM using the Fiat-Shamir transform.

We then construct incrementally verifiable computation (IVC) from folding schemes by using a "verifier circuit" that at each recursive step folds an entire R1CS instance representing computation (including a copy of the verifier circuit) at its prior step into a running relaxed R1CS instance. A distinctive aspect of our approach to IVC is that it achieves the smallest verifier circuit (a key metric to minimize in IVC) in the literature: the circuit is *constant-sized* and its size is dominated by *two group scalar multiplications*. We then show that the running relaxed R1CS instance can be proven in zero-knowledge with a succinct proof using a variant of an existing zkSNARK.

Putting these together, we obtain Nova, a new zero-knowledge proof system for incremental computations, where for an N-sized computation with C-sized steps, the prover runs in $O_\lambda(N)$ time to produce $O_\lambda(\log C)$-sized proofs that can be verified in $O_\lambda(C)$ time. Nova does not require a trusted setup nor performs FFTs, so it can be efficiently instantiated with *any* cycles of elliptic curves where DLOG is hard. Furthermore, at each step, the prover time is dominated by two $\approx C$-sized multiexponentiations. Finally, Nova can achieve $O_\lambda(\log C)$ verification time at the cost of employing a pairing-friendly elliptic curve where SXDH is hard.

1 Introduction

This paper introduces a new cryptographic primitive called *folding schemes*. A folding scheme is defined with respect to an NP relation, and it is an interactive protocol between an untrusted *prover* and a *verifier*. Both entities hold two N-sized NP instances, and the prover in addition holds purported witnesses for both instances. The protocol enables the prover and the verifier to output a single N-sized NP instance, which we refer to as a *folded instance*. Furthermore, the prover privately outputs a purported witness to the folded instance using purported witnesses for the original instances. Informally, a folding scheme guarantees that the folded instance is satisfiable only if the original instances are satisfiable. A folding scheme is said to be *non-trivial* if the verifier's costs and the communication are lower in the case the verifier participates in the folding scheme and then verifies an NP witness (provided by the prover) for the folded instance than the case where the verifier verifies NP witnesses (provided by the prover) for each of the original instances.

For a concrete instantiation of folding schemes, we first introduce a new characterization of NP, called *relaxed R1CS*, a small variant of R1CS, which is itself an NP-complete language that generalizes arithmetic circuit satisfiability. We then show that there exists a constant-round folding scheme for relaxed R1CS, where for N-sized relaxed R1CS instances over a finite field $\mathbb{F}$ with the same "structure" (i.e., R1CS coefficient matrices), the prover's work is $O_\lambda(N)$, and the verifier time and the communication are both $O_\lambda(1)$, 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. In addition, the folding scheme for relaxed R1CS satisfies honest-verifier zero-knowledge. Furthermore, it is public-coin, so it can be made non-interactive in the random oracle model using the Fiat-Shamir transform [FS86], and be instantiated (heuristically) in the plain model using a concrete hash function.

Relationship with Existing Techniques. A folding scheme is reminiscent of the folding technique used in protocols such as Bulletproofs [BBB+18], where the prover splits an N-sized inner product instance into two $N/2$-sized inner product instances, and then the prover and the verifier interactively combine the two $N/2$-sized instances into a single $N/2$-sized instance. A folding scheme differs from this technique in that it is defined with respect to an NP relation. Furthermore, our folding scheme for relaxed R1CS folds arbitrary NP instances.

A folding scheme is also reminiscent of the sum-check protocol [LFKN90] applied to layered data-parallel circuits [WJB+17, WTS+18], where the prover and the verifier interactively combine—using the sum-check protocol—$\beta \geq 2$ instances of depth-1 circuit satisfiability into a single depth-1 circuit satisfiability instance, and the combined instance is proven using additional invocations of the sum-check protocol. Once the protocol combines β instances of depth-1 circuit satisfiability instances into a single depth-1 circuit satisfiability instance, it is not entirely clear how to fold additional circuit satisfiability instances into a single running instance. Whereas, with a folding scheme, the prover and the verifier can continually fold N-sized NP instances into a single running N-sized instance.

1.1 Recursive Zero-Knowledge Arguments from Folding Schemes

A primary application of folding schemes is to construct recursive zero-knowledge argument systems for NP. In particular, we show how our folding scheme for relaxed R1CS can be used to construct *incrementally-verifiable computation (IVC)* [Val08].[1] We refer to our construction as Nova. Such recursive argument systems have a wide variety of applications such as constructing VDFs [Wes19, BBF18], succinct blockchains [BGH20, Lab20], incrementally-verifiable versions of verifiable state machines [SAGL18, LNS20], etc.

Recall that an IVC is an argument system [Kil92, Mic94][2] for incremental computations of the form $y = F^{(\ell)}(x)$, where F is a (possibly non-deterministic) computation, $\ell > 0$, x is a public input, and y is the public output. At each incremental step, the IVC prover produces a proof that the step was computed correctly *and* it has verified a proof for the prior step. In other words, at each incremental step, the IVC prover produces a proof of satisfiability for an augmented circuit that augments the circuit for F with a "verifier circuit" that verifies the proof of the prior step. Recursively, the final proof proves the correctness of the entire incremental computation. A key aspect of IVC is that the IVC verifier's work nor the IVC proof size depend on the number of steps in the incremental computation. In particular, the IVC verifier only verifies the proof produced at the last step of the incremental computation.

In Nova, we consider incremental computations, where each step of the incremental computation is expressed with R1CS (all the steps in the incremental computation share the same R1CS coefficient matrices). At step i of the incremental computation, as in other approaches to IVC, Nova's prover proves that the step i was computed correctly. Furthermore, at step i, instead of verifying a proof for step $i-1$ (as in traditional approaches to IVC), Nova's approach treats the computation at step $i-1$ as an R1CS instance and folds that into a running relaxed R1CS instance. Specifically, at each step, Nova's prover proves that it has performed the step's computation and has folded its prior step represented as an R1CS instance into a running relaxed R1CS instance. In other words, the circuit satisfiability instance that the prover proves at each incremental step computes a step of the incremental computation and includes a circuit for the computation of the verifier in the non-interactive folding scheme for relaxed R1CS. After the last step of the incremental computation, Nova's prover proves the running relaxed R1CS instance in zero-knowledge and with an $O_\lambda(\log|F|)$-sized succinct proof using Spartan [Set20] adapted to prove relaxed R1CS instances.

A distinctive aspect of Nova's approach to IVC is that it achieves the smallest "verifier circuit" in the literature. Since the verifier's costs in the folding scheme for relaxed R1CS is $O_\lambda(1)$, the size of the computation that Nova's prover proves at each incremental step is $\approx|F|$, assuming N-sized vectors are committed with an $O_\lambda(1)$-sized commitments (e.g., Pedersen's commitments in a cryptographic group where DLOG is hard). In particular, the verifier circuit in Nova is constant-sized and its size is dominated by two *group scalar multiplications*. Furthermore,

[1] Folding schemes can also be used to construct a generalization of IVC called proof-carrying data (PCD) [AC10], but we focus on IVC for simplicity.

[2] An *argument system of knowledge* for circuit satisfiability enables an untrusted polynomial-time prover to prove to a verifier the knowledge of a witness w such that $\mathcal{C}(w, x) = y$, where $\mathcal{C}$ is a circuit, x is some public input, and y is some public output.

Nova's prover's work at each step is dominated by two multiexponentiations of size $\approx|F|$. Nova's prover does not perform any FFTs either at each incremental step or when using the Spartan variant, so it can be instantiated efficiently using *any* cycles of elliptic curves where DLOG is hard.

Comparison with existing approaches to IVC and PCD. Figure 1 depicts Nova's costs and compares it with prior approaches. We now elaborate.

A well-known approach to construct IVC is to use SNARKs for NP [BCCT13, BCTV14]: at each incremental step i, the prover produces a SNARK that it applied F to the output of step $i-1$ and it has verified a SNARK from step $i-1$. Although earlier SNARKs [GGPR13, PGHR13] require a trusted setup, recent work [Set20, BFS20, COS20] eliminates the need for a trusted setup. Thus, one can construct IVC or PCD without trusted setup by relying on such SNARKs. Unfortunately, SNARK-based IVC requires representing an entire SNARK verifier as a circuit at each step of the incremental computation.

Halo [BGH19] provides an innovative approach to construct IVC without trusted setup (this approach was later formalized by Bünz et al. [BCMS20]). In Halo, the "verifier circuit" postpones certain expensive computations (i.e., verifying polynomial evaluation proofs) in its SNARK verifier.[3] As a result, Halo provides a verifier circuit size that is concretely smaller than what one can obtain from an approach based on SNARKs without trusted setup. Nova's approach can be viewed as taking the approach in Halo to the extreme. Specifically:

– At each incremental step, Halo's verifier circuit verifies a "partial" SNARK. This still requires Halo's prover to perform $|F|$-sized FFTs and $O(|F|)$ exponentiations (i.e., *not* an $|F|$-sized multiexponentiation). Whereas, in Nova, the verifier circuit folds an entire NP instance representing computation at the prior step into a running relaxed R1CS instance. This only requires the Nova's prover to commit to a satisfying assignment of an $\approx|F|$-sized circuit (which computes F and performs the verifier's computation in a folding scheme for relaxed R1CS), so at each step, Nova's prover only computes an $O(|F|)$-sized multiexponentiation and does not compute any FFTs. So, Nova's prover incurs lower costs than Halo's prover, both asymptotically and concretely.
– The verifier circuit in Halo is of size $O_\lambda(\log|F|)$ whereas in Nova, it is $O_\lambda(1)$. Concretely, the dominant operations in Halo's circuit is $O(\log|F|)$ group scalar multiplications, whereas in Nova, it is two group scalar multiplications.
– Halo and Nova have the same proof sizes $O_\lambda(\log|F|)$ and verifier time $O_\lambda(|F|)$.

Halo Infinite [BDFG20] generalizes Halo [BGH19] to other polynomial commitment schemes by defining certain abstract properties of polynomial commitments.

Bünz et al. [BCL$^+$20] propose a variant of the approach in Halo, where they realize PCD (and hence IVC) without relying on succinct arguments. Specifically, they first devise a non-interactive argument of knowledge (NARK) for R1CS with

[3] In Halo, verifying a polynomial evaluation proof takes $O_\lambda(N)$ time, where in the context of IVC, $N=O(|F|)$. In other words, Halo does not provide a SNARK with sub-linear verification, so without postponement, it does not lead to IVC.

	Prover (each step)	Proof size	Verifier	"Verifier circuit" (dominant ops)	assumption
[BCTV14] with [Gro16]†	$O(C)$ FFT $O(C)$ MSM	$O_\lambda(1)$	$O_\lambda(1)$	3 $\mathbb{P}$	q-type
[Set20]	$O(C)$ MSM	$O_\lambda(\sqrt{C})$	$O_\lambda(\sqrt{C})$	$O(\sqrt{C})$ $\mathbb{G}$	DLOG
[COS20]	$O(C)$ FFT $O(C)$ MHT	$O_\lambda(\log^2 C)$	$O_\lambda(\log^2 C)$	$O_\lambda(\log^2 C)$ $\mathbb{F}$ $O_\lambda(\log^2 C)$ $\mathbb{H}$	CRHF
[BGH19]	$O(C)$ FFT $O(C)$ EXP	$O_\lambda(\log C)$	$O_\lambda(C)$	$O(\log C)$ $\mathbb{G}$	DLOG
[BCL$^+$20]*	$O(C)$ FFT $O(C)$ MSM	$O_\lambda(C)$	$O_\lambda(C)$	8 $\mathbb{G}$	DLOG
Nova (this work)	$O(C)$ MSM	$O_\lambda(\log C)$	$O_\lambda(C)$	2 $\mathbb{G}$	DLOG
Nova (this work)	$O(C)$ MSM	$O_\lambda(\log C)$	$O_\lambda(\log C)$	2 $\mathbb{G}_T$	SXDH

† Requires per-circuit trusted setup
* In an update concurrent with our work, they avoid FFTs and improve the verifier circuit's size by $\approx 2\times$
$O(C)$ FFT: FFT over an $O(C)$-sized vector costing $O(C \log C)$ operations over $\mathbb{F}$
$O(C)$ MHT: Merkle tree over an $O(C)$-sized vector costing $O(C)$ hash computations
$O(C)$ EXP: $O(C)$ exponentiations in a cryptographic group
$O(C)$ MSM: $O(C)$-sized multi-exponentiation in a cryptographic group

Fig. 1. Asymptotic costs of Nova and its baselines to produce and verify a proof for an incremental computation where each incremental step applies a function F. C denotes the size of the computation at each incremental step i.e., $|F| + |\mathcal{C}_\mathcal{V}|$, where $\mathcal{C}_\mathcal{V}$ is the "verifier circuit" in IVC. For the "verifier circuit" column, we depict the number of dominant operations in a circuit that verifies a proof produced by a prior step in an incremental computation. $\mathbb{P}$ denotes a pairing in a pairing-friendly group, $\mathbb{F}$ denotes the number of finite field operations, $\mathbb{H}$ denotes a hash computation, and $\mathbb{G}$ denotes a scalar multiplication in a cryptographic group.

$O_\lambda(N)$-sized proofs and $O_\lambda(N)$ verification times for N-sized R1CS instances. Then, they show that most of the NARK's verifier's computation can be postponed by performing $O_\lambda(1)$ work in the verifier circuit. Nova is inspired by their work, but instead of devising a NARK for R1CS and then postpone the NARK's verifier's work, we introduce folding schemes for NP and devise a folding scheme for relaxed R1CS, which directly leads to IVC. For zero-knowledge, Nova relies on zero-knowledge arguments with succinct proofs, whereas their approach does not rely on succinct arguments. However, Nova's approach has several advantages.

– At each step, their prover performs an $O(|F|)$-sized FFT (which costs $O(|F| \log |F|)$ operations over $\mathbb{F}$). Whereas, Nova does not perform any FFTs.
– Their prover's work for multiexponentitions at each step and the size of their verifier circuit are both higher than in Nova by $\approx 4\times$.
– Proof sizes are $O_\lambda(|F|)$ in their work, whereas in Nova, they are $O_\lambda(\log |F|)$.

In an update concurrent with this work, Bünz et al. [BCL+20] provide an improved construction of their NARK for R1CS, which leads to an IVC that, like Nova, avoids FFTs. Furthermore, they improve the size of the verifier circuit by $\approx 2\times$, which is still larger than Nova's verifier circuit by $\approx 2\times$. The per-step computation of the prover remains concretely higher than Nova.

2 Preliminaries

Let $\mathbb{F}$ denote a finite field with $|\mathbb{F}| = 2^{\Theta(\lambda)}$, where λ is the security parameter.

2.1 A Constraint System to Capture NP

To define folding schemes and incrementally verifiable computation for any instance in NP, we need an NP-complete language. Our starting point is R1CS, a linear algebraic formulation of quadratic arithmetic programs (QAPs) [GGPR13, PGHR13] that is known to capture any instance in NP. R1CS is used with and without QAPs in subsequent works [SBV+13, BCR+19, Set20, COS20]. Later, in Section 4, we show how to extend R1CS so that it can be "folded".

Definition 1 (R1CS). *An R1CS instance is a tuple* $((\mathbb{F}, A, B, C, m, n, \ell), \mathsf{x})$, *where* $\mathsf{x} \in \mathbb{F}^{\ell}$ *is the public input and output of the instance,* $A, B, C \in \mathbb{F}^{m \times m}$, $m \geq |\mathsf{x}| + 1$, *and there are at most* $n = \Omega(m)$ *non-zero entries in each matrix. A witness vector* $W \in \mathbb{F}^{m-\ell-1}$ *satisfies an R1CS instance* $((\mathbb{F}, A, B, C, m, n, \ell), \mathsf{x})$ *if* $(A \cdot Z) \circ (B \cdot Z) = C \cdot Z$, *where* $Z = (W, \mathsf{x}, 1)$.

WLOG, we assume that m and n are powers of 2 and that $m = \ell + 1$.

2.2 A Commitment Scheme for Vectors over $\mathbb{F}$

We require an additively homomorphic and succinct commitment scheme for vectors over $\mathbb{F}$. We define commitment schemes and required properties below:

Definition 2 (Commitment Scheme). *A commitment scheme for* $\mathbb{F}^m$ *is a tuple of three protocols with the following syntax that satisfy the two properties listed below:*

- $\mathsf{pp} \leftarrow \mathsf{Gen}(1^\lambda, m)$*: produces public parameters* pp.
- $C \leftarrow \mathsf{Com}(\mathsf{pp}, x, r)$*: takes as input* $x \in \mathbb{F}^m$ *and* $r \in \mathbb{F}$*; produces a public commitment* C.
- $b \leftarrow \mathsf{Open}(\mathsf{pp}, C, x, r)$*: verifies the opening of commitment* C *to* $x \in \mathbb{F}^m$ *and* $r \in \mathbb{F}$*; outputs* $b \in \{0, 1\}$.

(1) Binding. *For any PPT adversary* $\mathcal{A}$,

$$\Pr\left[\begin{array}{l|l} b_0 = b_1 = 1, & \mathsf{pp} \leftarrow \mathsf{Gen}(1^\lambda, m), \\ x_0 \neq x_1 & (C, x_0 \in \mathbb{F}^m, x_1 \in \mathbb{F}^m, r_0 \in \mathbb{F}, r_1 \in \mathbb{F}) \leftarrow \mathcal{A}(\mathsf{pp}), \\ & b_0 \leftarrow \mathsf{Open}(\mathsf{pp}, C, x_0, r_0), \\ & b_1 \leftarrow \mathsf{Open}(\mathsf{pp}, C, x_1, r_1) \end{array}\right] \leq \mathsf{negl}(\lambda).$$

(2) ***Hiding.*** *For all PPT adversaries $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$:*

$$\left| \frac{1}{2} - \Pr\left[b = \bar{b} \middle| \begin{array}{l} (x_0, x_1, \mathsf{st}) \leftarrow \mathcal{A}_0(\mathsf{pp}), \\ b \leftarrow_R \{0,1\}, r \leftarrow_R \mathbb{F}, \\ C \leftarrow \mathsf{Com}(\mathsf{pp}, x_b, r), \\ \bar{b} \leftarrow \mathcal{A}_1(\mathsf{st}, C) \end{array} \right] \right| \leq \mathsf{negl}(\lambda).$$

If hiding holds for all adversaries, then the commitment is statistically hiding.

Definition 3 (Additively Homomorphic). *A commitment scheme* $(\mathsf{Gen}, \mathsf{Com}, \mathsf{Open})$ *for vectors over* $\mathbb{F}^m$ *is additively homomorphic if for all public parameters* pp *produced from* $\mathsf{Gen}(1^\lambda, m)$*, and for any* $x_1, x_2 \in \mathbb{F}^m$ *and for any* $r_1, r_2 \in \mathbb{F}$*,* $\mathsf{Com}(\mathsf{pp}, x_1, r_1) + \mathsf{Com}(\mathsf{pp}, x_2, r_2) = \mathsf{Com}(\mathsf{pp}, x_1 + x_2, r_1 + r_2)$.

Definition 4 (Succinctness). *A commitment scheme* $(\mathsf{Gen}, \mathsf{Com}, \mathsf{Open})$ *for vectors over* $\mathbb{F}^m$ *provides succinct commitments if for all public parameters* pp *produced from* $\mathsf{Gen}(1^\lambda, m)$*, and any* $x \in \mathbb{F}^m$ *and* $r \in \mathbb{F}$*,* $|\mathsf{Com}(\mathsf{pp}, x, r)| = O_\lambda(1)$.

Construction 1 (Pedersen Commitment). For group $\mathbb{G}$ such that $|\mathbb{G}| \geq 2^\lambda$ with scalar field $\mathbb{F}$ and where the discrete logarithm problem is hard, the Pedersen commitment scheme for vectors over $\mathbb{F}$ is defined as follows:

- $\mathsf{pp} \leftarrow \mathsf{Gen}(1^\lambda, m)$: Sample $g \leftarrow_R \mathbb{G}^m$, $h \leftarrow_R \mathbb{G}$. Output (g, h).
- $C \leftarrow \mathsf{Com}(\mathsf{pp}, x \in \mathbb{F}^m, r \in \mathbb{F})$: Output $h^r \cdot \prod_{i \in \{0,\ldots,m\}} g_i^{x_i}$.

Lemma 1 (Pedersen Commitment). *The Pedersen commitment scheme is hiding, binding, additively homomorphic, and succinct.*

2.3 Incrementally Verifiable Computation

Incrementally verifiable computation (IVC) [Val08] enables efficient verifiable computation for repeated function application. Intuitively, for function F that takes non-deterministic input and initial input z_0, an IVC scheme allows a prover to demonstrate that $z_n = \mathsf{F}^{(n)}(z_0)$ (i.e. n repeated applications of F on input z_0) for some final output z_n and application count n. We define IVC using notational conventions of modern argument systems.

Definition 5 (Incrementally Verifiable Computation). *An incrementally verifiable computation (IVC) scheme is defined by PPT algorithms $\mathcal{P}, \mathcal{V}$ denoted the prover and verifier. An IVC scheme $(\mathcal{P}, \mathcal{V})$ satisfies perfect completeness if for any polynomial-time computable function* F*, application count i, initial input z_0, output z_i, and for any intermediate values z_{i-1}, ω_{i-1}, and proof Π_{i-1}*

$$\Pr\left[\mathcal{V}(\mathsf{F}, i, z_0, z_i, \Pi_i) = 1 \middle| \begin{array}{l} z_i = \mathsf{F}(z_{i-1}, \omega_{i-1}), \\ \mathcal{V}(\mathsf{F}, i-1, z_0, z_{i-1}, \Pi_{i-1}) = 1, \\ \Pi_i \leftarrow \mathcal{P}(\mathsf{F}, i, z_0, z_i; z_{i-1}, \omega_{i-1}, \Pi_{i-1}) \end{array} \right] = 1.$$

Likewise, an IVC scheme satisfies knowledge-soundness if for all PPT adversaries $\mathcal{P}^*$, there exists PPT extractor $\mathcal{E}$ such that for all $(\mathsf{F}, n, z_0, z_n)$, and input randomness ρ

$$\Pr\left[z_i = \mathsf{F}(z_{i-1}, \omega_{i-1}) \quad \forall i \in \{1, \dots, n\} \,\middle|\, \begin{array}{c}(z_1, \dots, z_{n-1}) \\ (\omega_0, \dots, \omega_{n-1})\end{array} \leftarrow \mathcal{E}(\mathsf{F}, n, z_0, z_n; \rho) \right] \geq$$
$$\Pr\left[\mathcal{V}(\mathsf{F}, z_0, z_n, n, \Pi) = 1 \,\middle|\, \Pi \leftarrow \mathcal{P}^*(\mathsf{F}, n, z_0, z_n; \rho) \right] - \mathsf{negl}(\lambda).$$

Definition 6 (Zero-Knowledge). *An IVC scheme $(\mathcal{P}, \mathcal{V})$ satisfies zero-knowledge if there exists PPT simulator $\mathcal{S}$ such that for any polynomial-time computable function F, application count i, initial input z_0, output z_i, and for any intermediate values z_{i-1}, ω_{i-1}, and proof Π_{i-1}, and for all PPT adversaries $\mathcal{A}$*

$$\left| \begin{array}{l} \Pr\left[\mathcal{A}(\Pi) = 1 \,\middle|\, \begin{array}{l} z_i = \mathsf{F}(z_{i-1}, \omega_{i-1}), \\ \mathcal{V}(\mathsf{F}, i-1, z_0, z_{i-1}, \Pi_{i-1}) = 1, \\ \Pi \leftarrow \mathcal{P}(\mathsf{F}, i, z_0, z_i; z_{i-1}, \omega_{i-1}, \Pi_{i-1}) \end{array} \right] - \\ \Pr\left[\mathcal{A}(\Pi) = 1 \,\middle|\, \begin{array}{l} z_i = \mathsf{F}(z_{i-1}, \omega_{i-1}), \\ \Pi \leftarrow \mathcal{S}(\mathsf{F}, i, z_0, z_i) \end{array} \right] \end{array} \right| \leq \mathsf{negl}(\lambda).$$

3 Folding Schemes

This section formally introduces folding schemes. Intuitively, a folding scheme is an interactive protocol that reduces a verifier's task of checking two NP instances into the task of checking a single NP instance.

Definition 7 (Folding Scheme). *Consider binary relation $\mathcal{R}$ over instance-witness tuples. A folding scheme for $\mathcal{R}$ consists of a pair of interactive algorithms $\mathcal{P}$ and $\mathcal{V}$, denoted the prover and verifier respectively, with the following structure*

- $\mathcal{P}((u_1, w_1), (u_2, w_2)) \rightarrow (u, w)$: *On input instance-witness tuples (u_1, w_1) and (u_2, w_2) where each instance is N-sized, outputs a new instance-witness tuple (u, w) where the instance is N-sized.*
- $\mathcal{V}(u_1, u_2) \rightarrow u$: *On input two N-sized instances u_1 and u_2, outputs a new N-sized instance u.*

Let

$$(u, w) \leftarrow \langle \mathcal{P}(w_1, w_2), \mathcal{V} \rangle (u_1, u_2)$$

denote the the verifier's output instance u and the prover's output witness w from the interaction of $\mathcal{P}$ and $\mathcal{V}$ on witness input w_1, w_2 and instance input u_1, u_2. Likewise, let

$$\mathsf{tr} = \langle \mathcal{P}(w_1, w_2), \mathcal{V} \rangle (u_1, u_2)$$

denote the corresponding interaction transcript. A folding scheme satisfies perfect completeness if for all valid instance-witness tuples $(u_1, w_1) \in \mathcal{R}$ and $(u_2, w_2) \in \mathcal{R}$

$$\Pr\left[(u, w) \in \mathcal{R} \,\middle|\, (u, w) \leftarrow \langle \mathcal{P}(w_1, w_2), \mathcal{V} \rangle (u_1, u_2) \right] = 1.$$

We call a transcript accepting if $\mathcal{P}$ outputs a satisfying folded witness w for the folded instance u. We consider a folding scheme non-trivial if the communication costs and the verifier's computation are lower in the case the verifier participates in the folding scheme and then verifies an NP witness (provided by the prover) for the folded instance (using an NP checker) than the case where the verifier verifies NP witnesses (provided by the prover) for each of the original instances (using an NP checker).

Definition 8 (Knowledge Soundness). *A folding scheme satisfies knowledge soundness if for each PPT adversary $\mathcal{P}^*$ there exists a PPT extractor $\mathcal{E}$ such that for all instances u_1, u_2*

$$\begin{aligned}&\Pr\left[\,(u_1,w_1)\in\mathcal{R}\wedge(u_2,w_2)\in\mathcal{R}\,\middle|\,(w_1,w_2)\leftarrow\mathcal{E}(u_1,u_2,\rho)\,\right]\geq\\&\Pr\left[\,(u,w)\in\mathcal{R}\,\middle|\,(u,w)\leftarrow\langle\mathcal{P}^*(\rho),\mathcal{V}\rangle(u_1,u_2)\,\right]-\mathsf{negl}(\lambda)\end{aligned}$$

where ρ denotes the input randomness for $\mathcal{P}^*$.

Definition 9 (Zero-Knowledge). *A folding scheme satisfies zero-knowledge for relation $\mathcal{R}$ if there exists PPT simulator $\mathcal{S}$ such that for all valid instance-witness tuples $(u_1,w_1),(u_2,w_2)\in\mathcal{R}$, PPT adversaries $\mathcal{V}^*$, PPT adversaries $\mathcal{A}$, and input randomness ρ*

$$\left|\begin{array}{l}\Pr\left[\,\mathcal{A}(\mathsf{tr})=1\,\middle|\,\mathsf{tr}=\langle\mathcal{P}(w_1,w_2),\mathcal{V}^*(\rho)\rangle(u_1,u_2)\,\right]-\\\Pr\left[\,\mathcal{A}(\mathsf{tr})=1\,\middle|\,\mathsf{tr}\leftarrow\mathcal{S}(u_1,u_2,\rho)\,\right]\end{array}\right|\leq\mathsf{negl}(\lambda).$$

A folding scheme is considered to be honest-verifier zero-knowledge if $\mathcal{V}^*$ is constrained to be the honest verifier.

Definition 10 (Public Coin). *A folding scheme $(\mathcal{P},\mathcal{V})$ is called public coin if all the messages sent from $\mathcal{V}$ to $\mathcal{P}$ are chosen uniformly at random and independently of the prover's messages.*

Typically, knowledge soundness is difficult to prove directly. To assist our proofs, we introduce a variant of the forking lemma [BCC+16] which abstracts away much of the probabilistic reasoning.

Theorem 1 (Forking Lemma for Folding Schemes). *Consider $(2\mu+1)$-move folding scheme $(\mathcal{P},\mathcal{V})$. $(\mathcal{P},\mathcal{V})$ satisfies knowledge soundness if there exists PPT $\mathcal{X}$ such that for all input instance pairs u_1, u_2, outputs satisfying witnesses w_1, w_2 with probability $1-\mathsf{negl}(\lambda)$, given an $(n_1,\ldots,n_\mu)$-tree of accepting transcripts and corresponding folded instance-witness pairs (u,w). This tree comprises n_1 transcripts (and corresponding instance-witness pairs) with fresh randomness in $\mathcal{V}$'s first message; and for each such transcript, n_2 transcripts (and corresponding instance-witness pairs) with fresh randomness in $\mathcal{V}$'s second message; etc., for a total of $\prod_{i=1}^{\mu} n_i$ leaves bounded by $\mathsf{poly}(\lambda)$.*

Proof. A proof for our variant of the forking lemma is similar to that of Bootle et al. [BCC+16]. For completeness, we present a formal proof in Appendix A. □

4 (Committed) Relaxed R1CS

In this work, we consider a variant of R1CS that we refer to as *relaxed R1CS*. As we show in section 5, relaxed R1CS (more specifically its committed variant) is designed to be a characterization of NP that can be easily folded.

Definition 11 (Relaxed R1CS). *A relaxed R1CS instance is a tuple* $((\mathbb{F}, A, B, C, E, u, m, n, \ell), \mathsf{x})$*, where* $\mathsf{x} \in \mathbb{F}^{\ell}$ *is the public input and output of the instance,* $E \in \mathbb{F}^m$*,* $u \in \mathbb{F}$*,* $A, B, C \in \mathbb{F}^{m \times m}$*,* $m \geq |\mathsf{x}| + 1$*, and there are at most* $n = \Omega(m)$ *non-zero entries in each matrix. A witness vector* $W \in \mathbb{F}^{m-\ell-1}$ *satisfies a relaxed R1CS instance* $((\mathbb{F}, A, B, C, E, u, m, n, \ell), \mathsf{x})$ *if* $(A \cdot Z) \circ (B \cdot Z) = u \cdot (C \cdot Z) + E$*, where* $Z = (W, \mathsf{x}, 1)$*.*

We observe that an R1CS instance can be expressed as a relaxed R1CS instance and vice versa. This not only shows that relaxed R1CS characterizes NP, but also that we can (almost) generically repurpose an argument system for R1CS [Set20, CHM+20] to be an argument system for relaxed R1CS (Section 7).

Reduction 1 (From R1CS to Relaxed R1CS). Consider an R1CS instance $\phi = ((\mathbb{F}, A, B, C, m, n, \ell), \mathsf{x})$. We can create a relaxed R1CS instance ϕ' such that witness $W \in \mathbb{F}^{m-\ell-1}$ satisfies ϕ' if and only if it satisfies ϕ. In particular, we set scalar $u = 1$ and error vector E to be the zero vector with dimension m. We set the relaxed R1CS instance to be: $\phi' = ((\mathbb{F}, A, B, C, E, u, m, n, \ell), \mathsf{x})$.

Reduction 2 (From Relaxed R1CS back to R1CS). Consider a relaxed R1CS instance $\phi = ((\mathbb{F}, A, B, C, E, u, m, n, \ell), \mathsf{x})$, we can create R1CS instance ϕ' such that a witness W satisfies ϕ' if and only if it satisfies ϕ. In particular, let $\mathbf{0}$ be the zero matrix with dimensions $m \times (m-1)$. We first set $C' = u \cdot C + (\mathbf{0}, E)$, and then set $\phi' = ((\mathbb{F}, A, B, C', m, n, \ell), \mathsf{x})$.

Proof. Let $\mathbf{0}$ be the zero matrix with dimensions $m \times (m-1)$. Because the last element in Z is 1 we have that

$$(\mathbf{0}, E) \cdot Z = E$$

Next we observe that

$$\begin{aligned} u \cdot CZ + E &= u \cdot C \cdot Z + (\mathbf{0}, E) \cdot Z \\ &= (u \cdot C + (\mathbf{0}, E)) \cdot Z \\ &= C'Z \end{aligned}$$

Thus

$$C'Z = AZ \circ BZ$$

holds if and only if

$$u \cdot CZ + E = AZ \circ BZ.$$

□

We design a variant of relaxed R1CS, called committed relaxed R1CS, which represents a part of the instance with (succinct) commitments rather than in plaintext. This enables us to devise a folding scheme with $O_\lambda(1)$ communication and $O_\lambda(1)$ work for the verifier. We will later show how this helps achieve a zero-knowledge folding scheme.

Definition 12 (Committed Relaxed R1CS). *A committed relaxed R1CS instance is a tuple* $((\mathbb{F}, A, B, C, m, n, \ell), (\overline{E}, u, \overline{W}, \mathsf{x}))$*, defined with respect to two sets of public parameters* $(\mathsf{pp}_E, \mathsf{pp}_W)$ *produced from a commitment scheme* $(\mathsf{Gen}, \mathsf{Com}, \mathsf{Open})$ *for vectors over* $\mathbb{F}$ *of lengths* $(m, m-\ell-1)$*, where* $\mathsf{x} \in \mathbb{F}^\ell$ *is the public input and output of the instance,* $\overline{E} \in \mathsf{Com}(pp_E, \cdot)$*,* $u \in \mathbb{F}$*,* $\overline{W} \in \mathsf{Com}(pp_W, \cdot)$*,* $A, B, C \in \mathbb{F}^{m \times m}$*,* $m \geq |\mathsf{x}| + 1$*, and there are at most* $n = \Omega(m)$ *non-zero entries in each matrix.*

A tuple (E, r_E, W, r_W)*, where* $E \in \mathbb{F}^m$*,* $W \in \mathbb{F}^{m-\ell-1}$*, and* $r_E, r_W \in \mathbb{F}$ *satisfies a committed relaxed R1CS instance* $((\mathbb{F}, A, B, C, m, n, \ell), (\overline{E}, u, \overline{W}, \mathsf{x}))$ *if:*

- $(A \cdot Z) \circ (B \cdot Z) = u \cdot (C \cdot Z) + E$*, where* $Z = (W, \mathsf{x}, 1)$*; and*
- $\overline{E} = \mathsf{Com}(\mathsf{pp}_E, E, r_E) \wedge \overline{W} = \mathsf{Com}(\mathsf{pp}_W, W, r_W)$

We refer to the first part of the instance (i.e. $(\mathbb{F}, A, B, C, m, n, \ell)$*) as the structure, and the second part of the instance (i.e.* $(\overline{E}, u, \overline{W}, \mathsf{x})$*)) as the partial instance.*

In our IVC construction (Section 6), it is sometimes useful to fold an accepting committed relaxed R1CS instance into another "trivial" instance with the same structure that is also accepting. We prove below that for any structure, it is possible to efficiently compute a satisfying partial instance-witness pair.

Lemma 2 (Trivial Committed Relaxed R1CS Instance). *Given committed relaxed R1CS structure*

$$S = (\mathbb{F}, A, B, C, m, n, \ell)$$

there exists a trivial, and efficiently computable, partial instance-witness pair, $\perp_\mathrm{I}, \perp_\mathrm{W}$*, such that* $\perp_\mathrm{W}$ *satisfies* $(S, \perp_\mathrm{I})$*.*

Proof. Arbitrarily fix W and x, and let $u = 1$. Then compute $E = AZ \circ BZ - u \cdot CZ$ where $Z = (W, \mathsf{x}, 1)$. Produce commitments $\overline{E}$ and $\overline{W}$ accordingly with randomness $r_E, r_W \leftarrow_R \mathbb{F}$. Let

$$\perp_\mathrm{I} = (\overline{E}, u, \overline{W}, \mathsf{x})$$
$$\perp_\mathrm{W} = (E, r_E, W, r_W).$$

By observation, $\perp_\mathrm{W}$ satisfies $(S, \perp_\mathrm{I})$. □

5 A Folding Scheme for Committed Relaxed R1CS

This section describes a folding scheme for committed relaxed R1CS. The protocol is public-coin, so it can be made non-interactive in the random oracle model, and instantiated in the plain model using a concrete hash function.

Construction 2 (A Folding Scheme for Committed Relaxed R1CS). Assume that committed relaxed R1CS instances are defined with respect to a hiding, binding, and additively homomorphic commitment scheme. The verifier $\mathcal{V}$ holds two committed relaxed R1CS instances with the same structure, and defined with respect to the same public parameters $(\mathsf{pp}_E, \mathsf{pp}_W)$:

$$\phi_1 = ((\mathbb{F}, A, B, C, m, n, \ell), (\overline{E}_1, u_1, \overline{W}_1, \mathsf{x}_1))$$

$$\phi_2 = ((\mathbb{F}, A, B, C, m, n, \ell), (\overline{E}_2, u_2, \overline{W}_2, \mathsf{x}_2))$$

The prover $\mathcal{P}$, in addition to the two instances, holds witnesses to both instances: $(E_1, r_{E_1}, W_1, r_{W_1})$, $(E_2, r_{E_2}, W_2, r_{W_2}) \in (\mathbb{F}^m, \mathbb{F}, \mathbb{F}^{m-\ell-1}, \mathbb{F})$. Let $Z_1 = (W_1, \mathsf{x}_1, 1)$ and $Z_2 = (W_2, \mathsf{x}_2, 1)$. The prover and the verifier proceed as follows.

1. $\mathcal{P}$: $\overline{T} \leftarrow \mathsf{Com}(\mathsf{pp}_E, T, r_T)$, where $r_T \leftarrow_R \mathbb{F}$ and T is the cross-term:

$$T = AZ_1 \circ BZ_2 + AZ_2 \circ BZ_1 - u_1 \cdot CZ_2 - u_2 \cdot CZ_1$$

2. $\mathcal{P} \rightarrow \mathcal{V}$: $\overline{T}$.
3. $\mathcal{V} \rightarrow \mathcal{P}$: $r \leftarrow_R \mathbb{F}$.
4. $\mathcal{V}, \mathcal{P}$: Output a new instance with the same structure $\phi = ((\mathbb{F}, A, B, C, m, n, \ell), (\overline{E}, u, \overline{W}, \mathsf{x}))$, where:

$$\begin{aligned} \mathsf{x} &\leftarrow \mathsf{x}_1 + r \cdot \mathsf{x}_2 \\ \overline{E} &\leftarrow \overline{E}_1 + r \cdot \overline{T} + r^2 \cdot \overline{E}_2 \\ u &\leftarrow u_1 + r \cdot u_2 \\ \overline{W} &\leftarrow \overline{W}_1 + r \cdot \overline{W}_2. \end{aligned}$$

5. $\mathcal{P}$: Outputs a new witness (E, r_E, W, r_W), where

$$\begin{aligned} E &\leftarrow E_1 + r \cdot T + r^2 \cdot E_2 \\ r_E &\leftarrow r_{E_1} + r \cdot r_T + r^2 \cdot r_{E_2} \\ W &\leftarrow W_1 + r \cdot W_2 \\ r_W &\leftarrow r_{W_1} + r \cdot r_{W_2} \end{aligned}$$

Lemma 3 (Completeness). *Construction 2 is a folding scheme for committed relaxed R1CS with perfect completeness.*

Proof. Consider two committed relaxed R1CS instances with the same structure and defined with respect to the same public parameters $(\mathsf{pp}_E, \mathsf{pp}_W)$:

$$\phi_1 = ((\mathbb{F}, A, B, C, m, n, \ell), (\overline{E}_1, u_1, \overline{W}_1, \mathsf{x}_1))$$

$$\phi_2 = ((\mathbb{F}, A, B, C, m, n, \ell), (\overline{E}_2, u_2, \overline{W}_2, \mathsf{x}_2))$$

Suppose that the prover $\mathcal{P}$, in addition to the two instances, holds *satisfying* witnesses to both instances: (E_1, W_1), $(E_2, W_2) \in (\mathbb{F}^m, \mathbb{F}^{m-\ell-1})$. Let $Z_1 = (W_1, \mathsf{x}_1, 1)$ and $Z_2 = (W_2, \mathsf{x}_2, 1)$.

Now suppose that the prover and verifier compute a folded instance $\phi = ((\mathbb{F}, A, B, C, m, n, \ell), (\overline{E}, u, \overline{W}, \mathsf{x})$, and suppose that the prover computes a witness (E, W), using the procedure described in Construction 2. To prove completeness, we must show that (E, W) is a satisfying witness for instance ϕ. Let $Z = (W, x, 1)$.

For (E, W) to be a satisfying witness, we must have

$$AZ \circ BZ = u \cdot CZ + E \tag{1}$$

and

$$\overline{E} = \mathsf{Com}(\mathsf{pp}_E, E, r_E) \tag{2}$$
$$\overline{W} = \mathsf{Com}(\mathsf{pp}_W, W, r_W) \tag{3}$$

It is easy to see that Equations (2) and (3) hold from the additive homomorphism of the commitment scheme.

Thus, we focus on proving that Equation (1) also holds. By construction, for Equation (1) to hold, we must have for $r \in_R \mathbb{F}$

$$A(Z_1 + r \cdot Z_2) \circ B(Z_1 + r \cdot Z_2) = (u_1 + r \cdot u_2) \cdot C(Z_1 + r \cdot Z_2) + E.$$

Distributing, we must have

$$\begin{aligned}&AZ_1 \circ BZ_1 + r(AZ_1 \circ BZ_2 + AZ_2 \circ BZ_1) + r^2(AZ_2 \circ AZ_2) = \\ &u_1 \cdot CZ_1 + r(u_1 \cdot CZ_2 + u_2 CZ_1) + r^2 \cdot u_2 \cdot CZ_2 + E.\end{aligned}$$

Aggregating by powers of r, we must have

$$\begin{aligned}&(AZ_1 \circ BZ_1 - u_1 \cdot CZ_1) + \\ &r(AZ_1 \circ BZ_2 + AZ_2 \circ BZ_1 - u_1 \cdot CZ_2 - u_2 CZ_1) + \\ &r^2(AZ_2 \circ AZ_2 - u_2 \cdot CZ_2) = \\ &E.\end{aligned} \tag{4}$$

However, because W_1 and W_2 are satisfying witnesses, we have

$$\begin{aligned}AZ_1 \circ BZ_1 - u_1 \cdot CZ_1 &= E_1 \\ AZ_2 \circ AZ_2 - u_2 \cdot CZ_2 &= E_2.\end{aligned}$$

Additionally, by construction we have

$$AZ_1 \circ BZ_2 + AZ_2 \circ BZ_1 - u_1 \cdot CZ_2 - u_2 \cdot CZ_1 = T.$$

Thus, by substitution, for Equation (4) to hold we must have

$$E_1 + r \cdot T + r^2 \cdot E_2 = E,$$

which holds by construction.

□

Lemma 4 (Knowledge Soundness). *Construction 2 is a knowledge sound folding scheme for committed relaxed R1CS.*

Proof. Consider two committed relaxed R1CS instances with the same structure and defined with respect to the same public parameters $(\mathsf{pp}_E, \mathsf{pp}_W)$:

$$\phi_1 = ((\mathbb{F}, A, B, C, m, n, \ell), (\overline{E}_1, u_1, \overline{W}_1, \mathsf{x}_1))$$

$$\phi_2 = ((\mathbb{F}, A, B, C, m, n, \ell), (\overline{E}_2, u_2, \overline{W}_2, \mathsf{x}_2))$$

We prove knowledge soundness via the forking lemma (Theorem 1). That is we prove that there exists PPT $\mathcal{X}$ such that when given a tree of accepting transcripts and corresponding folded instance-witness pairs outputs a satisfying witness with probability $1 - \mathsf{negl}(\lambda)$.

Indeed, suppose $\mathcal{X}$ is provided three transcripts (τ_1, τ_2, τ_3), each of which comes attached with some accepting witness (W, E, r_W, r_E). Let (r_1, r_2, r_3) denote the verifier's random challenges in each of these transcripts. Interpolating points $(r_1, \tau_1.W)$ and $(r_2, \tau_2.W)$, $\mathcal{X}$ retrieves W_1, W_2 such that

$$W_1 + r_i \cdot W_2 = \tau_i.W \tag{5}$$

for $i \in \{1, 2\}$. Similarly interpolating points $(r_1, \tau_1.E)$, $(r_2, \tau_2.E)$, $(r_3, \tau_3.E)$, $\mathcal{X}$ retrieves E_1, E_2 and a cross-term T such that

$$E_1 + r_i \cdot T + r_i^2 \cdot E_2 = \tau_i.E \tag{6}$$

for $i \in \{1, 2, 3\}$. Using the same approach $\mathcal{X}$ can interpolate for r_{W_1}, r_{W_2} and r_{E_1}, r_T, r_{E_2}. We must argue that $(W_1, E_1, r_{W_1}, r_{E_1})$ and $(W_2, E_2, r_{W_2}, r_{E_2})$ are indeed satisfying witnesses for ϕ_1 and ϕ_2 respectively.

We pick W, E, r_W, r_E, and r, from the first accepting transcript. We first show that the retrieved witness elements are valid openings to the corresponding commitments in the instance. Because W, r_W is part of a satisfying witness, by construction, we must have

$$\begin{aligned}
&\mathsf{Com}(\mathsf{pp}_W, W_1, r_{W_1}) + r \cdot \mathsf{Com}(\mathsf{pp}_W, W_2, r_{W_2}) \\
&= \mathsf{Com}(\mathsf{pp}_W, W_1 + r \cdot W_2, r_{W_1} + r \cdot r_{W_2}) \\
&= \mathsf{Com}(\mathsf{pp}_W, W, r_W) \\
&= \overline{W}_1 + r \cdot \overline{W}_2.
\end{aligned}$$

Because r was chosen independently after $\overline{W}_1$ and $\overline{W}_2$ were committed, we must have that

$$\begin{aligned}
\mathsf{Com}(\mathsf{pp}_W, W_1, r_{W_1}) &= \overline{W}_1 \\
\mathsf{Com}(\mathsf{pp}_W, W_2, r_{W_2}) &= \overline{W}_2
\end{aligned}$$

with probability $1 - \mathsf{negl}(\lambda)$.

Similarly, because E, r_E is part of a satisfying witness, by construction, we must have

$$\begin{aligned}
&\mathsf{Com}(\mathsf{pp}_E, E_1, r_{E_1}) + r \cdot \mathsf{Com}(\mathsf{pp}_E, T, r_T) + r^2 \cdot \mathsf{Com}(\mathsf{pp}_E, E_2, r_{E_2})\\
&= \mathsf{Com}(\mathsf{pp}_E, E_1 + r \cdot T + r^2 \cdot E_2, r_{E_1} + r \cdot r_T + r^2 \cdot r_{E_2})\\
&= \mathsf{Com}(\mathsf{pp}_E, E, r_E)\\
&= \overline{E}_1 + r \cdot \overline{E}_2.
\end{aligned}$$

Because r was chosen independently after $\overline{E}_1$, $\overline{T}$, and $\overline{E}_2$ were committed, we have that

$$\begin{aligned}
\mathsf{Com}(\mathsf{pp}_E, E_1, r_{E_1}) &= \overline{E}_1\\
\mathsf{Com}(\mathsf{pp}_E, E_2, r_{E_1}) &= \overline{E}_2
\end{aligned}$$

with probability $1 - \mathsf{negl}(\lambda)$.

Next, we must show that (W_1, E_1) and (W_2, E_2) satisfy the relaxed R1CS relation. Because (W, E) is part of a satisfying witness, we have

$$AZ \circ BZ = u \cdot CZ + E$$

where $Z = (W, x, 1)$. However, by Equations (5) and (6) this implies that

$$\begin{aligned}
&A(Z_1 + r \cdot Z_2) \circ B(Z_1 + r \cdot Z_2) =\\
&(u_1 + r \cdot u_2) \cdot C(Z_1 + r \cdot Z_2) + (E_1 + r \cdot T + r^2 \cdot E_2)
\end{aligned}$$

where $Z_i = (W_i, x_i, 1)$. We observe that r was chosen independently and randomly after commitments to W_1, W_2, E_1, E_2, and T were provided. Thus, by expanding and applying the Schwartz-Zippel lemma, we must have with probability $1 - \mathsf{negl}(\lambda)$

$$\begin{aligned}
AZ_1 \circ BZ_1 &= u_1 \cdot CZ_1 + E_1\\
AZ_2 \circ BZ_2 &= u_2 \cdot CZ_2 + E_2.
\end{aligned}$$

Thus $(W_1, E_1, r_{W_1}, r_{E_1})$ and $(W_2, E_2, r_{W_2}, r_{E_2})$ meet all the requirements to be satisfying witnesses for ϕ_1 and ϕ_2 respectively. □

Lemma 5 (Honest-Verifier Zero-Knowledge). *Construction 2 is an honest-verifier zero-knowledge folding scheme for committed relaxed R1CS.*

Proof. Intuitively, zero-knowledge holds because the prover only sends a single hiding commitment. More formally, we construct a simulator as follows: Simulator $\mathcal{S}$ samples random $t \in \mathbb{F}^m$ and $r \in \mathbb{F}$ and computes

$$T = \mathsf{Com}(\mathsf{pp}_E, t, r). \qquad (7)$$

Next, $\mathcal{S}$ derives the verifier's challenge, r, using ρ and outputs $\mathsf{tr} = (T, r)$. (Perfect) zero-knowledge holds from the (perfect) hiding property of the underlying commitment scheme. □

As part of our construction of IVC from folding schemes, we require our folding scheme for committed relaxed R1CS to be non-interactive.

Construction 3 (Non-Interactive Folding Scheme). Because the verifier in our folding scheme for committed relaxed R1CS is public-coin, we can apply the Fiat-Shamir transform [FS86]: The non-interactive prover only sends a commitment to the cross term $\overline{T}$ to the verifier. Next, both the non-interactive prover and the verifier hash their public inputs and $\overline{T}$ to simulate the verifier's challenge and complete the folding procedure.

6 Nova: IVC from Folding Schemes

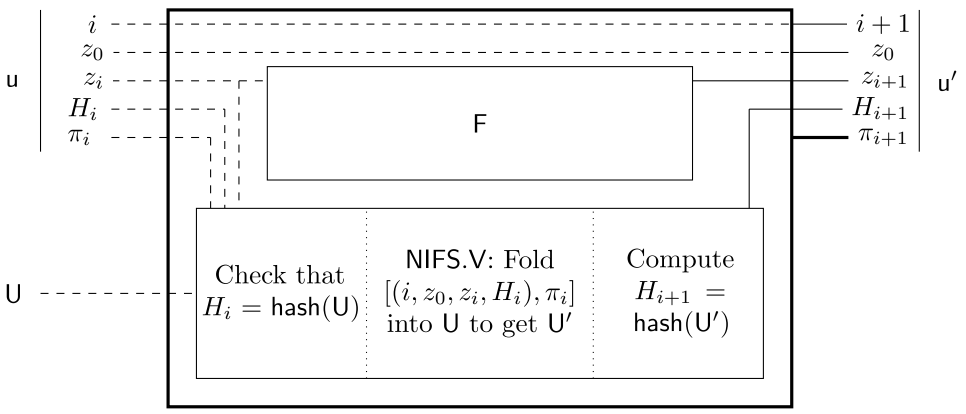
In this section we present Nova, a zero-knowledge proof system for incremental computations, where each incremental step is expressed with C-sized R1CS.

We first show how to achieve IVC from folding schemes for committed relaxed R1CS (Construction 4), which when instantiated with the any additively-homomorphic commitment scheme with succinct commitments and a circuit-friendly hash function, achieves the claimed efficiency results (Lemma 8). At each incremental step, the prover folds into a running C-sized committed relaxed R1CS instance. At the end of the incremental computation, the "IVC proof" is an NP witness to the running committed relaxed R1CS instance produced at the final step of the incremental computation.

We then show that, at any incremental step or after the final step, Nova's prover can prove in zero-knowledge and with a succinct proof—using a variant of an existing zkSNARK [Set20] adapted to handle committed relaxed R1CS instances (Section 7)—that it knows a valid NP witness to the running committed relaxed R1CS instance (Construction 5). We note that Nova is *not* a zero-knowledge IVC scheme (Definition 6), as that would additionally require that the IVC proof itself be zero-knowledge. This difference is immaterial in the context of a single prover, and we leave it to future work to achieve zero-knowledge IVC.

Constructing IVC from Folding. Recall that an incrementally verifiable computation scheme allows a prover to show that $z_n = \mathsf{F}^{(n)}(z_0)$ for some application count n, initial input z_0, and final output z_n. We show how to construct IVC for non-deterministic, polynomial-time computable function F using our non-interactive folding scheme for committed relaxed R1CS (Construction 3). However, our folding scheme achieves non-interactivity in the random oracle model using the Fiat-Shamir transform [FS86]. We must instantiate this (heuristically) in the plain model using a hash function *before* we can construct an IVC scheme. Similar transformations must be made in existing IVC/PCD schemes [Val08, BGH19, COS20, BCL^{+}20].

We first define polynomial-time function F' (Figure 2) with private inputs and public outputs, which, in addition to invoking F, performs additional bookkeeping to preserve invariants required by IVC. Because F' can be computed in polynomial-time, it can be represented as a Relaxed R1CS instance. Under this observation,



u
i
z_0
z_i
H_i
π_i
F
$i+1$
z_0
z_{i+1}
H_{i+1}
π_{i+1}
u′
U
Check that $H_i = \mathsf{hash}(\mathsf{U})$
NIFS.V: Fold $[(i, z_0, z_i, H_i), \pi_i]$ into U to get U′
Compute $H_{i+1} = \mathsf{hash}(\mathsf{U}')$

Fig. 2. Overview of our IVC recursion function F'. Dotted line indicates private inputs. Solid line indicates public outputs. Bold line indicates proof of correct execution. Minor checks and auxiliary inputs ω_i and $\overline{T}$ omitted. π_i represents the non-io portion of instance u (i.e. $(\overline{E}, u, \overline{W})$).

we achieve IVC by incrementally folding satisfying committed relaxed R1CS partial instance-witness pairs for invocations of F'.

Construction 4 (IVC for Committed Relaxed R1CS). Let $\mathsf{NIFS} = (\mathsf{P}, \mathsf{V})$ be the non-interactive folding scheme for committed relaxed R1CS (Construction 3). Consider polynomial-time function F that takes non-deterministic input.

At a high level, we construct polynomial-time recursion function F', represented as a committed relaxed R1CS instance with structure $S_{\mathsf{F}'}$. On invocation i, F' takes as input two partial instances u and U. Instance u represents the correct execution of the last invocation of F' with respect to $S_{\mathsf{F}'}$ (i.e. invocation $i-1$). Instance U is a folded instance representing the correct execution of all prior invocations of F' (i.e. invocations $0, \ldots, i-2$). The first instance contains z_i which F' uses to output $z_{i+1} = \mathsf{F}(z_i)$. F' additionally runs $\mathsf{NIFS.V}$ to fold its input instances and output a new folded instance U'. The IVC prover then generates a new instance u' which represents the correct execution of invocation i of F'. Together $(\mathsf{u}', \mathsf{U}')$ represent the input to the next invocation of F'.

To prevent progressively increasing public io sizes, we modify F' such that it only outputs the hash of the new folded instance, which the next invocation of F' checks. F' additionally keeps track of the initial input z_0 and invocation count i. For the base case, we define $(\bot_{\mathrm{I}}, \bot_{\mathrm{W}})$ to be the trivially satisfying partial instance-witness pair for $S_{\mathsf{F}'}$ (which can be computed via Lemma 2).

$\underline{\mathsf{F}'(\mathsf{u}, \mathsf{U}, \overline{T}, \omega_i) \to \mathsf{x}}$:

1. $(i, z_0, z_i, H_i) \leftarrow \mathsf{u.x}$, where $\mathsf{u.x}$ is the public io of u

2.
$$\mathsf{U}' \leftarrow \begin{cases} \bot_{\mathrm{I}} & \text{if } i = 0 \\ \mathsf{NIFS.V}(\mathsf{u}, \mathsf{U}, \overline{T}) & \text{otherwise} \end{cases}$$

3. If $i \neq 0$, check that $H_i = \mathsf{hash}(\mathsf{U})$ and $(\mathsf{u}.\overline{E}, \mathsf{u}.u) = (0, 1)$
4. Output $\mathsf{x} \leftarrow (i+1, z_0, \mathsf{F}(z_i, \omega_i), \mathsf{hash}(\mathsf{U}'))$

Given function F', we next describe a single incremental step for the IVC prover. At a high level, in each step, the prover takes as input previous (satisfying) instance for F' and a folded instance along with the corresponding witnesses. The prover then folds its inputs into a new folded instance-witness pair and additionally outputs a new (satisfying) instance to be folded, which represents a fresh invocation of F'. Assume, for notational simplicity, that the prover and verifier globally have access to function F. We define the IVC prover $\mathcal{P}$ and corresponding verifier $\mathcal{V}$:

$\underline{\mathcal{P}(i, z_0, \Pi_i, \omega_i) \rightarrow \Pi_{i+1}}$:

1. $((\mathsf{u}, \mathsf{w}), (\mathsf{U}, \mathsf{W})) \leftarrow \Pi_i$
2.
$$(\mathsf{U}', \mathsf{W}', \overline{T}) \leftarrow \begin{cases} \bot_{\mathrm{I}}, \bot_{\mathrm{W}}, 0 & \text{if } i = 0 \\ \mathsf{NIFS.P}((\mathsf{u}, \mathsf{w}), (\mathsf{U}, \mathsf{W})) & \text{otherwise} \end{cases}$$

3. Let $(\mathsf{u}', \mathsf{w}')$ be a satisfying committed relaxed R1CS instance-witness pair for the computation F' on input $(\mathsf{u}, \mathsf{U}, \overline{T}, \omega_i)$ with $(E, u) = (\mathbf{0}, 1)$
4. Output $\Pi_{i+1} \leftarrow (\mathsf{u}', \mathsf{w}'), (\mathsf{U}', \mathsf{W}')$

$\underline{\mathcal{V}(n, z_0, z_n, \Pi_n) \rightarrow \{0, 1\}}$:

1. $((\mathsf{u}, \mathsf{w}), (\mathsf{U}, \mathsf{W})) \leftarrow \Pi_n$.
2. If $n = 0$ and $(\mathsf{u}.i, \mathsf{u}.z_0) = (0, z_0)$ return 1
3. Check that $\mathsf{u.x} = (n, z_0, z_n, \mathsf{hash}(\mathsf{U}))$ and $(\mathsf{u}.\overline{E}, \mathsf{u}.u) = (0, 1)$
4. Check (u, w) and (U, W) are satisfying committed R1CS instance-witness pairs associated with function F'

Lemma 6 (Completeness). *Construction 4 is an IVC scheme that satisfies completeness.*

Proof. Consider *valid* $(\mathsf{F}, i+1, z_0, z_{i+1})$ and corresponding intermediate inputs (z_i, ω_i) such that

$$z_{i+1} = \mathsf{F}(z_i, \omega_i)$$

Now consider valid proof Π_i such that

$$\mathcal{V}(i, z_0, z_i, \Pi_i) = 1$$

We must show that given

$$\mathit{\Pi}_{i+1} \leftarrow \mathcal{P}(i, z_0, \mathit{\Pi}_i, \omega_i)$$

that

$$\mathcal{V}(i+1, z_0, z_n, \mathit{\Pi}_{i+1}) = 1$$

with probability 1. We show this by induction on i.

<u>Base Case ($i = 0$)</u>: Suppose the prover is provided $\mathit{\Pi}_0$ such that

$$\mathcal{V}(0, z_0, z_0, \mathit{\Pi}_0) = 1.$$

By the verifier's base case check, $\mathit{\Pi}_0$ can be arbitrary with the exception that $(\mathit{\Pi}_0.\mathsf{u}.i, \mathit{\Pi}_0.\mathsf{u}.z_0) = (0, z_0)$.

Given that $\mathit{\Pi}_0$ meets these conditions, by the base case of $\mathcal{P}$ and F' we have

$$\mathit{\Pi}_1 = ((\mathsf{u}, \mathsf{w}), (\bot_\mathrm{I}, \bot_\mathrm{W}))$$

for some (u, w). By definition, the instance-witness pair $(\bot_\mathrm{I}, \bot_\mathrm{W})$ satisfies F'. Moreover, in the base case when $i = 0$, F' performs no checks on its auxiliary inputs, and thus (u, w) must also be satisfying. Additionally, in the base case we have

$$\mathsf{u}.\mathsf{x} = (1, z_0, \mathsf{F}(z_0, w_0), \mathsf{hash}(\mathbf{0})).$$

Therefore we have

$$\mathcal{V}(1, z_0, z_1, \mathit{\Pi}_1) = 1.$$

<u>Inductive Step ($i > 0$)</u>: Assume that for

$$\mathit{\Pi}_i = ((\mathsf{u}, \mathsf{w}), (\mathsf{U}, \mathsf{W}))$$

we have that

$$\mathcal{V}(i, z_0, z_i, \mathit{\Pi}_i) = 1$$

and suppose that

$$\mathcal{P}(i, z_0, \mathit{\Pi}_i, \omega_i) = \mathit{\Pi}_{i+1} = ((\mathsf{u}', \mathsf{w}'), (\mathsf{U}', \mathsf{W}')).$$

By construction of $\mathcal{P}$, we have that

$$\mathsf{U}', \mathsf{W}', \overline{T} = \mathsf{NIFS.P}((\mathsf{u}, \mathsf{w}), (\mathsf{U}, \mathsf{W})).$$

Thus, by the correctness of the underlying folding scheme, and the premise that $(\mathsf{u}, \mathsf{w}), (\mathsf{U}, \mathsf{W})$ satisfy F', we have that $(\mathsf{U}', \mathsf{W}')$ satisfies F'. Additionally, by construction of $\mathcal{P}$ and the premise that

$$\mathsf{u}.H_i = \mathsf{hash}(\mathsf{U})$$

we have that $(\mathsf{u}', \mathsf{w}')$ satisfies F' on input $(\mathsf{u}, \mathsf{U}, \overline{T}, \omega_i)$. By construction, this particular input implies that

$$\begin{aligned} \mathsf{u}'.\mathsf{x} &= (\mathsf{u}.i + 1, \mathsf{u}.z_0, \mathsf{F}(z_i, w_i), \mathsf{hash}(\mathsf{NIFS.V}(\mathsf{u}, \mathsf{U}, \overline{T}))) \\ &= (\mathsf{u}.i + 1, \mathsf{u}.z_0, z_{i+1}, \mathsf{hash}(\mathsf{U}')) \end{aligned} \tag{8}$$

by the correctness of the underlying folding scheme. Thus by Equation (8) we have

$$\mathcal{V}(i + 1, z_0, z_{i+1}, \Pi_{i+1}) = 1.$$

□

Lemma 7 (Knowledge Soundness). *Construction 4 is an IVC scheme that satisfies knowledge soundness.*

Proof. Consider arbitrary PPT adversary $\mathcal{P}^*$. We must construct an extractor $\mathcal{E}$ such that for all $(\mathsf{F}, n, z_0, z_n)$ and input randomness ρ, can extract valid $(z_1, \ldots, z_{n-1})$, $(\omega_0, \ldots, \omega_{n-1})$ nearly as successfully as $\mathcal{P}^*$ can produce an accepting proof. Indeed, suppose $\mathcal{P}^*$ succeeds with probability ϵ in producing Π_n such that

$$\mathcal{V}(n, z_0, z_n, \Pi_n) = 1.$$

Initially, $\mathcal{E}$ runs $\mathcal{P}^*$ on input $(\mathsf{F}, n, z_0, z_n)$ to retrieve an accepting proof Π_n with probability $\epsilon - \mathsf{negl}(\lambda)$. With Π_n on hand, $\mathcal{E}$ inductively proceeds as follows:

Inductive Step ($i > 1$): Suppose that $\mathcal{E}$ has thus far extracted

$$\Pi_i = ((\mathsf{u}', \mathsf{w}'), (\mathsf{U}', \mathsf{W}'))$$

such that

$$\mathcal{V}(i, z_0, z_i, \Pi_i) = 1. \tag{9}$$

We will show that $\mathcal{E}$ can extract valid z_{i-1}, ω_{i-1}, and Π_{i-1}. Indeed, by the inductive hypothesis (9), we have that $(\mathsf{u}', \mathsf{w}')$ satisfies F'. Thus, using w', $\mathcal{E}$ can retrieve u and ω_{i-1} such that

$$\mathsf{u}.z_0 = \mathsf{u}'.z_0 \tag{10}$$

$$\mathsf{u}.i = \mathsf{u}'.i - 1 \tag{11}$$

$$\mathsf{F}(\mathsf{u}.z_{i-1}, \omega_{i-1}) = \mathsf{u}'.z_i \tag{12}$$

by the construction of F'. Thus, by Equation (12), we have that $\mathcal{E}$ has extracted valid z_{i-1} and ω_{i-1}. Next, by the inductive hypothesis (9), we have

$$\mathsf{u}'.H_i = \mathsf{hash}(\mathsf{U}').$$

Thus, by the binding property of hash, and by the construction of F', $\mathcal{E}$ can use w_i to retrieve U, $\overline{T}$ such that

$$\mathsf{U}' = \mathsf{NIFS.V}(\mathsf{U}, \mathsf{u}, \overline{T}) \tag{13}$$
$$\mathsf{u}.H_{i-1} = \mathsf{hash}(\mathsf{U}). \tag{14}$$

Once again, by the inductive hypothesis (9), we have that $(\mathsf{U}', \mathsf{W}')$ satisfies F'. Therefore, by Equation (13), and the knowledge-soundness of the underlying folding scheme, $\mathcal{E}$ can extract w, W such that (u, w), and (U, W) satisfy F' with probability $1 - \mathsf{negl}(\lambda)$. Additionally, by the inductive hypothesis, we have that $\mathsf{u}.\overline{E} = 0$ which implies that $\mathsf{w}.E = \mathbf{0}$ with overwhelming probability. Thus, by Equations (10), (11), (12), (14), the extractor has extracted

$$\Pi_{i-1} = ((\mathsf{u}, \mathsf{w}), (\mathsf{U}, \mathsf{W}))$$

such that

$$\mathcal{V}(i-1, z_0, z_{i-1}, \Pi_{i-1}) = 1$$

Base Case ($i = 1$): Suppose that $\mathcal{E}$ has extracted

$$\Pi_1 = ((\mathsf{u}', \mathsf{w}'), (\mathsf{U}', \mathsf{W}'))$$

such that

$$\mathcal{V}(1, z_0, z_1, \Pi_1) = 1. \tag{15}$$

$\mathcal{E}$ can retrieve some ω_0 using w'. We must show that this ω_0 is valid. Indeed, by Equation (15) we have that $(\mathsf{u}', \mathsf{w}')$ satisfies F'. This implies that

$$\mathsf{u}'.i = 1$$

and thus implies that $\mathsf{u}'.i - 1 = 0$ was provided as input to F'. Therefore, by the base case of F', we have

$$\mathsf{F}(z_0, \omega_0) = \mathsf{u}'.z_1$$

with probability $1 - \mathsf{negl}(\lambda)$. Thus, the retrieved ω_0 is valid.

Therefore, $\mathcal{E}$ can extracted valid $(z_1, \ldots, z_{n-1})$, $(\omega_0, \ldots, \omega_{n-1})$ with probability $\epsilon - \mathsf{negl}(\lambda)$. □

Lemma 8 (Efficiency). *Instantiated with a commitment scheme for vectors over $\mathbb{F}$ with succinct commitments, $O(|\mathsf{F}'|) = O(|\mathsf{F}|)$. Assuming $|\mathsf{hash}| \ll \mathbb{G}$, we have that*

$$|\mathsf{F}'| \approx |\mathsf{F}| + 2 \cdot \mathbb{G}$$

where $\mathbb{G}$ is the number of relaxed rank-1 constraints required to encode a group scalar multiplication.

Proof. The number of constraints needed to encode the additional work performed by F' is dominated by the scalar multiplications performed by $\mathsf{NIFS.V}$. On input instances u and U, $\mathsf{NIFS.V}$ must compute

$$\begin{aligned}\overline{E} &\leftarrow \mathsf{U}.\overline{E} + r \cdot \overline{T} + r^2 \cdot \mathsf{u}.\overline{E}\\ \overline{W} &\leftarrow \mathsf{U}.\overline{W} + r \cdot \mathsf{u}.\overline{W}\end{aligned}$$

However, we observe that, by construction, $\mathsf{u}.\overline{E} = 0$. Thus the verifier's work is dominated by 2 scalar multiplications, as it does not need to compute $\mathsf{u}.\overline{E}$. □

In practice, the prover may not want to send the final IVC proof in plain. We show how the prover can use a zero-knowledge argument for committed relaxed R1CS (Section 7) to prove that it possesses a valid IVC proof without revealing the associated witness.

Construction 5 (Zero-Knowledge Argument of Valid IVC Proof). Assume a non-interactive zero-knowledge argument of knowledge (NIZK) for committed relaxed R1CS, $\mathsf{NIZK} = (\mathsf{P}, \mathsf{V})$. That is, given a committed relaxed R1CS instance of size N, $\mathsf{NIZK.P}$ can convince $\mathsf{NIZK.V}$ in zero-knowledge and with a succinct proof (e.g., $O_\lambda(\log N)$-sized proof) that it knows a satisfying witness.

Now suppose at the end of n iterations, the IVC prover outputs proof $\Pi = ((\mathsf{u}, \mathsf{w}), (\mathsf{U}, \mathsf{W}))$. We construct prover $\mathcal{P}$, that can convince corresponding verifier $\mathcal{V}$ that it possesses valid witnesses for instances u and U. At a high level, we take advantage of the fact that Π is simply two committed relaxed R1CS instance-witness pairs: $\mathcal{P}$ first folds instance-witness pairs (u, w) and (U, W) to produce folded instance-witness pair $(\mathsf{U}', \mathsf{W}')$, using $\mathsf{NIFS.P}$. Next, $\mathcal{P}$ runs $\mathsf{NIZK.P}$ to prove that it knows a valid witness for U'. In more detail:

$\underline{\mathcal{P}(\Pi) \rightarrow \pi}$:

```
1. $((\mathsf{u}, \mathsf{w}), (\mathsf{U}, \mathsf{W})) \leftarrow \Pi$
2. $((\mathsf{U}', \mathsf{W}'), \overline{T}) \leftarrow \mathsf{NIFS.P}((\mathsf{u}, \mathsf{w}), (\mathsf{U}, \mathsf{W}))$ // fold the two instances in $\Pi$
3. $\pi_{\mathsf{U}'} \leftarrow \mathsf{NIZK.P}(\mathsf{U}', \mathsf{W}')$ // run NIZK.P to produce a succinct ZK proof
4. Output $(\mathsf{u}, \mathsf{U}, \overline{T}, \pi_{\mathsf{U}'})$
```

$\underline{\mathcal{V}(n, z_0, z_n, \pi) \rightarrow \{0, 1\}}$:

```
1. $(\mathsf{u}, \mathsf{U}, \overline{T}, \pi_{\mathsf{U}'}) \leftarrow \pi$
```

2. Check that u and U are well-formed:

$$\mathsf{u}.H_i \stackrel{?}{=} \mathsf{hash}(\mathsf{U})$$

$$(\mathsf{u}.i, \mathsf{u}.z_0, \mathsf{u}.z_i) \stackrel{?}{=} (n, z_0, z_n)$$

3. $\mathsf{U}' \leftarrow \mathsf{NIFS.V}(\mathsf{u}, \mathsf{U}, \overline{T})$ // fold u and U
4. $\mathsf{NIZK.V}(\mathsf{U}', \pi_{\mathsf{U}'}) \stackrel{?}{=} 1$ // check the succinct ZK proof with $\mathsf{NIZK.V}$

7 A zkSNARK for Committed Relaxed R1CS

This section presents a zkSNARK for NP relations expressed with committed relaxed R1CS. To achieve this, we adapt Spartan [Set20]. We build on techniques from Spartan [Set20] to avoid FFTs and a trusted setup.

To construct a zkSNARK for committed relaxed R1CS, we first observe that there is a corresponding indexed relation *indexed relaxed R1CS*; we then devise a polynomial IOP for indexed relaxed R1CS; we then then apply existing transformations to transform it into a zkSNARK for committed relaxed R1CS.

7.1 Background

This subsection is adapted from prior work [Set20, Tha20, LSTW21].

Polynomials. We recall a few basic facts about polynomials.

– A polynomial g over $\mathbb{F}$ is an expression consisting of a sum of *monomials* where each monomial is the product of a constant and powers of one or more variables; all arithmetic is performed over $\mathbb{F}$.
– The degree of a monomial is the sum of the exponents of variables in the monomial; the (total) degree of a polynomial g is the maximum degree of any monomial in g. Furthermore, the degree of a polynomial g in a particular variable x_i is the maximum exponent that x_i takes in any g's monomials.
– A *multivariate* polynomial is a polynomial with more than one variable. A multivariate polynomial is called a *multilinear* polynomial if the degree of the polynomial in each variable is at most one.
– A multivariate polynomial g over a finite field $\mathbb{F}$ is called *low-degree* if the degree of g in each variable is bounded above by a constant.

Definition 13. *Suppose $f : \{0,1\}^\ell \to \mathbb{F}$ is a function that maps ℓ-bit strings to an element of $\mathbb{F}$. A* polynomial extension *of f is a low-degree ℓ-variate polynomial $\widetilde{f}(\cdot)$ such that $\widetilde{f}(x) = f(x)$ for all $x \in \{0,1\}^\ell$.*

Definition 14. *A* multilinear extension (MLE) *of a function $f : \{0,1\}^\ell \to \mathbb{F}$ is a low-degree polynomial extension where the extension is a multilinear polynomial.*

It is well-known that every function $f : \{0,1\}^\ell \to \mathbb{F}$ has a unique multilinear extension, and similarly every ℓ-variate multilinear polynomial of $\mathbb{F}$ extends a unique function mapping $\{0,1\}^\ell \to \mathbb{F}$. In the rest of the document, for a function f, we use $\widetilde{f}$ to denote the unique MLE of f.

The sum-check protocol. The sum-check protocol [LFKN90] is an interactive proof system for proving claims of the form: $T = \sum_{x \in \{0,1\}^\ell} G(x)$, where G is ℓ-variate polynomial over $\mathbb{F}$ with the degree in each variable at most μ, and $T \in \mathbb{F}$. In the sum-check protocol, the verifier $\mathcal{V}_{SC}$ interacts with the prover $\mathcal{P}_{SC}$ over a sequence of ℓ rounds. At the end of this interaction, $\mathcal{V}_{SC}$ must evaluate $G(r)$ where $r \in_R \mathbb{F}^\ell$ is chosen by $\mathcal{V}_{SC}$ over the course of the sum-check protocol. Other than evaluating $G(r)$, $\mathcal{V}_{SC}$ performs $O(\ell \cdot \mu)$ field operations. As in prior work, we view the sum-check protocol as a mechanism to transform claims of the form $\sum_{x \in \{0,1\}^m} G(x) \overset{?}{=} T$ to the claim $G(r) \overset{?}{=} e$, where $e \in \mathbb{F}$. This is because in most cases, the verifier uses an auxiliary protocol to verify the latter claim.

7.2 Indexed relation associated with Committed Relaxed R1CS

Recall that a relation $\mathcal{R}$ is a set of tuples $(\mathbb{x}, w)$, where $\mathbb{x}$ is the instance and w is the witness. We consider a generalized notion of relations called *indexed relations*, which are implicit in [Set20] but formalized in [COS20]. An indexed relation $\mathcal{R}$ is a set of triples $(\mathbb{i}, \mathbb{x}, w)$, where $\mathbb{i}$ is the index, $\mathbb{x}$ is the instance, and w is the witness.

For a committed relaxed R1CS instance, $\phi = ((\mathbb{F}, A, B, C, m, n, \ell), (\overline{E}, u, \overline{W}, \mathsf{x}))$, defined with respect to two sets of public parameters $(\mathsf{pp}_E, \mathsf{pp}_W)$ produced from a commitment scheme for vectors over $\mathbb{F}$ of lengths $(m, m - \ell - 1)$, there is a natural indexed relation that we refer to as indexed relaxed R1CS.

Definition 15 (Indexed Relaxed R1CS). *An indexed relaxed R1CS instance is a tuple* $(\mathbb{i}, \mathbb{x})$*, where* $\mathbb{i} = (\mathbb{i}_\mathcal{V}, \mathbb{i}_\mathcal{P})$*,* $\mathbb{i}_\mathcal{V} = (\mathbb{F}, A, B, C, m, n, \ell)$*,* $\mathbb{i}_\mathcal{P} = (E, W)$*,* $\mathsf{x} \in \mathbb{F}^\ell$ *is the public input and output of the instance,* $E \in \mathbb{F}^m$*,* $u \in \mathbb{F}$*,*$W \in \mathbb{F}^{m-\ell-1}$*,* $A, B, C \in \mathbb{F}^{m \times m}$*,* $m \geq |\mathsf{x}| + 1$*, and there are at most* $n = \Omega(m)$ *non-zero entries in each matrix.*

An indexed relaxed R1CS instance $(((\mathbb{F}, A, B, C, m, n, \ell), (E, W)), (u, \mathsf{x}))$ *is satisfiable if:* $(A \cdot Z) \circ (B \cdot Z) = u \cdot (C \cdot Z) + E$*, where* $Z = (W, \mathsf{x}, 1)$*.*

7.3 A polynomial IOP for Indexed Relaxed R1CS

Our exposition below is based on Spartan [Set20] and its recent recapitulation [LSTW21]. The theorem below and its proof is a verbatim adaptation of Spartan's polynomial IOP for indexed R1CS to indexed relaxed R1CS.

Recall that an interactive proof (IP) [GMR85] for a relation $\mathcal{R}$ is an interactive protocol between a prover and a verifier where the prover proves the knowledge of a witness w for an instance $\mathbb{x}$ such that $(\mathbb{x}, w) \in \mathcal{R}$. An interactive oracle proof (IOP) [BCS16, RRR16] generalizes interactive proofs where in each round the prover sends an oracle (e.g., a string) and the verifier may query the prover's oracle. A polynomial IOP [BFS20] is an IOP except for: (1) the prover's message in each round i consists of a message that the verifier reads in entirety, followed optionally by a polynomial g_i over $\mathbb{F}$; and (2) a verifier may request an evaluation of g_i at a point in its domain.

For an indexed relaxed R1CS instance $(\mathbb{i}, \mathbb{x})$, where $\mathbb{i} = ((\mathbb{F}, A, B, C, m, n, \ell), (E, W))$ and $\mathbb{x} = (u, \mathsf{x})$, we interpret the matrices A, B, C as functions with signature $\{0,1\}^{\log m} \times \{0,1\}^{\log m} \to \mathbb{F}$ in a natural manner. In other words, an input in $\{0,1\}^{\log M} \times \{0,1\}^{\log m}$ is interpreted as the binary representation of an index $(i,j) \in [m] \times [m]$, where $[m] \coloneqq \{1, \ldots, m\}$ and the function outputs the (i,j)'th entry of the matrix. Furthermore, let $\widetilde{A}, \widetilde{B}, \widetilde{C}$ denote multilinear extensions of A, B, C interpreted as functions, so they are $2\log m$-variate sparse multilinear polynomials of size n.

Similarly, we interpret E and W as functions with respective signatures: $\{0,1\}^{\log m} \to \mathbb{F}$ and $\{0,1\}^{\log m - 1} \to \mathbb{F}$. Furthermore, let $\widetilde{E}$ and $\widetilde{W}$ denote the multilinear extensions of E and W interpreted as functions, so they are multilinear polynomials in $\log m$ and $\log m - 1$ variables respectively.

Theorem 2. *There exists a polynomial IOP for indexed relaxed R1CS defined over a finite field $\mathbb{F}$, with the following parameters, where m denotes the dimension of the R1CS coefficient matrices, and n denotes the number of non-zero entries in the matrices:*

- *soundness error is $O(\log m)/|\mathbb{F}|$*
- *round complexity is $O(\log m)$;*
- *the verifier has a query access to a $(\log m) - 1$-variate multilinear polynomial $\widetilde{W}$ and a $\log m$-variate multilinear polynomial $\widetilde{E}$; the verifier also has a query access to three $2\log m$-variate multilinear polynomials of size n $\widetilde{A}, \widetilde{B}, \widetilde{C}$;*
- *the verifier issues one evaluation query to polynomials $\widetilde{W}, \widetilde{E}, \widetilde{A}, \widetilde{B}, \widetilde{C}$ and otherwise performs $O(\log m)$ operations over $\mathbb{F}$;*
- *the prover performs $O(n)$ operations over $\mathbb{F}$ to compute its messages in the polynomial IOP and to respond to the verifier's queries to $\widetilde{W}, \widetilde{E}, \widetilde{A}, \widetilde{B}, \widetilde{C}$.*

Proof. Let $s = \log m$. For an indexed relaxed R1CS instance,

$$\phi = (((\mathbb{F}, A, B, C, m, n, \ell), (E, W)), (u, \mathsf{x})),$$

let $Z = (W, 1, \mathsf{x})$. Similar to interpreting matrices as functions, we interpret Z and $(1, \mathsf{x})$ as functions with the following respective signatures: $\{0,1\}^s \to \mathbb{F}$ and $\{0,1\}^{s-1} \to \mathbb{F}$. Observe that the MLE $\widetilde{Z}$ of Z satisfies

$$\widetilde{Z}(X_1, \ldots, X_{\log m}) = (1 - X_1) \cdot \widetilde{W}(X_2, \ldots, X_{\log m}) + X_1 \cdot \widetilde{(1, \mathsf{x})}(X_2, \ldots, X_{\log m}) \tag{16}$$

Similar to [Set20, Theorem 4.1], checking if ϕ is satisfiable is equivalent, except for a soundness error of $\log m/|\mathbb{F}|$ over the choice of $\tau \in \mathbb{F}^s$, to checking if the following identity holds:

$$0 \overset{?}{=} \sum_{x \in \{0,1\}^s} \widetilde{\mathsf{eq}}(\tau, x) \cdot F(x), \tag{17}$$

where

$$F(x) = \left(\sum_{y\in\{0,1\}^s} \widetilde{A}(x,y)\cdot\widetilde{Z}(y)\right)\cdot\left(\sum_{y\in\{0,1\}^s} \widetilde{B}(x,y)\cdot\widetilde{Z}(y)\right) -$$
$$\left(u\cdot\sum_{y\in\{0,1\}^s} \widetilde{C}(x,y)\cdot\widetilde{Z}(y)+\widetilde{E}(x)\right),$$

and $\widetilde{\mathsf{eq}}$ is the multilinear extension of $\mathsf{eq} : \{0,1\}^s \times \{0,1\}^s \to \mathbb{F}$:

$$\mathsf{eq}(x,e) = \begin{cases} 1 & \text{if } x = e \\ 0 & \text{otherwise.} \end{cases}$$

That is, if ϕ is satisfiable, then Equation (17) holds with probability 1 over the choice of τ, and if not, then Equation (17) holds with probability at most $O(\log m/|\mathbb{F}|)$ over the random choice of τ.

To compute the RHS in Equation (17), the prover and the verifier can apply the sum-check protocol to the following polynomial:

$$g(x) \coloneqq \widetilde{\mathsf{eq}}(\tau,x)\cdot F(x)$$

From the verifier's perspective, this reduces the task of computing the right hand side of Equation (17) to the task of evaluating g at a random input $r_x \in \mathbb{F}^s$. Note that the verifier can locally evaluate $\widetilde{\mathsf{eq}}(\tau,r_x)$ in $O(\log m)$ field operations via $\widetilde{\mathsf{eq}}(\tau,r_x) = \prod_{i=1}^{s}(\tau_i r_{x,i} + (1-\tau_i)(1-r_{x,i}))$. With $\widetilde{\mathsf{eq}}(\tau,r_x)$ in hand, $g(r_x)$ can be computed in $O(1)$ time given the four quantities: $\sum_{y\in\{0,1\}^s}\widetilde{A}(r_x,y)\cdot\widetilde{Z}(y)$, $\sum_{y\in\{0,1\}^s}\widetilde{B}(r_x,y)\cdot\widetilde{Z}(y)$, $\sum_{y\in\{0,1\}^s}\widetilde{C}(r_x,y)\cdot\widetilde{Z}(y)$, and $\widetilde{E}(r_x)$.

The last quantity can be computed with a single query to polynomial $\widetilde{E}$. Furthermore, the first three quantities can be computed by applying the sum-check protocol three more times in parallel, once to each of the following three polynomials (using the same random vector of field elements, $r_y \in \mathbb{F}^s$, in each of the three invocations):

$$\widetilde{A}(r_x,y)\cdot\widetilde{Z}(y),$$
$$\widetilde{B}(r_x,y)\cdot\widetilde{Z}(y),$$
$$\widetilde{C}(r_x,y)\cdot\widetilde{Z}(y).$$

To perform the verifier's final check in each of these three invocations of the sum-check protocol, it suffices for the verifier to evaluate each of the above 3 polynomials at the random vector r_y, which means it suffices for the verifier to evaluate $\widetilde{A}(r_x,r_y)$, $\widetilde{B}(r_x,r_y)$, $\widetilde{C}(r_x,r_y)$, and $\widetilde{Z}(r_y)$. The first three evaluations can be obtained via the verifier's assumed query access to $\widetilde{A}$, $\widetilde{B}$, and $\widetilde{C}$. $\widetilde{Z}(r_y)$ can be obtained from one query to $\widetilde{W}$ and one query to $\widetilde{(1,\mathsf{x})}$ via Equation (16).

In summary, we have the following polynomial IOP.

Construction 6 (Polynomial IOP for Indexed Relaxed R1CS). The verifier $\mathcal{V}$ holds the instance $\mathbb{x} = (\mathbb{F}, A, B, C, m, n, \ell, u, x)$ and has a query access to an index $\mathbb{I} = (E, W)$. The prover is given as input both $(\mathbb{I}, \mathbb{x})$.

1. $\mathcal{V} \rightarrow \mathcal{P}$: $\tau \in_R \mathbb{F}^s$
2. $\mathcal{V} \leftrightarrow \mathcal{P}$: run the sum-check reduction to reduce the check in Equation (17) to checking if the following hold, where r_x, r_y are vectors in $\mathbb{F}^s$ chosen at random by the verifier over the course of the sum-check protocol:
   – $\widetilde{A}(r_x, r_y) \overset{?}{=} v_A$, $\widetilde{B}(r_x, r_y) \overset{?}{=} v_B$, and $\widetilde{C}(r_x, r_y) \overset{?}{=} v_C$;
   – $\widetilde{E}(r_x) \overset{?}{=} v_E$; and
   – $\widetilde{Z}(r_y) \overset{?}{=} v_Z$.
3. $\mathcal{V}$:
   – check if $\widetilde{A}(r_x, r_y) \overset{?}{=} v_A$, $\widetilde{B}(r_x, r_y) \overset{?}{=} v_B$, and $\widetilde{C}(r_x, r_y) \overset{?}{=} v_C$, with a query to $\widetilde{A}, \widetilde{B}, \widetilde{C}$ at (r_x, r_y);
   – check if $\widetilde{E}(r_x) \overset{?}{=} v_E$ with an oracle query to $\widetilde{E}$; and
   – check if $\widetilde{Z}(r_y) \overset{?}{=} v_Z$ by checking if: $v_Z = (1 - r_y[1]) \cdot v_W + r_y[1] \cdot \widetilde{(\mathsf{x}, 1)}(r_y[2..])$, where $r_y[2..]$ refers to a slice of r_y without the first element of r_y, and $v_W \leftarrow \widetilde{W}(r_y[2..])$ via an oracle query (see Equation (16)).

Completeness. Perfect completeness follows from perfect completeness of the sum-check protocol and the fact that Equation (17) holds with probability 1 over the choice of τ if ϕ is satisfiable.

Soundness. Applying a standard union bound to the soundness error introduced by probabilistic check in Equation (17) with the soundness error of the sum-check protocol [LFKN90], we conclude that the soundness error for the depicted polynomial IOP as at most $O(\log m)/|\mathbb{F}|$.

Round and communication complexity. The sum-check protocol is applied 4 times (although 3 of the invocations occur in parallel and in practice combined into one [Set20]). In each invocation, the polynomial to which the sum-check protocol is applied has degree at most 3 in each variable, and the number of variables is $s = \log m$. Hence, the round complexity of the polynomial IOP is $O(\log m)$. Since each polynomial has degree at most 3 in each variable, the total communication cost is $O(\log m)$ field elements.

Verifier time. The asserted bounds on the verifier's runtime are immediate from the verifier's runtime in the sum-check protocol, and the fact that $\widetilde{\mathsf{eq}}$ can be evaluated at any input $(\tau, r_x) \in \mathbb{F}^{2s}$ in $O(\log m)$ field operations.

Prover Time. As in Spartan [Set20], the prover's computation in the polynomial IOP in $O(n)$ operations over $\mathbb{F}$ using prior techniques for linear-time sum-checks [Tha13, XZZ$^+$19]. This includes the time required to compute $\widetilde{E}(r_x)$, $\widetilde{W}(r_y)$, $\widetilde{A}$, $\widetilde{B}$, and $\widetilde{C}$. $\square$

7.4 Compiling polynomial IOPs to zkSNARKs

As in prior works [Set20, BFS20, CHM$^+$20], we compile the polynomial IOP into a zkSNARK using a polynomial commitment scheme.

Interpreting commitments to vectors as polynomial commitments. It is well known that commitments to m-sized vectors over $\mathbb{F}$ are commitments to $\log m$-variate multilinear polynomials represented with evaluations over $\{0,1\}^m$ [ZGK$^+$17, WTS$^+$18, Set20, Lee20]. Furthermore, there is a polynomial commitment scheme for $\log m$-variate multilinear polynomials if there exists an argument protocol to prove an inner product computation between a committed vector and an m-sized public vector $((r_1, 1-r_1) \otimes \ldots \otimes (r_{\log m}, 1-r_{\log m}))$, where $r \in \mathbb{F}^{\log m}$ is an evaluation point. There are two candidate constructions in the literature.

1. $\mathsf{PC}_{\mathsf{BP}}$. If the commitment scheme for vectors over $\mathbb{F}$ is Pedersen's commitments (Construction 1), as in prior work [WTS$^+$18], Bulletproofs [BBB$^+$18] provides a suitable inner product argument protocol. The polynomial commitment scheme here achieves the following efficiency characteristics, assuming the hardness of the discrete logarithm problem. For a $\log m$-variate multilinear polynomial, committing takes $O_\lambda(m)$ time to produce an $O_\lambda(1)$-sized commitment; the prover incurs $O_\lambda(m)$ costs to produce an evaluation proof of size $O_\lambda(\log m)$ that can be verified in $O_\lambda(m)$. Note that $\mathsf{PC}_{\mathsf{BP}}$ is a special case of Hyrax's polynomial commitment scheme [WTS$^+$18].
2. $\mathsf{PC}_{\mathsf{Dory}}$. If vectors over $\mathbb{F}$ are committed with a two-tiered "matrix" commitment (see for example, [BMMV19, Lee20]), which provides $O_\lambda(1)$-sized commitments to N-sized vectors under the SXDH assumption. With this commitment scheme, Dory [Lee20] provides the necessary inner product argument. The polynomial commitment here achieves the following efficiency characteristics, assuming the hardness of SXDH. For a $\log m$-variate multilinear polynomial, committing takes $O_\lambda(m)$ time to produce an $O_\lambda(1)$-sized commitment; the prover incurs $O_\lambda(m)$ costs to produce an evaluation proof of size $O_\lambda(\log m)$ that can be verified in $O_\lambda(\log m)$.

Note that the primary difference between two schemes is in the verifier's time.

Theorem 3. *Assuming the hardness of the discrete logarithm problem, there exists a non-interactive zero-knowledge argument of knowledge for indexed relaxed R1CS with the following efficiency characteristics, where m denotes the dimensions of the R1CS coefficient matrices and n denotes the number of non-zero entries in the matrices.*

- *The verifier's preprocessing time is $O_\lambda(n)$;*
- *The prover's preprocessing time is $O_\lambda(m)$;*
- *The prover runs in time $O_\lambda(n)$;*
- *The proof length is $O_\lambda(\log n)$; and*
- *The verifier runs in time $O_\lambda(n)$.*

Proof. Applying the compiler of [BFS20] to the polynomial IOP from Construction 6 using $\mathsf{PC}_{\mathsf{BP}}$ provides a public-coin interactive argument of knowledge for committed relaxed R1CS. Applying standard transformations to the resulting interactive argument provide zero-knowledge [CD98, WTS+18, Set20] and non-interactivity [FS86].

The claimed efficiency follows from the efficiency of the polynomial IOP and $\mathsf{PC}_{\mathsf{BP}}$. In more detail:

Preprocessing costs. In an indexed relaxed R1CS instance, the index $\mathbb{I} = (\mathbb{I}_{\mathcal{V}}, \mathbb{I}_{\mathcal{P}})$. The verifier runs the indexer on $\mathbb{I}_{\mathcal{V}}$ and the prover runs the indexer on $\mathbb{I}_{\mathcal{P}}$. $\mathbb{I}_{\mathcal{P}}$ consists of two $O(\log m)$-variate multilinear polynomials, so creating commitments to using $\mathsf{PC}_{\mathsf{BP}}$ incurs $O_\lambda(m)$ costs. $\mathbb{I}_{\mathcal{V}}$ consists of three $2 \log m$-variate sparse multilinear polynomials of size n. Using the SPARK compiler [Set20] (and its optimization [SL20]) applied to $\mathsf{PC}_{\mathsf{BP}}$, creating commitments to three polynomials in its index requires $O_\lambda(n)$ costs to the verifier.

Online costs. The prover's costs to participate in the polynomial IOP is $O(n)$. To prove evaluations of two $O(\log m)$-variate multilinear polynomials, it takes $O_\lambda(m)$ time, and to prove evaluations of three $2 \log m$-variate sparse multilinear polynomials of size n, using the SPARK compiler (and its optimization [SL20]) applied to $\mathsf{PC}_{\mathsf{BP}}$, it takes $O_\lambda(n)$ time. In total, the prover time is $O_\lambda(n)$.

The proof length in the polynomial IOP is $O(\log m)$, and the proof sizes in the polynomial evaluation proofs is $O_\lambda(\log n)$, so the proof length is $O_\lambda(\log n)$.

The verifier's time in the polynomial IOP is $O(\log m)$. In addition, it verifies five polynomial evaluations, which costs $O_\lambda(n)$ (the two polynomial in the prover's index take $O_\lambda(m)$ and the two polynomials in the verifier's index takes $O_\lambda(n)$ using the SPARK compiler along with its optimization [SL20] applied to $\mathsf{PC}_{\mathsf{BP}}$). So, in total, the verifier time is $O_\lambda(n)$. □

Optimization. In Theorem 3, the verifier preprocesses three sparse multilinear polynomials of size n. In $\mathsf{PC}_{\mathsf{BP}}$, given that the verifier's runtime to verify an evaluation is at least linear in the polynomial size, the verifier does not gain from preprocessing $\mathbb{I}_{\mathcal{V}}$. Avoiding this preprocessing step not only simplifies the protocol (since one does not need a polynomial commitment scheme for sparse multilinear polynomials), it also improves costs, both asymptotically and concretely.

Furthermore, when transforming a polynomial IOP for indexed relaxed R1CS to an interactive argument using a polynomial commitment scheme (as we do so later in the section), we obtain an interactive argument for committed relaxed R1CS since the polynomials in $\mathbb{I}_{\mathcal{P}}$ appear in the instance as (polynomial) commitments and the polynomials in $\mathbb{I}_{\mathcal{V}}$ appear in the instance as polynomials.

Corollary 1. *Assuming the hardness of the discrete logarithm problem, there exists a non-interactive zero-knowledge argument of knowledge for committed relaxed R1CS with the following efficiency characteristics, where m denotes the dimensions of the R1CS coefficient matrices and n denotes the number of non-zero entries in the matrices.*

– *The prover's preprocessing time is $O_\lambda(m)$;*
– *The prover runs in time $O(n) + O_\lambda(m)$;*
– *The proof length is $O_\lambda(\log m)$; and*
– *The verifier runs in time $n + O_\lambda(m)$.*

The proof of the following theorem is identical to the proof of Theorem 3, with analysis adjusted to use $\mathsf{PC}_{\mathsf{Dory}}$ instead of $\mathsf{PC}_{\mathsf{BP}}$.

Theorem 4. *Assuming the hardness of the SXDH problem, there exists a non-interactive zero-knowledge argument of knowledge for indexed relaxed R1CS with the following efficiency characteristics, where m denotes the dimensions of the R1CS coefficient matrices and n denotes the number of non-zero entries in the matrices.*

– *The verifier's preprocessing time is $O_\lambda(n)$;*
– *The prover's preprocessing time is $O_\lambda(m)$;*
– *The prover runs in time $O_\lambda(n)$;*
– *The proof length is $O_\lambda(\log n)$; and*
– *The verifier runs in time $O_\lambda(\log n)$;*

Proof. Applying the compiler of [BFS20] to the polynomial IOP from Construction 6 using $\mathsf{PC}_{\mathsf{Dory}}$ provides a public-coin interactive argument of knowledge for committed relaxed R1CS. Applying standard transformations to the resulting interactive argument provide zero-knowledge [CD98, WTS$^+$18, Set20] and non-interactivity [FS86].

The claimed efficiency follows from the efficiency of the polynomial IOP and $\mathsf{PC}_{\mathsf{Dory}}$. In more detail:

Preprocessing costs. In an indexed relaxed R1CS instance, the index $\mathbb{I} = (\mathbb{I}_{\mathcal{V}}, \mathbb{I}_{\mathcal{P}})$. The verifier runs the indexer on $\mathbb{I}_{\mathcal{V}}$ and the prover runs the indexer on $\mathbb{I}_{\mathcal{P}}$. $\mathbb{I}_{\mathcal{P}}$ consists of two $O(\log m)$-variate multilinear polynomials, so creating commitments to using $\mathsf{PC}_{\mathsf{Dory}}$ incurs $O_\lambda(m)$ costs. $\mathbb{I}_{\mathcal{V}}$ consists of three $2\log m$-variate sparse multilinear polynomials of size n. Using the SPARK compiler [Set20] (and its optimization [SL20]) applied to $\mathsf{PC}_{\mathsf{Dory}}$, creating commitments to three polynomials in its index requires $O_\lambda(n)$ costs to the verifier.

Online costs. The prover's costs to participate in the polynomial IOP is $O(n)$. To prove evaluations of two $O(\log m)$-variate multilinear polynomials, it takes $O_\lambda(m)$ time, and to prove evaluations of three $2\log m$-variate sparse multilinear polynomials of size n, using the SPARK compiler (and its optimization [SL20]) applied to $\mathsf{PC}_{\mathsf{Dory}}$, it takes $O_\lambda(n)$ time. In total, the prover time is $O_\lambda(n)$.

The proof length in the polynomial IOP is $O(\log m)$, and the proof sizes in the polynomial evaluation proofs is $O_\lambda(\log n)$, so the proof length is $O_\lambda(\log n)$.

The verifier's time in the polynomial IOP is $O(\log m)$. In addition, it verifies five polynomial evaluations, which costs $O_\lambda(\log n)$ (the two polynomial in the prover's index take $O_\lambda(\log m)$ and the two polynomials in the verifier's index takes $O_\lambda(\log n)$ using the SPARK compiler along with its optimization [SL20] applied to $\mathsf{PC}_{\mathsf{Dory}}$). So, in total, the verifier time is $O_\lambda(\log n)$. $\square$

Acknowledgments

We thank Justin Thaler for comments on a prior version of this paper.

References

[AC10] Eran Tromer Alessandro Chiesa. Proof-carrying data and hearsay arguments from signature cards. In *Innovations in Computer Science (ICS)*, 2010.

[BBB$^+$18] Benedikt Bünz, Jonathan Bootle, Dan Boneh, Andrew Poelstra, Pieter Wuille, and Greg Maxwell. Bulletproofs: Short proofs for confidential transactions and more. In *S&P*, 2018.

[BBF18] Dan Boneh, Benedikt Bünz, and Ben Fisch. A survey of two verifiable delay functions. Cryptology ePrint Archive, Report 2018/712, 2018.

[BCC$^+$16] Jonathan Bootle, Andrea Cerulli, Pyrros Chaidos, Jens Groth, and Christophe Petit. Efficient zero-knowledge arguments for arithmetic circuits in the discrete log setting. In *EUROCRYPT*, 2016.

[BCCT13] Nir Bitansky, Ran Canetti, Alessandro Chiesa, and Eran Tromer. Recursive composition and bootstrapping for SNARKs and proof-carrying data. In *STOC*, 2013.

[BCL$^+$20] Benedikt Bünz, Alessandro Chiesa, William Lin, Pratyush Mishra, and Nicholas Spooner. Proof-carrying data without succinct arguments. Cryptology ePrint Archive, Report 2020/1618, 2020.

[BCMS20] Benedikt Bünz, Alessandro Chiesa, Pratyush Mishra, and Nicholas Spooner. Proof-carrying data from accumulation schemes. In *TCC*, 2020.

[BCR$^+$19] Eli Ben-Sasson, Alessandro Chiesa, Michael Riabzev, Nicholas Spooner, Madars Virza, and Nicholas P. Ward. Aurora: Transparent succinct arguments for R1CS. In *EUROCRYPT*, 2019.

[BCS16] Eli Ben-Sasson, Alessandro Chiesa, and Nicholas Spooner. Interactive Oracle Proofs. In *TCC*, 2016.

[BCTV14] Eli Ben-Sasson, Alessandro Chiesa, Eran Tromer, and Madars Virza. Scalable zero knowledge via cycles of elliptic curves. In *CRYPTO*, 2014.

[BDFG20] Dan Boneh, Justin Drake, Ben Fisch, and Ariel Gabizon. Halo Infinite: Recursive zk-SNARKs from any Additive Polynomial Commitment Scheme. Cryptology ePrint Archive, Report 2020/1536, 2020.

[BFS20] Benedikt Bünz, Ben Fisch, and Alan Szepieniec. Transparent SNARKs from DARK compilers. In *EUROCRYPT*, 2020.

[BGH19] Sean Bowe, Jack Grigg, and Daira Hopwood. Halo: Recursive proof composition without a trusted setup. Cryptology ePrint Archive, Report 2019/1021, 2019.

[BGH20] Sean Bowe, Jack Grigg, and Daira Hopwood. Halo2, 2020. https://github.com/zcash/halo2.

[BMMV19] Benedikt Bünz, Mary Maller, Pratyush Mishra, and Noah Vesely. Proofs for inner pairing products and applications. Cryptology ePrint Archive, Report 2019/1177, 2019.

[CD98] Ronald Cramer and Ivan Damgård. Zero-knowledge proofs for finite field arithmetic, or: Can zero-knowledge be for free? In *CRYPTO*, pages 424–441, 1998.

[CHM$^+$20] Alessandro Chiesa, Yuncong Hu, Mary Maller, Pratyush Mishra, Noah Vesely, and Nicholas Ward. Marlin: Preprocessing zkSNARKs with universal and updatable SRS. In *EUROCRYPT*, 2020.

[COS20] Alessandro Chiesa, Dev Ojha, and Nicholas Spooner. Fractal: Post-quantum and transparent recursive proofs from holography. In *EUROCRYPT*, 2020.

[FS86] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *CRYPTO*, pages 186–194, 1986.

[GGPR13] Rosario Gennaro, Craig Gentry, Bryan Parno, and Mariana Raykova. Quadratic span programs and succinct NIZKs without PCPs. In *EUROCRYPT*, 2013.

[GMR85] S Goldwasser, S Micali, and C Rackoff. The knowledge complexity of interactive proof-systems. In *STOC*, 1985.

[Gro16] Jens Groth. On the size of pairing-based non-interactive arguments. In *EUROCRYPT*, 2016.

[Kil92] Joe Kilian. A note on efficient zero-knowledge proofs and arguments (extended abstract). In *STOC*, 1992.

[Lab20] O(1) Labs. Mina cryptocurrency, 2020. https://minaprotocol.com.

[Lee20] Jonathan Lee. Dory: Efficient, transparent arguments for generalised inner products and polynomial commitments. Cryptology ePrint Archive, Report 2020/1274, 2020.

[LFKN90] Carsten Lund, Lance Fortnow, Howard Karloff, and Noam Nisan. Algebraic methods for interactive proof systems. In *FOCS*, October 1990.

[LNS20] Jonathan Lee, Kirill Nikitin, and Srinath Setty. Replicated state machines without replicated execution. In *S&P*, 2020.

[LSTW21] Jonathan Lee, Srinath Setty, Justin Thaler, and Riad Wahby. Linear-time zero-knowledge SNARKs for R1CS. Cryptology ePrint Archive, Report 2021/030, 2021.

[Mic94] Silvio Micali. CS proofs. In *FOCS*, 1994.

[PGHR13] Bryan Parno, Craig Gentry, Jon Howell, and Mariana Raykova. Pinocchio: Nearly practical verifiable computation. In *S&P*, May 2013.

[RRR16] Omer Reingold, Guy N. Rothblum, and Ron D. Rothblum. Constant-round interactive proofs for delegating computation. In *STOC*, pages 49–62, 2016.

[SAGL18] Srinath Setty, Sebastian Angel, Trinabh Gupta, and Jonathan Lee. Proving the correct execution of concurrent services in zero-knowledge. In *OSDI*, October 2018.

[SBV$^+$13] Srinath Setty, Benjamin Braun, Victor Vu, Andrew J. Blumberg, Bryan Parno, and Michael Walfish. Resolving the conflict between generality and plausibility in verified computation. In *EuroSys*, April 2013.

[Set20] Srinath Setty. Spartan: Efficient and general-purpose zkSNARKs without trusted setup. In *CRYPTO*, 2020.

[SL20] Srinath Setty and Jonathan Lee. Quarks: Quadruple-efficient transparent zkSNARKs. Cryptology ePrint Archive, Report 2020/1275, 2020.

[Tha13] Justin Thaler. Time-optimal interactive proofs for circuit evaluation. In *CRYPTO*, 2013.

[Tha20] Justin Thaler. Proofs, arguments, and zero-knowledge. http://people.cs.georgetown.edu/jthaler/ProofsArgsAndZK.html, 2020.

[Val08] Paul Valiant. Incrementally verifiable computation or proofs of knowledge imply time/space efficiency. In *TCC*, pages 552–576, 2008.

[Wes19] Benjamin Wesolowski. Efficient verifiable delay functions. In *EUROCRYPT*, pages 379–407, 2019.

[WJB$^+$17] Riad S. Wahby, Ye Ji, Andrew J. Blumberg, Abhi Shelat, Justin Thaler, Michael Walfish, and Thomas Wies. Full accounting for verifiable outsourcing. In *CCS*, 2017.

[WTS$^+$18] Riad S. Wahby, Ioanna Tzialla, Abhi Shelat, Justin Thaler, and Michael Walfish. Doubly-efficient zkSNARKs without trusted setup. In *S&P*, 2018.

[XZZ$^+$19] Tiancheng Xie, Jiaheng Zhang, Yupeng Zhang, Charalampos Papamanthou, and Dawn Song. Libra: Succinct zero-knowledge proofs with optimal prover computation. In *CRYPTO*, 2019.

[ZGK$^+$17] Yupeng Zhang, Daniel Genkin, Jonathan Katz, Dimitrios Papadopoulos, and Charalampos Papamanthou. vSQL: Verifying arbitrary SQL queries over dynamic outsourced databases. In *S&P*, 2017.

Supplementary Materials

A Proof of Theorem 1 (Forking Lemma for Folding Schemes)

The proof for our variant of the forking lemma closely follows that of Bootle et al. [BCC+16]. We present a simplified version of the proof below adapted for folding schemes.

Proof. Consider $(2\mu + 1)$-move folding scheme $(\mathcal{P}, \mathcal{V})$ for relation $\mathcal{R}$ with randomness sampled uniformly from $\mathbb{F}$ where $|\mathbb{F}| \geq 2^{\lambda}$. Suppose there exists PPT $\mathcal{X}$ such that for arbitrary inputs u_1, u_2, outputs satisfying witness w_1, w_2 with probability $1 - \mathsf{negl}(\lambda)$, given an $(n_1, \ldots, n_\mu)$-tree of accepting transcripts and corresponding folded instance-witness pairs where $\prod_{i=1}^{\mu} n_i$. Consider arbitrary prover $\mathcal{P}^*$ and let ϵ be the probability that, on inputs u_1, u_2, $\mathcal{P}^*$ succeeds in producing a valid witness w for folded instance u. To prove that $(\mathcal{P}, \mathcal{V})$ satisfies knowledge-soundness we will show that there exists extractor $\mathcal{E}$ that outputs valid witness w_1, w_2 given input instance u_1, u_2 and prover randomness ρ with probability $\epsilon - \mathsf{negl}(\lambda)$.

We first construct extractor $\mathcal{E}$ as follows:

$\underline{\mathcal{E}(u_1, u_2, \rho) \to (w_1, w_2)}$:

1. Compute $\mathsf{tree} \leftarrow T(1)$.
2. If tree is not a valid $(n_1, \ldots, n_\mu)$-tree (i.e. there are collisions in the verifier's randomness) return $\bot$.
3. Output $(w_1, w_2) \leftarrow \mathcal{X}(u_1, u_2, \mathsf{tree})$.

where the function T is defined as follows:

$\underline{\mathcal{T}(i) \to \mathsf{tree}}$:

1. Sample the verifier's randomness for round i, r_i.
2. If $i = \mu + 1$, compute
$$(u, w) \leftarrow \langle \mathcal{P}^*(\rho), \mathcal{V} \rangle (u_1, u_2)$$
and let tr be the corresponding transcript. If $(u, w) \in \mathcal{R}$, output $\{(\mathsf{tr}, (u, w))\}$. Otherwise output $\bot$.
3. With fixed verifier randomness $(r_1, \ldots, r_i)$, compute $\mathsf{tree} \leftarrow \mathcal{T}(i + 1)$ *once*. If $\mathsf{tree} = \bot$ output $\bot$.
4. With fixed verifier randomness $(r_1, \ldots, r_i)$, repeatedly run $\mathsf{T}(i + 1)$ until $n_i - 1$ additional lists of accepting transcripts are acquired. Append all the results to tree.
5. Output tree.

Let E_1 denote the event that $\mathcal{T}(1)$ outputs $\mathsf{tree} \neq \bot$ in less than T time steps (we will specify T later). Given E_1, let E_2 denote the event that the resulting tree is a valid $(n_1, \ldots, n_\mu)$-tree (i.e. there are no collisions in the verifier's randomness). Given E_1 and E_2, let E_3 denote the event that $\mathcal{X}$ succeeds with tree as input. Then, we have that $\mathcal{E}$ succeeds with probability

$$P_{\mathcal{E}} = \Pr\left[E_3\right] \cdot \Pr\left[E_2\right] \cdot \Pr\left[E_1\right].$$

We will compute each of these probabilities. To compute $\Pr\left[E_1\right]$, we start by observing that $T(1)$ fails (i.e. returns $\bot$) only when its *first* call to $T(2)$ fails. Likewise $T(2)$ fails only when its first call to $T(3)$ fails. Chaining these assertions, we have that $T(1)$ fails with probability $(1-\epsilon)$, which, by assumption, is the probability that $T(\mu+1)$ fails Thus, the expected number of times $T(i)$ calls $T(i+1)$ is

$$\begin{aligned}
&1 + \Pr\left[\text{First call to } T(i+1) \text{ succeeds}\right] \cdot \frac{(n_i - 1)}{\Pr\left[T(i+1) \text{ returns } \mathsf{tree} \neq \bot\right]} \\
&= 1 + \epsilon \cdot \frac{(n_i - 1)}{\epsilon} \\
&= n_i.
\end{aligned}$$

Hence, the total runtime is expected to be $t = O(\prod_{i=1}^{\mu} n_i)$ which is bounded above by $\mathsf{poly}(\lambda)$ by assumption. Moreover, by Markov's inequality, we have that $\mathcal{T}$ runs for time longer than $T > t$ with probability $\frac{t}{T}$. Thus we have that

$$\Pr[E_1] = (1 - \frac{t}{T}) \cdot \epsilon.$$

Given E_1 we have that $\mathcal{T}$ runs in at most T time-steps. This ensures that there are at most T random challenges produced for the verifier, and that the probability of collision is at most $\frac{T^2}{|\mathbb{F}|}$. Thus we have that

$$\Pr[E_2] = 1 - \frac{T^2}{|\mathbb{F}|}$$

Finally, given E_1 and E_2, we have that $\Pr[E_3] = (1 - \mathsf{negl}(\lambda))$ by assumption. Consolidating our calculations we have that $\mathcal{E}$ succeeds with probability

$$P_{\mathcal{E}} = (1 - \mathsf{negl}(\lambda)) \cdot \left(1 - \frac{T^2}{|\mathbb{F}|}\right) \cdot \left(1 - \frac{t}{T}\right) \cdot \epsilon$$

Setting $T = \sqrt[3]{|\mathbb{F}|}$ (and assuming that $T \geq t$) we have that

$$\begin{aligned}
P_{\mathcal{E}} &= (1 - \mathsf{negl}(\lambda)) \cdot \left(1 - \frac{1}{\sqrt[3]{|\mathbb{F}|}}\right) \cdot \left(1 - \frac{t}{\sqrt[3]{|\mathbb{F}|}}\right) \cdot \epsilon \\
&= \epsilon - \mathsf{negl}(\lambda)
\end{aligned}$$

□