Mova: Nova folding without committing to error terms

Nikolaos Dimitriou[1], Albert Garreta[1], Ignacio Manzur[1], and Ilia Vlasov[1]

[1]Nethermind Research, {nikolaos,albert,ignacio,ilia}@nethermind.io

Abstract

We present Mova, a folding scheme for R1CS instances that does not require committing to error or cross terms, nor makes use of the sumcheck protocol. We compute concrete costs and provide benchmarks showing that, for reasonable parameter choices, Mova's Prover is about 5 to 10 times faster than Nova's Prover, and between 1.05 to 1.3 times faster than Hypernova's Prover (applied to R1CS instances) – assuming the R1CS witness vector contains only small elements. Mova's Verifier has a similar cost as Hypernova's Verifier, but Mova has the advantage of having only 3 rounds of communication, while Hypernova has a logarithmic number of rounds.

Mova, which is based on the Nova folding scheme, manages to avoid committing to Nova's so-called error term $\mathbf{E}$ and cross term $\mathbf{T}$ by replacing said commitments with evaluations of the Multilinear Extension (MLE) of $\mathbf{E}$ and $\mathbf{T}$ at a random point sampled by the Verifier. A key observation used in Mova's soundness proofs is that $\mathbf{E}$ is implicitly committed by a commitment to the input-witness vector $\mathbf{Z}$, since $\mathbf{E} = (A \cdot \mathbf{Z}) \circ (B \cdot \mathbf{Z}) - u(C \cdot \mathbf{Z})$.

We also note that ProtoGalaxy [EG23] can be specialised to an R1CS folding scheme with similar properties. Some of our further contributions are that 1) Mova is described with a language that sheds new insights into the topic of "Nova-style folding"; 2) we provide concrete costs, benchmarks, and optimisations for the Prover; 3) we describe how to fold two accumulated instances (which is important for applications in Proof Carrying Data); and 4) provide non-trivial knowledge soundness proofs in the context of multilinear polynomials.

Contents

1 Introduction

Folding schemes have been an active area of research in the last years [BGH19, BCMS20, BCL+20, KST21, KS23b, BC23, EG23, BC24b]. Informally, these schemes can be described as interactive proofs in which a Prover and Verifier create a new instance-witness pair in a certain relation $\mathbf{R}_2$ from two instance-witness pairs in relations $\mathbf{R}_1, \mathbf{R}_2$. The validity of the newly created instance-witness pair implies the validity of the original instance-witness pair, except with negligible probability. The idea is that if this combination process is less expensive than proving an individual instance-witness pair (in prover time, memory requirements, or proof size), one can save costs by reducing the task of proving many instance-witness pairs to proving a single pair. Initially, folding schemes were created with the intention of improving the construction of primitives like Incrementally Verifiable Computation (IVC) [Val08] and Proof-Carrying-Data (PCD) [CT10].

Nova [KST21] is a celebrated folding scheme due to Kothapalli, Setty, and Tzialla where $\mathbf{R}_1$ is the R1CS relation, and $\mathbf{R}_2$ is the committed relaxed R1CS relation. Nova features only one round of communication, a very lean Verifier with $O(1)$ complexity, and a relatively simple and efficient Prover. For the purposes of this paper, the main drawback of Nova is the need for the Prover to compute a commitment to a vector $\mathbf{T}$ with arbitrarily large entries, i.e. a vector whose entries have roughly $\log(|\mathbb{F}|)$ bits, where $\mathbb{F}$ is the underlying finite field used in $\mathbf{R}_1$ and $\mathbf{R}_2$. Nova requires using a homomorphic commitment scheme, which in practical scenarios is typically chosen to be an elliptic curve-based scheme such as Pedersen or KZG. Then, to compute the commitment to $\mathbf{T}$, the Prover ultimately has to perform a Multiscalar Multiplication (MSM) of a vector with large entries. This is a costly operation that dominates by far the rest of the Prover's costs (assuming the R1CS matrices are reasonably sparse, and not counting the cost of committing to the R1CS witness – see below for a discussion on this). For reference, over the Pallas curve, Table 1 in [Hab22] estimates the cost to be, roughly, $349|\mathbf{T}|$ field operations when $|\mathbf{T}| = 2^{16}$, and $256|\mathbf{T}|$ field operations when $|\mathbf{T}| = 2^{20}$.

Notice that, in all folding schemes discussed in this work, the witness in $(\mathbb{x}, \mathbb{w}) \in \mathbf{R}_1$ always needs to be committed. In many practical scenarios[1], the witness is a vector that contains small entries, i.e. much smaller than the size of the underlying field $\mathbb{F}$. In that case, as observed in [STW23b] and in our benchmarks in Table 7, an MSM for such a vector can easily be an order of magnitude cheaper than an MSM with a vector containing arbitrarily large entries. Hence, for such applications, removing or lowering the cost of committing to $\mathbf{T}$ in Nova can lead to dramatic efficiency improvements of the system where Nova is being used. Otherwise, in an application where witnesses have arbitrarily sized entries, removing the commitment to $\mathbf{T}$ in Nova has less of an impact (still significant though, cf. Tables 2 and 8), as already the overall system is paying for expensive commitments (cf. Section 5).

Hypernova [KS23b] is a folding scheme due to Kothapalli and Setty where, unlike in Nova, the Prover does not commit to any vectors (within the folding scheme). However, Hypernova performs $\log(m)$ rounds of communication, as opposed to Nova, which performs only 1. This difference is significant when using folding schemes to enable IVC or PCD. This is because in that scenario, at each folding step, besides running the folding protocol, the Prover also has to create a proof that the folding is done correctly. Then, each round of communication of the folding scheme translates into the need to create a proof of correct hashing (where the hash is used as a replacement of the Verifier's randomness). Thus, having a large number of communication rounds can introduce significant overheads when using folding in IVC/PCD.

We remark that Hypernova has the benefit of being usable also for folding instance-witness pairs from the CCS [STW23a] relation, which is a high degree generalisation of the R1CS relation. Nova cannot efficiently handle this relation when the degree d is large, due to certain costs scaling by a quadratic factor on d.

Since their publication, many subsequent works have built upon Nova and Hypernova [BC24a, NDC$^+$24, ZGGX23, BC23, EG23, KS23a, ZZD23, Sou24]. We refer to [Awe] for a vaired colletion of resources on the topic of folding.

In this work we investigate the the following (loosely formulated) question:

Does there exist efficient folding schemes for the R1CS relation where the Prover does not commit to any vector besides the commitment to the R1CS witness, and with $O(1)$ rounds of communication?

We describe and benchmark a scheme that answers this question affirmatively, and which we call Mova[2]. We also identify ProtoGalaxy as a scheme that can be specialised so as to solve the above question (see below).

ProtoGalaxy Upon completion of this work, the authors of [EG23] brought to our attention that the ProtoGalaxy scheme can be specialised in a way that resolves the above

[1]For example, when proving statements involving non-native arithmetic, in which numbers are decomposed into small chunks (or limbs),

[2]Mova stands for Nova, with an "M" stemming from the fact that the commitment to Nova's error and cross term vectors are replaced by evaluations of their **M**ultilinear Extension (MLE) at a random point.

question affirmatively as well. In fact, our Mova scheme can be seen as a variation of this specialisation, in which, among others, the multilinear Lagrange base is used, as opposed to the monomial base. To obtain such specialisation, one should apply ProtoGalaxy on the trivial special sound protocol for the R1CS relation where the Prover simply sends the witness to the Verifier, and the Verifier checks whether the witness is valid. Then, the $\mathsf{pow}_i(\mathbf{X})$ polynomials from [EG23] play the role of our $\widetilde{\mathsf{eq}}(i;\mathbf{X})$ polynomials.

In [EG23], ProtoGalaxy is presented with a very different language and with different goals than ours. Indeed, the scheme in [EG23] is described as being a Protostar-based [BC23] folding scheme for special sound protocols, featuring an efficient Verifier, and with capacity for folding multiple instances at once.

We believe that our description and benchmarking of Mova both provides new insights into the topic of "Nova-style folding" (cf. Section 2), and highlights further attractive properties and applications of ProtoGalaxy. These insights are essentially that the commitment to the error term in Nova is redundant in many steps of the scheme, and that, whenever this commitment is needed, one can use alternative, less expensive, methods.

We also: 1) provide specific Prover optimisations and cost counts for our use-case (cf. Section 5.2), resulting in a especially lean protocol for folding R1CS instances (cf. Section 5); 2) allow for efficient R1CS-based Proof Carrying Data (PCD) by describing how to fold two accumulated instances – this is necessary in PCD due to its inherent non-linear recursion topology (we believe that similar methods can be used to extend [EG23]); and 3) provide non-trivial knowledge soundness proofs in the context of the multilinear Lagrange basis, whose principles may be applicable in other similar proofs (cf. Remark 4.3).

1.1 Mova in a nutshell

In Mova, $\mathbf{R}_1 = \mathbf{R}_{\mathsf{R1CS}}$ is the R1CS relation (cf. Eq. (1)) and $\mathbf{R}_2 = \mathbf{R}_{\mathsf{acc}}$ is, essentially, the committed relaxed R1CS relation $\mathbf{R}_{\mathsf{rcR1CS}}$ of Nova (cf. Eq. (2)), except that the commitment to the error term $\mathbf{E}$ is replaced by an evaluation v of the MLE $\mathsf{mle}[\mathbf{E}](\mathbf{X})$ of $\mathbf{E}$ at a point $\mathbf{r}$, i.e. the constraint $\mathsf{Com}(\mathbf{E}) = \bar{\mathbf{E}}$ from Nova's committed relaxed R1CS relation is replaced by the constraint $\mathsf{mle}[\mathbf{E}](\mathbf{r}) = v$. Here $\mathbf{r}, v$ are part of the public instance, see Eq. (3) for a full description of $\mathbf{R}_{\mathsf{acc}}$.

At a high level, the reason why Mova does not require commitments to the error term $\mathbf{E}$ is because, intuitively speaking, $\mathbf{E}$ is implicitly committed by the commitment to the instance-witness vector $\mathbf{Z}$ (or to the underlying witness). This is because $\mathbf{E}$ is uniquely determined by such vector, indeed: $\mathbf{E} = A \cdot \mathbf{Z} \circ B \cdot \mathbf{Z} - uC \cdot \mathbf{Z}$ (see Eq. (2) for the details). Hence, one can see that, when it comes to folding instance-witness pairs of the committed relaxed R1CS relation, it is actually possible to remove the commitments to $\mathbf{E}$ altogether, without any further modification. This is explained in more detail in the first part of Section 2.1.

However, this is not enough when aiming to fold instance-witness pairs for the R1CS relation $\mathbf{R}_{\mathsf{R1CS}}$ with instance-witness pairs for the relaxed R1CS relation $\mathbf{R}_{\mathsf{rcR1CS}}$. This is because the first instance-witness pair, say $(\mathbb{x}, \mathbb{w})$, needs to first be transformed into an instance-witness pair from $\mathbf{R}_{\mathsf{rcR1CS}}$. In Nova, this is achieved by simply adding an error term $\mathbf{E}$ which equals zero, and adding the constraint that $\mathsf{Com}(\mathbf{E})$ is a commitment to the zero vector. Since our goal is to not use commitments to $\mathbf{E}$, we instead have the Verifier send a random point $\mathbf{r} \in \mathbb{F}^{\log m}$, and add the constraint that $\mathsf{mle}[\mathbf{E}](\mathbf{r}) = 0$. This is the reason why our accumulated relation $\mathbf{R}_{\mathsf{acc}}$ is precisely the committed relaxed R1CS relation $\mathbf{R}_{\mathsf{rcR1CS}}$, after removing all mentions of a commitment to $\mathbf{E}$, and adding the constraint $\mathsf{mle}[\mathbf{E}](\mathbf{r}) = v$ for some public values $\mathbf{r}, v$. Precisely (we use blue to highlight the differences with the standard committed relaxed R1CS relation),

$$\mathbf{R}_{\mathsf{acc}} = \left\{ (\mathbb{x}; \mathbb{w}) = (\mathbf{x}, \overline{\mathbf{W}}, u, \ell_{\mathbf{E}}, \mathbf{r}; \mathbf{W}, \mathbf{E}) \;\middle|\; \begin{array}{l} (A \cdot \mathbf{Z}) \circ (B \cdot \mathbf{Z}) = u \cdot (C \cdot \mathbf{Z}) + \mathbf{E}, \\ \mathbf{Z} = (\mathbf{W}, \mathbf{x}, 1), \\ \mathsf{Com}(\mathbf{W}) = \overline{\mathbf{W}},\ \mathsf{mle}[\mathbf{E}](\mathbf{r}) = \ell_{\mathbf{E}}, \\ \mathbf{x} \in \mathbb{F}^{\ell}, \mathbf{W} \in \mathbb{F}^{m-\ell-1}, \mathbf{r} \in \mathbb{F}^{\log n} \end{array} \right\}.$$

This aspect of Mova is explained in more detail in our technical overview (cf. Protocol 3), or in full formality in Section 4.1.

We are almost done now. It remains to design a way of folding two instance-witness pairs from $\mathbf{R}_{\mathsf{acc}}$, say $(\mathbb{x}_1; \mathbb{w}_1)$ and $(\mathbb{x}_2; \mathbb{w}_2)$. Say the MLE evaluation point in these instances is $\mathbf{r}_1$ and $\mathbf{r}_2$, respectively. It is relatively simple to see that, if $\mathbf{r}_1 = \mathbf{r}_2$, then Nova's scheme works as is, replacing any operation with the commitments to errors or cross terms with the evaluation of the MLE of these terms at $\mathbf{r}_1 = \mathbf{r}_2$. See Protocol 2 for further details, or Section 4.3.

Finally, if $\mathbf{r}_1 \neq \mathbf{r}_2$ (which, in general, will always be the case), we employ a reduction of knowledge whereby the two claims $\mathsf{mle}[\mathbf{E}_1](\mathbf{r}_1) = v_1$ and $\mathsf{mle}[\mathbf{E}_2](\mathbf{r}_2) = v_2$ are transformed into two claims of the form $\mathsf{mle}[\mathbf{E}_1](\mathbf{r}) = v_1'$ and $\mathsf{mle}[\mathbf{E}_2](\mathbf{r}) = v_2'$ for some new $\mathbf{r}', v_1', v_2'$. At this point, we have reached the case above, where $\mathbf{r}_1 = \mathbf{r}_2$, and we can proceed accordingly. This reduction of knowledge is based on a technique used in GKR, in which the polynomials $\mathsf{mle}[\mathbf{E}_i]$ are composed with a line passing through the points $\mathbf{r}_1$ and $\mathbf{r}_2$. This is explained in more detail in Protocol 4 from our technical overview, and in full formality in Section 4.2.

1.2 Mova compared to Nova and Hypernova

In Table 1 we provide the concrete dominating costs of Mova, Nova, and Hypernova when used on the same R1CS structure (cf. Table 3 for more detailed costs). Later in the paper, in Table 4, we see that, for reasonable concrete parameter choices, Mova's Prover is roughly 5 to 10 times more efficient than Nova's Prover, and roughly 1.2 to 1.4 times more efficient than Hypernova's, assuming the entries in the R1CS witness vector $\mathbf{W}$ are small, by which we mean they all belong to the range $\{0, \ldots, |\mathbf{W}| - 1\}$. If the entries of $\mathbf{W}$ are arbitrarily large, then the improvement is of a factor between 1.8 to 2.8 in Nova, and of about 1.02 to 1.1 in Hypernova (cf. Table 4 for some details).

On the other hand, Mova's Verifier concrete cost is similar to Hypernova's, but Mova has only 3 rounds of communication, as opposed to Hypernova, which has $\log(m)$.

We implemented Mova and benchmarked its Prover against Nova and Hypernova's Prover, corroborating the above findings for the case where the R1CS matrices A, B, C

[3]For the Prover's costs, we only display the number of field multiplications, as these can be up to an order of magnitude more expensive than field additions. For the Verifier's costs, we display the number of additions and multiplications because, in many applications such as IVC or PCD, V's algorithm will be arithmetised and proved recursively.

	P	V	Round(s)
Nova [KST21]	$3n + 5m\,\mathbb{F}$ $2\,\mathbb{G}$ ops., $2\,\mathbb{G}$ exp. Com. vector of m elements in $\mathbb{F}$ Com. $\mathbf{W}$	$2\ell\,\mathbb{F}$ $2\,\mathbb{G}$ ops., $2\,\mathbb{G}$ exp.	1
Hypernova [KS23b]	$6n + 14m\ \mathbb{F}$ $1\,\mathbb{G}$ op., $1\,\mathbb{G}$ exp. Com. $\mathbf{W}$	$2\ell + O(\log(m))\ \mathbb{F}$ $1\,\mathbb{G}$ op., $1\,\mathbb{G}$ exp.	$\log(m) + O(1)$
Mova (this work)	$3n + 12m\ \mathbb{F}$ $1\,\mathbb{G}$ op., $1\,\mathbb{G}$ exp. Com. $\mathbf{W}$	$2\ell + 7\log(m) + 5\,\mathbb{F}$ $1\,\mathbb{G}$ op., $1\,\mathbb{G}$ exp.	3

Table 1: Dominating concrete costs of Mova, Nova, and Hypernova. Here m is the size of the vectors used in the various R1CS relations. In particular, the R1CS matrices A, B, C are $m \times m$ matrices. n denotes the number of nonzero entries in each of these matrices. ℓ is the size of the public input vector. The $\mathbb{F}$ symbol indicates the number of field multiplications in column 1, and the number of field additions and multiplications in column 2[3]. The $\mathbb{G}$ symbol indicates the number of group operations or exponentiations. We write "Com. a elements in $\mathbb{F}$" to mean that a commitment to a field elements of bit-size roughly $\log(|\mathbb{F}|)$ must be made. "Com. $\mathbf{W}$" means that the witness vector $\mathbf{W}$ of the R1CS relation has to be committed.

are the identity matrix. We remark that the performance improvement is expected to be less steep when A, B, C are more complicated matrices (cf. Table 4 and Remark 5.1). Running benchmarks for more general matrices is currently work-in-progress of ours. Further work-in-progress is to implement the optimised method for computing the cross-term $\mathbf{T}$, described in Section 5.2.

A simplified report of the results is provided in Table 2. Further details can be found in Section 5.1. We have made the code publicly available[4].

2 Techniques

In this section we provide a technical overview of how and why Mova works.

2.1 The Mova folding scheme

Fix a finite field $\mathbb{F}$ and three $m \times m$ matrices A, B, C with $n = \Omega(m)$ nonzero entries in $\mathbb{F}$. Let Com be an additively homomorphic vector commitment scheme. Define the R1CS relation as

$$\mathbf{R}_{\mathsf{cR1CS}} = \left\{ (\mathbb{x}; \mathbb{w}) = (\mathbf{x}; \mathbf{W}) \;\middle|\; \begin{array}{l} (A \cdot \mathbf{Z}) \circ (B \cdot \mathbf{Z}) = (C \cdot \mathbf{Z}), \\ \mathbf{Z} = (\mathbf{W}, \mathbf{x}, 1), \\ \mathbf{x} \in \mathbb{F}^{\ell},\ \ \mathbf{W} \in \mathbb{F}^{m-\ell-1} \end{array} \right\}, \tag{1}$$

[4]https://github.com/NethermindEth/sonobe/tree/paper

m	$\mathbf{W}$ entries	Mova	Nova	Hypernova
2^{16}	$\emptyset$	36.0216 ms	400.4406 ms	150.0280 ms
2^{16}	Small	64.5093 ms	428.9283 ms	178.5157 ms
2^{16}	Large	411.6056 ms	776.0246 ms	525.6120 ms
2^{20}	$\emptyset$	761.5190 ms	5470.6681 ms	3195.2789 ms
2^{20}	Small	1316.0204 ms	6025.1695 ms	3749.7803 ms
2^{20}	Large	5675.1007 ms	10384.2498 ms	8108.8606 ms

Table 2: Benchmarks of Mova, Nova, and Hypernova's Prover runtime on a single core, when the R1CS matrices are the identity matrix, using an unoptimised way of computing the cross term $\mathbf{T}$ (see Section 5.2 for the optimised method). As in Table 1, m denotes the size of the R1CS vectors. The column $\mathbf{W}$ indicates whether the entries in the R1CS witness are "small" (in the range $\{0, \ldots, |\mathbf{W}| - 1\}$) or "large" (sampled randomly in $|\mathbb{F}|$). The $\emptyset$ symbol indicates that the time to commit to $\mathbf{W}$ was not included. See Section 5.1 for further details. Hypernova's implementation does not use certain known optimizations and thus its runtime does not correspond to the concrete costs from Table 1, cf. Remark 5.2.

where $\ell \geq 0$ is fixed, $\circ$ is the Hadamard product, and $A \cdot \mathbf{Z}$ denotes matrix-vector multiplication (and similarly for $B \cdot \mathbf{Z}, C \cdot \mathbf{Z}$). For the purposes of this overview, we omit several formalities; for example we fix the public parameters $\mathbb{F}, A, B, C, n, m, \ell$ and omit referring to them as such. We also omit refering to the parameters and keys used in Com. We refer to the beginning of Section 3 for an explanation of the basic notation used in this section and throughout the paper.

In the Nova paper [KST21], Kothapalli, Setty and Tzialla introduce a new relation called relaxed committed R1CS relation. This is defined as

$$\mathbf{R}_{\mathsf{rcR1CS}} = \left\{ (\mathbb{x}; \mathbb{w}) = (\mathbf{x}, \overline{\mathbf{W}}, u, \bar{\mathbf{E}}; \mathbf{W}, \mathbf{E}) \;\middle|\; \begin{array}{l} (A \cdot \mathbf{Z}) \circ (B \cdot \mathbf{Z}) = u \cdot (C \cdot \mathbf{Z}) + \mathbf{E}, \\ \mathbf{Z} = (\mathbf{W}, \mathbf{x}, 1), \\ \mathsf{Com}(\mathbf{W}) = \overline{\mathbf{W}},\; \mathsf{Com}(\mathbf{E}) = \bar{\mathbf{E}}, \\ \mathbf{x} \in \mathbb{F}^{\ell}, \mathbf{W} \in \mathbb{F}^{m-\ell-1} \end{array} \right\}, \qquad (2)$$

The Nova authors go on to describe a reduction of knowledge [KP23] (i.e. a folding scheme) from $\mathbf{R}_{\mathsf{rcR1CS}} \times \mathbf{R}_{\mathsf{rcR1CS}}$ to $\mathbf{R}_{\mathsf{rcR1CS}}$. We reproduce the protocol in Protocol 1. Since it is required to understand why our scheme Mova is sound, in Section 2.2 we provide a quick overview of its knowledge soundness proof.

Suppose, for illustrative purposes, that in $\mathbf{R}_{\mathsf{rcR1CS}}$ we use two commitments schemes, Com_1 and Com_2, so that $\mathbf{W}$ is committed with Com_1 and $\mathbf{E}$ is committed with Com_2. Then suppose we modify Protocol 1 so that P uses Com_2 to commit to $\mathbf{T}$. It is clear that the resulting scheme works in the same way as the original one.

Our first observation is that, perhaps surprisingly, one does not need Com_2 to be binding for Protocol 1 to be knowledge sound. This is readily apparent from our overview in Section 2.2 of the knowledge soundness proof of Protocol 1. Indeed, the only property used from Com_2 is its additivity. With this observation in mind, it is not difficult to see that one can replace Com_2 in $\mathbf{R}_{\mathsf{rcR1CS}}$ and in Protocol 1 by any linear map $\mathcal{L}$, and still obtain a complete, publicly reducible, and knowledge sound folding scheme. Looking ahead, for us $\mathcal{L}$

Protocol 1 Nova [KST21] folding scheme $\mathbf{R}_{\mathsf{rcR1CS}} \times \mathbf{R}_{\mathsf{rcR1CS}} \to \mathbf{R}_{\mathsf{rcR1CS}}$

Input: Let $(\mathbb{x}_i; \mathbb{w}_i) = (\mathbf{x}_i, \overline{\mathbf{W}}_i, u_i, \bar{\mathbf{E}}_i; \mathbf{W}_i, \mathbf{E}_i) \in \mathbf{R}_{\mathsf{rcR1CS}}$ $(i = 1, 2)$.
P receives $(\mathbb{x}_i; \mathbb{w}_i)$ as input $(i = 1, 2)$.
V receives $\mathbb{x}_i$ as input $(i = 1, 2)$.

```
1: First, P computes $\mathbf{T} = (A \cdot \mathbf{Z}_1) \circ (B \cdot \mathbf{Z}_2) + (A \cdot \mathbf{Z}_2) \circ (B \cdot \mathbf{Z}_1) - u_1 \cdot (C \cdot \mathbf{Z}_2) - u_2 \cdot (C \cdot \mathbf{Z}_1)$,
   and sends V a commitment $\bar{\mathbf{T}}$ to $\mathbf{T}$.
2: V replies with a uniformly sampled challenge $\alpha \leftarrow \mathbb{F}$.
3: P and V both output $\mathbb{x} = (\mathbf{x}, \overline{\mathbf{W}}, u, \bar{\mathbf{E}})$ where $\mathbf{x} = \mathbf{x}_1 + \alpha \mathbf{x}_2$, $\overline{\mathbf{W}} = \overline{\mathbf{W}}_1 + \alpha \overline{\mathbf{W}}_2$,
   $u = u_1 + \alpha u_2$, and $\bar{\mathbf{E}} = \bar{\mathbf{E}}_1 + \alpha \bar{\mathbf{T}} + \alpha^2 \bar{\mathbf{E}}_2$.
4: Additionally, P outputs $\mathbb{w} = (\mathbf{W}, \mathbf{E})$, where $\mathbf{W} = \mathbf{W}_1 + \alpha \mathbf{W}_2$ and $\mathbf{E} = \mathbf{E}_1 + \alpha \mathbf{T} + \alpha^2 \mathbf{E}_2$.
```

will be the map that assigns, to a vector $\mathbf{E}$, the evaluation of its Multilinear Extension (MLE) at a fixed point $\mathbf{r}$ (with $\mathbf{r}$ being the same each time $\mathcal{L}$ is applied), i.e. $\mathcal{L}(\mathbf{E}) = \mathsf{mle}[\mathbf{E}](\mathbf{r})$.

More precisely, let $\mathcal{L} : \mathbb{F}^n \to \mathbb{G}$ be an additive homomorphism from $\mathbb{F}^n$ to some group $\mathbb{G}$. Define:

$$\mathbf{R}^{\mathcal{L}}_{\mathsf{rcR1CS}} = \left\{ (\mathbb{x}; \mathbb{w}) = (\mathbf{x}, \overline{\mathbf{W}}, u, \ell_{\mathbf{E}}; \mathbf{W}, \mathbf{E}) \;\middle|\; \begin{array}{l} (A \cdot \mathbf{Z}) \circ (B \cdot \mathbf{Z}) = u \cdot (C \cdot \mathbf{Z}) + \mathbf{E}, \\ \mathbf{Z} = (\mathbf{W}, \mathbf{x}, 1), \\ \mathsf{Com}(\mathbf{W}) = \overline{\mathbf{W}}, \mathcal{L}(\mathbf{E}) = \ell_{\mathbf{E}}, \\ \mathbf{x} \in \mathbb{F}^{\ell}, \mathbf{W} \in \mathbb{F}^{m-\ell-1} \end{array} \right\}.$$

Then the same proof as in Section 2.2 (i.e. the knowledge soundness proof of Nova) can be adapted to show that Protocol 2 is a reduction of knowledge from $\mathbf{R}^{\mathcal{L}}_{\mathsf{rcR1CS}} \times \mathbf{R}^{\mathcal{L}}_{\mathsf{rcR1CS}}$ to $\mathbf{R}_{\mathsf{rcR1CS}}$. We use the color blue to highlight the differences between $\mathbf{R}_{\mathsf{rcR1CS}}$ and $\mathbf{R}^{\mathcal{L}}_{\mathsf{rcR1CS}}$, and between Protocol 1 and Protocol 2.

Protocol 2 Mova folding scheme $\mathbf{R}^{\mathcal{L}}_{\mathsf{rcR1CS}} \times \mathbf{R}^{\mathcal{L}}_{\mathsf{rcR1CS}} \to \mathbf{R}^{\mathcal{L}}_{\mathsf{rcR1CS}}$

Input: Let $(\mathbb{x}_i; \mathbb{w}_i) = (\mathbf{x}_i, \overline{\mathbf{W}}_i, u_i, \mathcal{L}(\mathbf{E}_i); \mathbf{W}_i, \mathbf{E}_i) \in \mathbf{R}^{\mathcal{L}}_{\mathsf{rcR1CS}}$ $(i = 1, 2)$.
P receives $(\mathbb{x}_i; \mathbb{w}_i)$ as input $(i = 1, 2)$.
V receives $\mathbb{x}_i$ as input $(i = 1, 2)$.

```
1: First, P computes $\mathbf{T} = (A \cdot \mathbf{Z}_1) \circ (B \cdot \mathbf{Z}_2) + (A \cdot \mathbf{Z}_2) \circ (B \cdot \mathbf{Z}_1) - u_1 \cdot (C \cdot \mathbf{Z}_2) - u_2 \cdot (C \cdot \mathbf{Z}_1)$, and
   sends $\mathcal{L}(T)$ to V.
2: V replies with a uniformly sampled challenge $\alpha \leftarrow \mathbb{F}$.
3: P and V both output $\mathbb{x} = (\mathbf{x}, \overline{\mathbf{W}}, u, \mathcal{L}(\mathbf{E}))$ where $\mathbf{x} = \mathbf{x}_1 + \alpha \mathbf{x}_2$, $\overline{\mathbf{W}} = \overline{\mathbf{W}}_1 + \alpha \overline{\mathbf{W}}_2$, $u = u_1 + \alpha u_2$,
   and $\mathcal{L}(\mathbf{E}) = \mathcal{L}(\mathbf{E}_1) + \alpha \mathcal{L}(\mathbf{T}) + \alpha^2 \mathcal{L}(\mathbf{E}_2)$.
4: Additionally, P outputs $\mathbb{w} = (\mathbf{W}, \mathbf{E})$, where $\mathbf{W} = \mathbf{W}_1 + \alpha W_2$ and $\mathbf{E} = \mathbf{E}_1 + \alpha T + \alpha^2 \mathbf{E}_2$.
```

In fact, when looking only at folding schemes for $\mathbf{R}_{\mathsf{rcR1CS}} \times \mathbf{R}_{\mathsf{rcR1CS}} \to \mathbf{R}_{\mathsf{rcR1CS}}$, it is even possible to forget altogether about the commitments to $\mathbf{E}$ and the linear maps $\mathcal{L}$. Another way of seeing this is that, when Protocol 2 is instantiated with $\mathcal{L} : \mathbb{F}^m \to \{0\}$ being a degenerate map that maps all vectors to 0, then Protocol 2 is still a reduction of knowledge from $\mathbf{R}^{\mathcal{L}}_{\mathsf{rcR1CS}} \times \mathbf{R}^{\mathcal{L}}_{\mathsf{rcR1CS}}$ to $\mathbf{R}^{\mathcal{L}}_{\mathsf{rcR1CS}}$. We remark, however, that for such degenerate map, the relation $\mathbf{R}^{\mathcal{L}}_{\mathsf{rcR1CS}}$ is equally degenerate. Indeed, for any $\mathbf{x}, \overline{\mathbf{W}} = \mathsf{Com}(\mathbf{W}), u, \mathbf{W}$, there exists $\mathbf{E}$ such that $(\mathbb{x}; \mathbb{w}) = (\mathbf{x}, \overline{\mathbf{W}}, u, 0; \mathbf{W}, \mathbf{E}) \in \mathbf{R}^{\mathcal{L}}_{\mathsf{rcR1CS}}$. It suffices to take $\mathbf{E} = (A \cdot \mathbf{Z}) \circ (B \cdot \mathbf{Z}) - u(C \cdot \mathbf{Z})$, where $\mathbf{Z} = (\mathbf{W}, \mathbf{x}, 1)$.

The situation is more involved if one wishes to fold an instance-witness pair from $\mathbf{R}_{\mathsf{R1CS}}$ with an instance-witness pair from $\mathbf{R}_{\mathsf{rcR1CS}}$, i.e. if one wants to build a reduction of knowledge from $\mathbf{R}_{\mathsf{R1CS}} \times \mathbf{R}_{\mathsf{rcR1CS}}$ to $\mathbf{R}_{\mathsf{rcR1CS}}$. One can easily construct one as follows: first, design a reduction of knowledge Π_1 from $\mathbf{R}_{\mathsf{R1CS}}$ to $\mathbf{R}_{\mathsf{rcR1CS}}$:

$$\Pi_1 : \mathbf{R}_{\mathsf{R1CS}} \to \mathbf{R}_{\mathsf{rcR1CS}}.$$

Then, build a reduction of knowledge Π_2 from $\mathbf{R}_{\mathsf{rcR1CS}} \times \mathbf{R}_{\mathsf{rcR1CS}}$ to $\mathbf{R}_{\mathsf{rcR1CS}}$:

$$\Pi_2 : \mathbf{R}_{\mathsf{rcR1CS}} \times \mathbf{R}_{\mathsf{rcR1CS}} \to \mathbf{R}_{\mathsf{rcR1CS}}.$$

Once this is done, use the parallel composition theorem from [KP23] (cf. Theorem 3.4) to obtain a reduction of knowledge $\widehat{\Pi}_1$ from $\mathbf{R}_{\mathsf{R1CS}} \times \mathbf{R}_{\mathsf{rcR1CS}}$ to $\mathbf{R}_{\mathsf{rcR1CS}} \times \mathbf{R}_{\mathsf{rcR1CS}}$ (here we compose in parallel Π_1 and the trivial reduction of knowledge from $\mathbf{R}_{\mathsf{rcR1CS}}$ to $\mathbf{R}_{\mathsf{rcR1CS}}$ where the instance-witness pair from $\mathbf{R}_{\mathsf{rcR1CS}}$ remains unchanged). Then, use the sequential composition theorem from [KP23] (cf. Theorem 3.3) to obtain the desired reduction of knowledge (here we compose $\widehat{\Pi}_1$ with Π_2).

We next describe a simple way to obtain a reduction of knowledge Π_1 from $\mathbf{R}_{\mathsf{R1CS}}$ to $\mathbf{R}_{\mathsf{rcR1CS}}$. This construction is implicit in the Nova paper [KST21]. Let $(\mathbb{x}; \mathbb{w}) = (\mathbf{x}; \mathbf{W}) \in \mathbf{R}_{\mathsf{cR1CS}}$. Then P sends a commitment $\overline{\mathbf{W}}$ to $\mathbf{W}$, and P and V output $\mathbb{x}' = (\mathbf{x}, \overline{\mathbf{W}}, u, \overline{\mathbf{E}})$. P additionally outputs $\mathbb{w}' = (\mathbf{W}, \mathbf{E})$, where $u = 1$ and $\mathbf{E} = \mathbf{0}$. Here, we assume V has precomputed $\mathsf{Com}_2(\mathbf{0})$ (with a fixed commitment randomness that is reused every time this reduction of knowledge is applied).

Thus, even though we concluded above that the commitments to $\mathbf{E}$ are not necessary for the Nova folding scheme, one does in principle need to include them when reducing from $\mathbf{R}_{\mathsf{R1CS}}$ to $\mathbf{R}_{\mathsf{rcR1CS}}$, and afterwards carry them over to the reduction $\mathbf{R}_{\mathsf{rcR1CS}} \times \mathbf{R}_{\mathsf{rcR1CS}} \to \mathbf{R}_{\mathsf{rcR1CS}}$ (Protocol 1).

Our second main observation is the following: assume one has designed a reduction of knowledge from $\mathbf{R}_{\mathsf{R1CS}}$ to $\mathbf{R}^{\mathcal{L}}_{\mathsf{rcR1CS}}$, for some efficient linear function $\mathcal{L}$, in particular, for $\mathcal{L}$ not requiring committing to the error vector. Then, following the same strategy as above, we obtain a reduction of knowledge from $\mathbf{R}_{\mathsf{R1CS}} \times \mathbf{R}^{\mathcal{L}}_{\mathsf{rcR1CS}}$ to $\mathbf{R}^{\mathcal{L}}_{\mathsf{rcR1CS}}$. In particular, the resulting scheme does not require committing to the error vector $\mathbf{E}$, instead, it requires computing $\mathcal{L}(\mathbf{E})$. Again, looking ahead, and informally speaking, we will take $\mathcal{L}$ to be the evaluation of $\mathsf{mle}[\mathbf{E}]$ at a random point sampled by the Verifier.

As we will see next, our final Mova folding scheme almost follows the blueprint above. The difference is that, instead of using a reduction from $\mathbf{R}_{\mathsf{R1CS}}$ to $\mathbf{R}^{\mathcal{L}}_{\mathsf{rcR1CS}}$ for a suitable map $\mathcal{L}$, we describe a reduction from $\mathbf{R}_{\mathsf{R1CS}}$ to a union of relations of the form $\mathbf{R}^{\mathcal{L}}_{\mathsf{rcR1CS}}$, for different maps $\mathcal{L}$.

More precisely, define

$$\mathbf{R}_{\mathsf{acc}} = \left\{ (\mathbb{x}; \mathbb{w}) = (\mathbf{x}, \overline{\mathbf{W}}, u, \ell_{\mathbf{E}}, \mathbf{r}; \mathbf{W}, \mathbf{E}) \;\middle|\; \begin{array}{l} (A \cdot \mathbf{Z}) \circ (B \cdot \mathbf{Z}) = u \cdot (C \cdot \mathbf{Z}) + \mathbf{E}, \\ \mathbf{Z} = (\mathbf{W}, \mathbf{x}, 1), \\ \mathsf{Com}(\mathbf{W}) = \overline{\mathbf{W}},\ \mathsf{mle}[\mathbf{E}](\mathbf{r}) = \ell_{\mathbf{E}}, \\ \mathbf{x} \in \mathbb{F}^{\ell}, \mathbf{W} \in \mathbb{F}^{m-\ell-1}, \mathbf{r} \in \mathbb{F}^{\log n} \end{array} \right\}. \quad (3)$$

Notice that, for a fixed $\mathbf{r} \in \mathbb{F}^{\log n}$, this relation becomes $\mathbf{R}_{\mathsf{rcR1CS}}^{\mathcal{L}_\mathbf{r}}$, where $\mathcal{L}_r$ maps vectors $\mathbf{E}$ from $\mathbb{F}^n$ to the value $\mathsf{mle}[\mathbf{E}](\mathbf{r})$, i.e. $\mathcal{L}_\mathbf{r}(\mathbf{E}) = \mathsf{mle}[\mathbf{E}](\mathbf{r})$. Indeed,

$$\mathbf{R}_{\mathsf{acc}} = \bigcup_{\mathbf{r} \in \mathbb{F}^{\log n}} \mathbf{R}_{\mathsf{rcR1CS}}^{\mathcal{L}_\mathbf{r}}.$$

In Protocol 3 we describe our reduction of knowledge from $\mathbf{R}_{\mathsf{R1CS}}$ to $\mathbf{R}_{\mathsf{acc}}$. The main idea is that the constraint $\mathsf{Com}_2(\mathbf{E}) = \mathsf{Com}_2(\mathbf{0})$ in Nova's reduction of knowledge, is replaced by the constraint $\mathsf{mle}[\mathbf{E}](\mathbf{r}) = 0$ for a point $\mathbf{r} \in \mathbb{F}^{\log n}$ randomly sampled by V. Loosely speaking, as we see later in the paper, by the Schwartz-Zippel lemma, this forces $\mathbf{E}$ to be the zero vector with high probability, and allows the extractor to obtain a satisfying witness for the initial instance from $\mathbf{R}_{\mathsf{R1CS}}$. We remark that the knowledge soundness proof of this reduction of knowledge is not trivial (cf. Lemma 4.1 and Remark 4.3). One of the main conceptual points in the proof is that the vector $\mathbf{E}$ is implicitly committed by the commitment to the witness $\mathbf{W}$, because $\mathbf{E} = (A \cdot \mathbf{Z}) \circ (B \cdot \mathbf{Z}) - u(C \cdot \mathbf{Z})$. This comes up in a few other spots throughout our proofs.

Protocol 3 Mova's reduction of knowledge $\mathbf{R}_{\mathsf{R1CS}} \to \mathbf{R}_{\mathsf{acc}}$

Input: Let $(\mathbb{x}; \mathbb{w}) = (\mathbf{x}; \mathbf{W}) \in \mathbf{R}_{\mathsf{R1CS}}$. P receives $(\mathbb{x}; \mathbb{w})$ as input. V receives $\mathbb{x}$.

```
1: P computes a commitment $\overline{\mathbf{W}} = \mathsf{Com}(\mathbf{W})$ to $\mathbf{W}$, and sends $\overline{\mathbf{W}}$ to V.
2: V samples $\mathbf{r} \leftarrow \mathbb{F}^{\log n}$.
3: P and V both output $\mathbb{x}' = (\mathbf{x}, \overline{\mathbf{W}}, u, \ell_\mathbf{E}, \mathbf{r})$ where $u = 1$ and $\ell_\mathbf{E} = 0$.
4: Additionally, P outputs $\mathbb{w}' = (\mathbf{W}, \mathbf{E})$, where $\mathbf{E} = \mathbf{0}$.
```

Are we done? Not yet, since we have not described a reduction from $\mathbf{R}_{\mathsf{cR1CS}}$ to $\mathbf{R}_{\mathsf{rcR1CS}}^{\mathcal{L}}$ for a suitable $\mathcal{L}$, and thus we cannot use our reduction from $\mathbf{R}_{\mathsf{rcR1CS}}^{\mathcal{L}} \times \mathbf{R}_{\mathsf{rcR1CS}}^{\mathcal{L}}$ to $\mathbf{R}_{\mathsf{rcR1CS}}^{\mathcal{L}}$ from Protocol 2. We need to build instead a reduction of knowledge from $\mathbf{R}_{\mathsf{acc}} \times \mathbf{R}_{\mathsf{acc}}$ to $\mathbf{R}_{\mathsf{acc}}$. We do this in two steps:

- First, we describe a reduction of knowledge from $\mathbf{R}_{\mathsf{acc}} \times \mathbf{R}_{\mathsf{acc}}$ to $\mathbf{R}_{\mathsf{equal}}$, where $\mathbf{R}_{\mathsf{equal}}$ consists of pairs $(\mathbb{x}_1, \mathbb{w}_1), (\mathbb{x}_2, \mathbb{w}_2)$ from $\mathbf{R}_{\mathsf{acc}}$ with the same evaluation point $\mathbf{r}$ both in $\mathbb{x}_1$ and $\mathbb{x}_2$. Precisely,

$$\mathbf{R}_{\mathsf{equal}} = \left\{ (\mathbb{x}_1, \mathbb{x}_2; \mathbb{w}_1, \mathbb{w}_2) \,\middle|\, \text{Exists } \mathbf{r} \in \mathbb{F}^{\log n} \text{ such that } (\mathbb{x}_i; \mathbb{w}_i) \in \mathbf{R}_{\mathsf{rcR1CS}}^{\mathcal{L}_\mathbf{r}},\ i = 1, 2 \right\}$$

- Observe that an instance-witness pair from $\mathbf{R}_{\mathsf{equal}}$ consists of two instance-witness pairs each from $\mathbf{R}_{\mathsf{rcR1CS}}^{\mathcal{L}_\mathbf{r}}$ for some $\mathbf{r} \in \mathbb{F}^{\log n}$. Hence, here we can use our reduction from $\mathbf{R}_{\mathsf{acc}}^{\mathcal{L}_\mathbf{r}} \times \mathbf{R}_{\mathsf{acc}}^{\mathcal{L}_\mathbf{r}}$ to $\mathbf{R}_{\mathsf{acc}}^{\mathcal{L}_\mathbf{r}}$ in Protocol 2.

Once this is done, it suffices to use the sequential and parallel composition theorems from [KP23] (cf. Theorems 3.3 and 3.4) so as to obtain our desired reduction from $\mathbf{R}_{\mathsf{R1CS}} \times \mathbf{R}_{\mathsf{acc}}$ to $\mathbf{R}_{\mathsf{acc}}$. Precisely, the proof workflow is as follows: First, use the parallel composition theorem to obtain a reduction from $\mathbf{R}_{\mathsf{R1CS}} \times \mathbf{R}_{\mathsf{acc}}$ to $\mathbf{R}_{\mathsf{R1CS}} \times \mathbf{R}_{\mathsf{acc}}$ (composing Protocol 3 and the trivial reduction from $\mathbf{R}_{\mathsf{acc}}$ to $\mathbf{R}_{\mathsf{acc}}$). Then, use the reduction from $\mathbf{R}_{\mathsf{acc}} \times \mathbf{R}_{\mathsf{acc}}$ to $\mathbf{R}_{\mathsf{equal}}$ from Protocol 4. Finally, Protocol 2 is naturally adapted to become a reduction of knowledge from $\mathbf{R}_{\mathsf{equal}}$ to $\mathbf{R}_{\mathsf{acc}}$.

In Protocol 4 we describe our reduction from $\mathbf{R}_{\mathsf{acc}} \times \mathbf{R}_{\mathsf{acc}}$ to $\mathbf{R}_{\mathsf{equal}}$. The scheme is based on what we call the "point-vs-line" argument (cf. Section 4.5.2 from Thaler's book [Tha22]).

The point-vs-line argument allows to take two evaluation claims for multilinear polynomials (at different points) and turn them into evaluation two evaluation claims at the same point. The argument works as follows. Say we have two claims of the form $f_1(\mathbf{r}_1) = c_1$ and $f_2(\mathbf{r}_2) = c_2$ for some multilinear polynomials f_1, f_2 in n variables, some $\mathbf{r}_1, \mathbf{r}_2 \in \mathbb{F}^n$, and some $c_1, c_2 \in \mathbb{F}$. Let $\ell : \mathbb{F} \to \mathbb{F}^n$ be the parameterised line such that $\ell(0) = \mathbf{r}_1$ and $\ell(1) = \mathbf{r}_2$, and set $h_i := f_i \circ \ell$ for $i = 1, 2$. Then, we have that the h_i are univariate polynomials of degree at most n, and $h_1(0) = c_1$, $h_2(1) = c_2$. Once this has been checked, we ask for the Verifier to pick a random uniform $\beta \in \mathbb{F}$, and the new claims are $f_i(\mathbf{r}') = c'_i$, where $\mathbf{r}' := \ell(\beta)$ and $c'_i := h_i(\beta)$ for $i = 1, 2$. Importantly, the new claims are about the evaluations of the f_i at the same point $\mathbf{r}'$. We will show that this is a knowledge sound reduction of knowledge from the initial evaluation claims to the new evaluation claims at the same point (see Section 7.1).

Protocol 4 Mova's reduction of knowledge $\mathbf{R}_{\mathsf{acc}} \times \mathbf{R}_{\mathsf{acc}} \to \mathbf{R}_{\mathsf{equal}}$

Input: P receives $(\mathbb{x}_i; \mathbb{w}_i) = (\mathbf{x}_i, \overline{\mathbf{W}}_i, u_i, \ell_{\mathbf{E}_i}, \mathbf{r}_i; \mathbf{W}_i, \mathbf{E}_i) \in \mathbf{R}_{\mathsf{acc}}$ as input, for $i = 1, 2$. V receives $\mathbb{x}_1, \mathbb{x}_2$ as input.

```
1: If $\mathbf{r}_1 = \mathbf{r}_2$, then $\mathsf{P}$ outputs $(\mathsf{pp}, \mathbb{x}_1, \mathbb{x}_2; \mathbb{w}_1, \mathbb{w}_2)$, and $\mathsf{V}$ outputs $(\mathsf{pp}, \mathbb{x}_1, \mathbb{x}_2)$.
   Otherwise, let $\ell : \mathbb{F} \to \mathbb{F}^{\log(m)}$ be the linear function satisfying $\ell(0) = \mathbf{r}_1, \ell(1) = \mathbf{r}_2$. $\mathsf{P}$ sends $\mathsf{V}$
   polynomials $h_1, h_2$ of degree at most $\log(m)$. Supposedly,
       $h_i(X) := \mathsf{mle}[\mathbf{E}_i] \circ \ell(\mathbf{X}), \quad i = 1, 2.$
2: $\mathsf{V}$ checks that $h_1(0) = v_1$ and $h_2(1) = v_2$, and aborts if this check fails. $\mathsf{V}$ samples a random
   $\beta \leftarrow \mathbb{F}$.
3: $\mathsf{P}$ and $\mathsf{V}$ set $\ell'_{\mathbf{E}_i} := h_i(\beta)$ and $\mathbf{r}' := \ell(\beta)$.
4: $\mathsf{P}$ and $\mathsf{V}$ output $\mathbb{x}'_i = (\mathbf{x}_i, \overline{\mathbf{W}}_i, u_i, \ell'_{\mathbf{E}_i}, \mathbf{r}')$ for $i = 1, 2$.
5: Additionally, $\mathsf{P}$ outputs $\mathbb{w}'_i = (\mathbf{W}_i, \mathbf{E}_i)$, for $i = 1, 2$.
```

2.2 Knowledge soundness proof of Protocol 2

For completeness, in this section we provide a brief overview of the knowledge soundness proof of our reduction of knowledge from $\mathbf{R}^{\mathcal{L}}_{\mathsf{rcR1CS}} \times \mathbf{R}^{\mathcal{L}}_{\mathsf{rcR1CS}}$ to $\mathbf{R}^{\mathcal{L}}_{\mathsf{rcR1CS}}$ cf. (Protocol 2). The knowledge soundness proof of Nova [KST21] (i.e. Protocol 1) can be recovered by replacing $\mathcal{L}$ with any homomorphic commitment Com_2 (typically, $\mathsf{Com}_2 = \mathsf{Com}$).

The proof uses the forking lemma for reductions of knowledge [KP23, KST21, BCS21]. This lemma states that, to prove that a reduction of knowledge is knowledge sound, it suffices to describe a PPT extractor that outputs valid witnesses, given a tree of accepting transcripts.

When applied to Protocol 1, the forking lemma can be applied in the following way. Let $\mathbb{x}_1, \mathbb{x}_2$ be a pair of instances for the relation $\mathbf{R}^{\mathcal{L}}_{\mathsf{rcR1CS}}$, let

$$\mathsf{tr}^{(i)} = (\ell^{(i)}_{\mathbf{T}}, \alpha^{(i)}), \quad i = 1, 2, 3$$

be three accepting transcripts for Protocol 1, and let $(\mathbb{x}^{(i)}, \mathbb{w}^{(i)})$ be instance-witness pairs in $\mathbf{R}^{\mathcal{L}}_{\mathsf{rcR1CS}}$ output after each of the transcripts $\mathsf{tr}^{(i)}$, $i = 1, 2, 3$. Assume the challenges $\alpha^{(i)}$ are pairwise different, and that $\ell^{(1)}_{\mathbf{T}} = \ell^{(2)}_{\mathbf{T}} = \ell^{(3)}_{\mathbf{T}}$, thus we can omit the superscript and write $\ell_{\mathbf{T}}$. Now the forking lemma states that knowledge soundness of Protocol 1 is guaranteed by the existence of a PPT extractor Ext that outputs valid witnesses $\mathbb{w}_1, \mathbb{w}_2$, when given $\mathbb{x}_1, \mathbb{x}_2, \tau^{(i)}$ $(i = 1, 2, 3)$ as input.

Let $(\mathbb{x}^{(i)}; \mathbb{w}^{(i)}) = (\mathbf{x}^{(i)}, \overline{\mathbf{W}}^{(i)}, u^{(i)}, \bar{\mathbf{E}}^{(i)}; \mathbf{W}^{(i)}, \mathbf{E}^{(i)})$. By correctness of transcripts, it holds that

$$\overline{\mathbf{W}}^{(i)} = \overline{\mathbf{W}}_1 + \alpha^{(i)} \overline{\mathbf{W}}_2, \quad i = 1, 2, 3. \tag{4}$$

Using interpolation for $i = 1, 2$, Ext constructs vectors $\mathbf{W}_1^*, \mathbf{W}_2^*$ such that $\mathbf{W}^{(i)} = \mathbf{W}_1^* + \alpha^{(i)} \mathbf{W}_2^*$ for $i = 1, 2$. Using the linear properties of Com together with Eq. (4), one deduces that

$$\mathsf{Com}(\mathbf{W}_1^*) + \alpha^{(i)} \mathsf{Com}(\mathbf{W}_2^*) = \mathsf{Com}(\mathbf{W}^{(i)}) = \overline{\mathbf{W}}^{(i)} = \overline{\mathbf{W}}_1 + \alpha^{(i)} \overline{\mathbf{W}}_2, \quad i = 1, 2.$$

Then, again by interpolation, $\mathsf{Com}(\mathbf{W}_1^*) = \overline{\mathbf{W}}_1$ and $\mathsf{Com}(\mathbf{W}_2^*) = \overline{\mathbf{W}}_2$. Now

$$\mathsf{Com}(\mathbf{W}^{(3)}) = \overline{\mathbf{W}}^{(3)} = \overline{\mathbf{W}}_1 + \alpha^{(3)} \overline{\mathbf{W}}_2 = \mathsf{Com}(\mathbf{W}_1^* + \alpha^{(3)} \mathbf{W}_2^*),$$

and hence, by the binding property of Com, we have $\mathbf{W}^{(3)} = \mathbf{W}_1^* + \alpha^{(3)} \mathbf{W}_2^*$, e.w.n.p.

Next, by correctness of transcripts,

$$\ell_{\mathbf{E}^{(i)}} = \ell_{\mathbf{E}_1} + \alpha^{(i)} \ell_{\mathbf{T}} + (\alpha^{(i)})^2 \ell_{\mathbf{E}_2}, \quad i = 1, 2, 3. \tag{5}$$

Using interpolation, Ext constructs vectors $\mathbf{E}_1^*, \mathbf{E}_2^*, \mathbf{T}^*$ such that $\mathbf{E}^{(i)} = \mathbf{E}_1^* + \alpha^{(i)} \mathbf{T}^* + (\alpha^{(i)})^2 \mathbf{E}_2^*$ for $i = 1, 2, 3$. Now, similarly, as above, using the linearity of $\mathcal{L}$, together with Eq. (5), one obtains

$$\mathcal{L}(\mathbf{E}_1^*) + \alpha^{(i)} \mathcal{L}(\mathbf{T}^*) + (\alpha^{(i)})^2 \mathcal{L}(\mathbf{E}_2^*) = \ell_{\mathbf{E}_1} + \alpha^{(i)} \ell_{\mathbf{T}} + (\alpha^{(i)})^2 \ell_{\mathbf{E}_2}.$$

Again by interpolation, $\mathcal{L}(\mathbf{E}_1^*) = \ell_{\mathbf{E}_1}, \mathcal{L}(\mathbf{E}_2^*) = \ell_{\mathbf{E}_2}, \mathcal{L}(\mathbf{T}^*) = \ell_{\mathbf{T}}$.

Now Ext outputs $\mathbb{w}_1 = (\mathbf{W}_1^*, \mathbf{E}_1^*)$ and $\mathbb{w}_2 = (\mathbf{W}_2^*, \mathbf{E}_2^*)$. We are left to show that $(A \cdot \mathbf{Z}_j) \circ (B \cdot \mathbf{Z}_j) = u_j \cdot (C \cdot \mathbf{Z}_j) + \mathbf{E}_j$ for $\mathbf{Z}_j = (\mathbf{W}_j, \mathbf{x}_j, 1)$, $j = 1, 2$. This follows in the same exact way as in the Nova paper [KST21], and does not require using any properties of Com or $\mathcal{L}$.

3 Preliminaries

Throughout the article we fix a finite field $\mathbb{F}$. Given an integer $k \geq 1$ we denote $[k] := \{1, \ldots, n\}$. We let $\mathbb{B} := \{0, 1\}$. Similarly

$$\mathbb{B}^k = \{0, 1\}^k := \{(b_1, \ldots, b_k) \mid b_i \in \mathbb{B}, \text{ for all } i \in [k]\}$$

is the hypercube of dimension k, or, in other words, the set of all sequences of k bits.

For $n \geq 1$ and $d \geq 0$, we let $\mathbb{F}^{\leq d}[\mathbf{X}]$ be the set of multivariate polynomials in the variables $\mathbf{X} = (X_1, \ldots, X_n)$ with degree in each variable at most d.

Given an $m_1 \times m_2$ matrix A and a vector $\mathbf{Z}$ of size m_2, we write $A \cdot \mathbf{Z}$ to denote the multiplication of the matrix A with $\mathbf{Z}$ in column form.

Given two interactive algorithms $\mathcal{A}_1, \mathcal{A}_2$, we let

$$\langle \mathcal{A}_1(\mathsf{inp}_1), \mathcal{A}_2(\mathsf{inp}_2) \rangle(\mathsf{inp}_3)$$

be the random variable whose outcome is the output of the interaction of $\mathcal{A}_1$ and $\mathcal{A}_2$ on inputs $(\mathsf{inp}_1, \mathsf{inp}_3)$ and $(\mathsf{inp}_2, \mathsf{inp}_3)$, respectively.

Negligible functions We use λ to denote the security parameter. A function $f(\lambda)$ is said to be *negligible* if for all $c \in \mathbb{N}$ and $k \in \mathbb{R}$ with $k > 0$, there exists λ_0 such that $f(\lambda) < k\lambda^{-c}$ for all $\lambda \geq \lambda_0$. In that case we write $f(\lambda) = \mathsf{negl}(\lambda)$. Whenever an event occurs with probability $1 - \mathsf{negl}(\lambda)$ we will say that it holds *except with negligible probability*, and we abbreviate this as *e.w.n.p.*

Remark 3.1. Let $f(\lambda), g(\lambda)$ be two negligible functions and $g(\lambda) \neq 1$ for all λ. Define $h(\lambda) = f(\lambda)/(1 - g(\lambda))$. Then $h(\lambda)$ is also negligible. Indeed, let λ_0 be such that $g(\lambda) < \frac{1}{2} \cdot 1$ for all $\lambda \geq \lambda_0$. Thus for such λ we have $1 - g(\lambda) > \frac{1}{2}$. Thus

$$h(\lambda) = \frac{f(\lambda)}{1 - g(\lambda)} < 2 \cdot f(\lambda), \text{ for all } \lambda \geq \lambda_0.$$

It is clear that the function $2 \cdot f(\lambda)$ is negligible and a function dominated by a negligible function is again negligible.

Indexed relations An *indexed relation* is a subset $\mathbf{R} \subseteq \{0,1\}^* \times \{0,1\}^* \times \{0,1\}^*$. Given $(\mathsf{pp}, \mathbb{x}; \mathbb{w}) \in \mathbf{R}$, the string pp are the *public parameters* (sometimes referred to as *index* in the literature); $\mathbb{x}$ is called an *instance*, and $\mathbb{w}$ a *witness*. In this work we often interpret instances and witnesses as vectors of field elements, natural numbers, and field descriptions.

All relations considered in this work are indexed relations. We always use the syntax $(\mathsf{pp}, \mathbb{x}; \mathbb{w})$ to denote a triple in $\mathbf{R}$. The usage of ; denotes a separation between public and private data.

3.1 Multilinear polynomials

Let $n \geq 1$ and let $\mathbf{X} = (X_1, \ldots, X_n)$ be a tuple of variables. It is well-known that a multilinear polynomial $f(\mathbf{X}) \in \mathbb{F}^{\leq 1}[\mathbf{X}]$ is uniquely determined by the values it takes on $\mathbb{B}^n$, i.e. its restriction to $\mathbb{B}^n$. In other words, for any two polynomials $f, g \in \mathbb{F}^{\leq 1}[\mathbf{X}]$ the following holds

$$f(\mathbf{x}) = g(\mathbf{x}) \text{ for all } \mathbf{x} \in \mathbb{B}^n \implies f = g.$$

Further, given a map $f : \mathbb{B}^n \to \mathbb{F}$, there always exists a unique multilinear polynomial in n variables, denoted $\mathsf{mle}[f](\mathbf{X})$, such that $\mathsf{mle}[f](\mathbf{x}) = f(\mathbf{x})$ for all $\mathbf{x} \in \mathbb{B}^n$. It is given by the expression

$$\mathsf{mle}[f](\mathbf{X}) := \sum_{\mathbf{x} \in \mathbb{B}^n} f(\mathbf{x}) \widetilde{\mathsf{eq}}(\mathbf{x}; \mathbf{X}) \tag{6}$$

where $\widetilde{\mathsf{eq}}(\mathbf{x}; \mathbf{X})$ is the unique multilinear polynomial in n variables that takes the value 0 on all points of the hypercube $\mathbb{B}^n$, except at $\mathbf{x}$ where it takes the value 1. Precisely,

$$\widetilde{\mathsf{eq}}(\mathbf{x}; \mathbf{X}) := \prod_{i \in [n]} \left(x_i X_i - (1 - x_i)(1 - X_i) \right).$$

This unique multilinear polynomial $\mathsf{mle}[f](\mathbf{X})$ is called the *multilinear extension (MLE)* of f. Given a vector $\mathbf{v} = (v_1, \ldots, v_N) \in \mathbb{F}^N$, where $N = 2^n$, we define the MLE of $\mathbf{v}$ as the MLE of the map $\mathbf{v} : \mathbb{B}^n \to \mathbb{F}$ assigning to each element $\mathbf{x} \in \mathbb{B}^n$ the element $v_{\mathbf{x}}$, where here we interpret $\mathbf{x}$ as the natural number whose binary representation is $\mathbf{x}$. We denote the MLE of $\mathbf{v}$ by $\mathsf{mle}[\mathbf{v}](\mathbf{X})$.

Throughout the paper we use the following observation without further reference:

Lemma 3.2. *Let $\mathbf{v}, \mathbf{u} \in \mathbb{F}^N$ be two vectors, and let $\mathbf{r} \in \mathbb{F}^n$. Then*

$$\mathsf{mle}[\mathbf{v} + \mathbf{u}](\mathbf{r}) = \mathsf{mle}[\mathbf{v}](\mathbf{r}) + \mathsf{mle}[\mathbf{u}](\mathbf{r})$$

Proof. This follows immediately from Eq. (6). □

3.2 Commitment schemes

Throughout this section we follow [KST21].

Definition 3.1. *A commitment scheme for vectors in $\mathbb{F}^k$ is a tuple of three protocols*

- $\mathsf{Gen}(1^\lambda, k) \to \mathsf{pp}_{\mathsf{Com}}$*: Takes a security parameter 1^λ and a length parameter k. Outputs public parameters $\mathsf{pp}_{\mathsf{Com}}$.*
- $\mathsf{Com}(\mathsf{pp}_{\mathsf{Com}}, \mathbf{W}, s) \to C$*: Takes as input $\mathsf{pp}_{\mathsf{Com}}$, a vector $\mathbf{W} \in \mathbb{F}^k$ and $s \in \mathbb{F}$. Outputs a commitment C.*
- $\mathsf{Open}(\mathsf{pp}_{\mathsf{Com}}, C, \mathbf{W}, s) \to \{0, 1\}$*: Checks whether $C = \mathsf{Com}(\mathsf{pp}_{\mathsf{Com}}, \mathbf{W}, s)$.*

The scheme is required to be binding*. This means that for any PPT adversary $\mathcal{A}$, the following probability is $\mathsf{negl}(\lambda)$:*

$$\mathsf{Pr}\left[\begin{array}{l|l} & \mathsf{pp}_{\mathsf{Com}} \leftarrow \mathsf{Gen}(1^\lambda, k), \\ b_0 = b_1 = 1, & (C, \mathbf{W}_0 \in \mathbb{F}^k, \mathbf{W}_1 \in \mathbb{F}^k, s_0 \in \mathbb{F}, s_1 \in \mathbb{F}) \leftarrow \mathcal{A}(\mathsf{pp}_{\mathsf{Com}}), \\ \mathbf{W}_0 \neq \mathbf{W}_1 & b_0 \leftarrow \mathsf{Open}(\mathsf{pp}_{\mathsf{Com}}, C, \mathbf{W}_0, s_0), \\ & b_1 \leftarrow \mathsf{Open}(\mathsf{pp}_{\mathsf{Com}}, C, \mathbf{W}_1, s_1) \end{array}\right]$$

Throughout the paper we fix a commitment scheme that is succinct and additively homomorphic, as defined below.

Definition 3.2 (Succinctness)**.** *A commitment scheme for vectors in $\mathbb{F}^k$, $(\mathsf{Gen}, \mathsf{Com}, \mathsf{Open})$, is* succinct *if for all $\mathsf{pp}_{\mathsf{Com}} \leftarrow \mathsf{Gen}(1^\lambda, k)$, and for any $\mathbf{W} \in \mathbb{F}^k$, $s \in \mathbb{F}$, $|\mathsf{Com}(\mathsf{pp}_{\mathsf{Com}}, \mathbf{W}, s)| = O_\lambda(\mathsf{polylog}(|\mathbf{W}|))$.*

Definition 3.3 (Additively Homomorphic)**.** *A commitment scheme for vectors in $\mathbb{F}^k$, $(\mathsf{Gen}, \mathsf{Com}, \mathsf{Open})$, is* additively homomorphic *if for all $\mathsf{pp}_{\mathsf{Com}} \leftarrow \mathsf{Gen}(1^\lambda, k)$, and for any $\mathbf{W}_1, \mathbf{W}_2 \in \mathbb{F}^k$, $s_1, s_2 \in \mathbb{F}$, it holds that $\mathsf{Com}(\mathsf{pp}_{\mathsf{Com}}, \mathbf{W}_1, s_1) + \mathsf{Com}(\mathsf{pp}_{\mathsf{Com}}, \mathbf{W}_2, s_2) = \mathsf{Com}(\mathsf{pp}_{\mathsf{Com}}, \mathbf{W}_1 + \mathbf{W}_2, s_1 + s_2)$.*

3.3 Reductions of knowledge

Reductions of knowledge were formally defined in [KP23]. We reproduce the main concepts from [KP23] that will be used in this paper.

Definition 3.4 (Reduction of Knowledge)**.** *Consider indexed relations* $\mathbf{R}_1$ *and* $\mathbf{R}_2$ *consisting of public parameters, input, witness tuples. A reduction of knowledge from* $\mathbf{R}_1$ *to* $\mathbf{R}_2$ *is defined by PPT algorithms* $(\mathsf{Gen}, \mathsf{P}, \mathsf{V})$ *denoting the generator, the Prover, and the Verifier, respectively, with the following interface.*

- $\mathsf{Gen}(1^\lambda) \to \mathsf{pp}$*: Takes security parameter* λ*. Outputs public parameters* pp*.*
- $\mathsf{P}(\mathsf{pp}, \mathbb{x}_1, \mathbb{w}_1) \to (\mathbb{x}_2; \mathbb{w}_2)$*: Takes as input public parameters* pp*, and input-witness pair* $(\mathbb{x}_1; \mathbb{w}_1)$*. Interactively reduces the input* $(\mathsf{pp}, \mathbb{x}_1; \mathbb{w}_1) \in \mathbf{R}_1$ *to a new input* $(\mathsf{pp}, \mathbb{x}_2; \mathbb{w}_2) \in \mathbf{R}_2$*.*
- $V(\mathsf{pp}, \mathbb{x}_1) \to \mathbb{x}_2$*: Takes as input public parameters* pp*, and input* $\mathbb{x}_1$ *associated with* $\mathbf{R}_1$*. Interactively reduces the task of checking* $\mathbb{x}_1$ *to the task of checking a new input* $\mathbb{x}_2$ *associated with* $\mathbf{R}_2$*.*

Let $\langle \mathsf{P}, \mathsf{V} \rangle$ *denote the interaction between* P *and* V*. We treat* $\langle \mathsf{P}, \mathsf{V} \rangle$ *as a function that takes as input* $(\mathsf{pp}, \mathbb{x}_1, \mathbb{w}_1)$ *and runs the interaction on Prover's input* $(\mathsf{pp}, \mathbb{x}_1, \mathbb{w}_1)$ *and Verifier's input* $(\mathsf{pp}, \mathbb{x}_1)$*. At the end of the interaction,* $\langle \mathsf{P}, \mathsf{V} \rangle$ *outputs the Verifier's input* $\mathbb{x}_2$ *and the Prover's witness* $\mathbb{w}_2$*. A reduction of knowledge* $(\mathsf{Gen}, \mathsf{P}, \mathsf{V})$ *satisfies the following conditions.*

1. ***Perfect Completeness:*** *For any PPT adversary* $\mathcal{A}$*, given* $\mathsf{pp} \leftarrow \mathsf{Gen}(1^\lambda)$ *and* $(\mathbb{x}_1; \mathbb{w}_1) \leftarrow \mathcal{A}(\mathsf{pp})$ *such that* $(\mathsf{pp}, \mathbb{x}_1; \mathbb{w}_1) \in \mathbf{R}_1$*, we have that the Verifier accepts at the end of the protocol, the Prover's output is equal to the Verifier's output, and*
$$(\mathsf{pp}, \langle \mathsf{P}, \mathsf{V} \rangle(\mathsf{pp}, \mathbb{x}_1, \mathbb{w}_1)) \in \mathbf{R}_2.$$
2. ***Knowledge Soundness:*** *For any expected polynomial-time adversaries* $\mathcal{A}$ *and* P^**, there exists an expected polynomial-time extractor* Ext *such that given* $\mathsf{pp} \leftarrow \mathsf{Gen}(1^\lambda)$ *and* $(\mathbb{x}_1, \mathsf{st}) \leftarrow \mathcal{A}(\mathsf{pp})$*, we have that*
$$\Pr[(\mathsf{pp}, \mathbb{x}_1, \mathsf{Ext}(\mathsf{pp}, \mathbb{x}_1, \mathsf{st})) \in \mathbf{R}_1] \approx \Pr[(\mathsf{pp}, \langle \mathsf{P}^*, \mathsf{V} \rangle(\mathsf{pp}, \mathbb{x}_1, \mathsf{st})) \in \mathbf{R}_2].$$
3. ***Public Reducibility:*** *There exists a deterministic polynomial-time function* φ *such that for any PPT adversary* $\mathcal{A}$ *and expected polynomial-time adversary* P^**, given* $\mathsf{pp} \leftarrow \mathsf{Gen}(1^\lambda)$*,* $(\mathbb{x}_1, \mathsf{st}) \leftarrow \mathcal{A}(\mathsf{pp})$*, and* $(\mathbb{x}_2; \mathbb{w}_2) \leftarrow \langle \mathsf{P}^*, \mathsf{V} \rangle(\mathsf{pp}, \mathbb{x}_1, \mathsf{st})$ *with interaction transcript* τ*, we have that* $\varphi(\mathsf{pp}, \mathbb{x}_1, \tau) = \mathbb{x}_2$*.*

We write $\Pi : \mathbf{R}_1 \to \mathbf{R}_2$ *to denote that protocol* Π *is a reduction of knowledge from relation* $\mathbf{R}_1$ *to relation* $\mathbf{R}_2$*.*

Theorem 3.3 (Sequential Composition, Theorem 5 of [KP23])**.** *Consider ternary relations* $\mathbf{R}_1$*,* $\mathbf{R}_2$*, and* $\mathbf{R}_3$*. For reductions of knowledge* $\Pi_1 = (\mathsf{Gen}, \mathsf{P}_1, V_1) : \mathbf{R}_1 \to \mathbf{R}_2$ *and* $\Pi_2 = (\mathsf{Gen}, \mathsf{P}_2, \mathsf{V}_2) : \mathbf{R}_2 \to \mathbf{R}_3$*, we have that* $\Pi_2 \circ \Pi_1 = (\mathsf{Gen}, \mathsf{P}, \mathsf{V})$ *is a reduction of knowledge from* $\mathbf{R}_1$ *to* $\mathbf{R}_3$ *where*
$$\begin{aligned}\mathsf{P}(\mathsf{pp}, \mathbb{x}_1, \mathbb{w}_1) &= \mathsf{P}_2(\mathsf{pp}, \mathsf{P}_1(\mathsf{pp}, \mathbb{x}_1, \mathbb{w}_1))\\ \mathsf{V}(\mathsf{pp}, \mathbb{x}_1) &= \mathsf{V}_2(\mathsf{pp}, \mathsf{V}_1(\mathsf{pp}, \mathbb{x}_1, \mathbb{w}_1)).\end{aligned}$$

Theorem 3.4 (Parallel Composition, Theorem 6 of [KP23])**.** *Consider ternary relations* $\mathbf{R}_1$*,* $\mathbf{R}_2$*,* $\mathbf{R}_3$*, and* $\mathbf{R}_4$*. For reductions of knowledge* $\Pi_1 = (\mathsf{Gen}, \mathsf{P}_1, \mathsf{V}_1) : \mathbf{R}_1 \to \mathbf{R}_2$ *and* $\Pi_2 = (\mathsf{Gen}, \mathsf{P}_2, \mathsf{V}_2) : \mathbf{R}_3 \to \mathbf{R}_4$*, we have that* $\Pi_1 \times \Pi_2 = (\mathsf{Gen}, \mathsf{P}, \mathsf{V})$ *is a reduction of knowledge from* $\mathbf{R}_1 \times \mathbf{R}_3$ *to* $\mathbf{R}_2 \times \mathbf{R}_4$ *where*

$$\mathsf{P}(\mathsf{pp}, (\mathbb{x}_1, \mathbb{x}_3), (\mathbb{w}_1, \mathbb{w}_3)) = (\mathsf{P}_1(\mathsf{pp}, \mathbb{x}_1, \mathbb{w}_1), \mathsf{P}_2(\mathsf{pp}, \mathbb{x}_3, \mathbb{w}_3))$$
$$\mathsf{V}(\mathsf{pp}, (\mathbb{x}_1, \mathbb{x}_3)) = (\mathsf{V}_1(\mathsf{pp}, \mathbb{x}_1), \mathsf{V}_2(\mathsf{pp}, \mathbb{x}_3)).$$

Definition 3.5 (Tree of transcripts)**.** *Consider an* m*-round public coin interactive protocol* $(\mathsf{Gen}, \mathsf{P}, \mathsf{V})$ *satisfying Definition 3.4. A* $(n_1, \ldots, n_m)$*-tree of accepting transcripts, for an input* $\mathbb{x}$*, is a rooted tree of depth* m*, with each node of depth* i *having* n_i *descendants, such that:*

- *each vertex of layer* i *is a Prover's message of round* i*;*
- *each edge connecting a node of layer* i *to a node of layer* $i + 1$ *is labeled by a different Verifier's challenge from round* i*;*
- *each leaf of layer* m *is labeled with an accepting instance-witness pair output, that corresponds to the interaction along the path.*

Lemma 3.5 (Tree Extraction Lemma)**.** *Consider an* m*-round public-coin interactive protocol* $(\mathsf{Gen}, \mathsf{P}, \mathsf{V})$ *that satisfies the interface described in Definition 3.4 and satisfies perfect completeness. Then* $(\mathsf{Gen}, \mathsf{P}, \mathsf{V})$ *is a reduction of knowledge if there exists a PPT extractor* χ *such that for all instances* $\mathbb{x}$*, outputs a satisfying witness* $\mathbb{w}$ *with probability* $1 - \mathsf{negl}(\lambda)$*, given a* $(n_1, \ldots, n_m)$*-tree of accepting transcripts for* $\mathbb{x}$ *where the Verifier's challenges are sampled from a space* Q *such that* $|Q| = O(2^\lambda)$*, and* $\prod_i n_i = poly(\lambda)$*.*

4 The Mova folding scheme

Throughout the rest of the paper we fix a security parameter λ and a finite field $\mathbb{F}$ with $|\mathbb{F}|^{-1} = \mathsf{negl}(\lambda)$. Further, we let pp denote public parameters $\mathsf{pp}_{\mathsf{R1CS}} = (\mathbb{F}, m, n, \ell, A, B, C)$, where $m, n, \ell \geq 0$ are nonnegative integers with $m \leq |\mathbb{F}|$; A, B, C are $m \times m$ matrices each with at most $\Omega(n)$ nonzero entries in $\mathbb{F}$. The parameter ℓ denotes the size of public input vectors. We fix a vector commitment scheme $(\mathsf{Gen}_{\mathsf{com}}, \mathsf{Com}, \mathsf{Open})$ for vectors in $\mathbb{F}^m$, and public parameters $\mathsf{pp}_{\mathsf{Com}} \leftarrow \mathsf{Gen}_{\mathsf{com}}(1^\lambda, m)$. Then, we let $\mathsf{pp} = (\mathsf{pp}_{\mathsf{R1CS}}, \mathsf{pp}_{\mathsf{com}})$.

In this section we describe a reduction of knowledge from $\mathbf{R}_{\mathsf{R1CS}} \times \mathbf{R}_{\mathsf{acc}}$ to $\mathbf{R}_{\mathsf{acc}}$. Along the way, we also construct a reduction of knowledge from $\mathbf{R}_{\mathsf{acc}} \times \mathbf{R}_{\mathsf{acc}}$ to $\mathbf{R}_{\mathsf{acc}}$. Here $\mathbf{R}_{\mathsf{R1CS}}$ denotes the R1CS relation:

$$\mathbf{R}_{\mathsf{R1CS}} := \left\{ (\mathsf{pp}_{\mathsf{R1CS}}, \mathbb{x} = \mathbf{x}; \mathbb{w} = \mathbf{W}) \;\middle|\; \begin{array}{l} \mathbf{x} \in \mathbb{F}^\ell,\ \mathbf{W} \in \mathbb{F}^{m-\ell-1} \\ (A \cdot \mathbf{Z}) \circ (B \cdot \mathbf{Z}) = C \cdot \mathbf{Z} \\ Z = (\mathbf{W}, \mathbf{x}, 1) \end{array} \right\}. \tag{7}$$

The relation $\mathbf{R}_{\mathsf{acc}}$ is defined as

$$\mathbf{R}_{\mathsf{acc}} := \left\{ \begin{pmatrix} \mathsf{pp}, \\ \mathbb{x} = (\mathbf{x}, v, u, \overline{\mathbf{W}}, \mathbf{r}); \\ \mathbb{w} = (\mathbf{W}, \mathbf{E}, s) \end{pmatrix} \middle| \begin{array}{l} \mathbf{x} \in \mathbb{F}^{\ell},\ \mathbf{E} \in \mathbb{F}^{m}, \mathbf{r} \in \mathbb{F}^{\log m}, \\ v, s \in \mathbb{F},\ \mathbf{W} \in \mathbb{F}^{m-\ell-1} \\ (A \cdot \mathbf{Z}) \circ (B \cdot \mathbf{Z}) = u \cdot (C \cdot \mathbf{Z}) + \mathbf{E} \\ \mathbf{Z} = (\mathbf{W}, \mathbf{x}, u) \\ \mathsf{Com}(\mathsf{pp}_{\mathsf{Com}}, \mathbf{W}, s) = \overline{\mathbf{W}}, \\ \mathsf{mle}[\mathbf{E}](\mathbf{r}) = v \end{array} \right\} . \tag{8}$$

To construct our reduction of knowledge from $\mathbf{R}_{\mathsf{R1CS}} \times \mathbf{R}_{\mathsf{acc}}$ to $\mathbf{R}_{\mathsf{acc}}$ we describe three separate reductions of knowledge, and then we apply the sequential and parallel composition theorems from [KP23] (cf. Theorems 3.3 and 3.4) to obtain our desired protocol. Precisely, we describe reductions of knowledge for:

- **$\mathbf{R}_{\mathsf{R1CS}}$ to $\mathbf{R}_{\mathsf{acc}}$.** This reduction transforms an instance-witness pair from $\mathbf{R}_{\mathsf{R1CS}}$ into an instance-witness pair from $\mathbf{R}_{\mathsf{acc}}$.
- **$\mathbf{R}_{\mathsf{acc}} \times \mathbf{R}_{\mathsf{acc}}$ to $\mathbf{R}_{\mathsf{equal}}$.** Here we define $\mathbf{R}_{\mathsf{equal}}$ as follows:

  $$\mathbf{R}_{\mathsf{equal}} = \left\{ (\mathsf{pp}, (\mathbb{x}_1, \mathbb{x}_2); (\mathbb{w}_1; \mathbb{w}_2)) \middle| \begin{array}{l} (\mathsf{pp}, \mathbb{x}_i; \mathbb{w}_i) \in \mathbf{R}_{\mathsf{acc}},\ i = 1, 2, \\ \mathbb{x}_1.\mathbf{r} = \mathbb{x}_2.\mathbf{r} \end{array} \right\} .$$

  In words, $\mathbf{R}_{\mathsf{equal}}$ is the set of pairs of instance-witness tuples from $\mathbf{R}_{\mathsf{acc}}$ that have the same evaluation point.
- **$\mathbf{R}_{\mathsf{equal}}$ to $\mathbf{R}_{\mathsf{acc}}$.** Intuitively, this reduction transforms two instance-witness pairs $(\mathsf{pp}, \mathbb{x}_1; \mathbb{w}_1), (\mathsf{pp}, \mathbb{x}_2; \mathbb{w}_2)$ from $\mathbf{R}_{\mathsf{acc}}$ with $\mathbb{x}_1.\mathbf{r} = \mathbb{x}_2.\mathbf{r}$, into an instance-witness pair from $\mathbf{R}_{\mathsf{acc}}$.

In Section 4.4 we put together these reductions of knowledge and describe the full Mova folding scheme.

4.1 From $\mathbf{R}_{\mathsf{R1CS}}$ to $\mathbf{R}_{\mathsf{acc}}$

In Protocol 5 we describe Mova's reduction of knowledge from $\mathbf{R}_{\mathsf{R1CS}}$ to $\mathbf{R}_{\mathsf{acc}}$. In Lemma 4.1 we prove that the protocol is, indeed, a reduction of knowledge. In Remark 4.3 we explain why the proof of Lemma 4.1 is more involved than one would expect at first.

Protocol 5 Mova's reduction of knowledge $\mathbf{R}_{\mathsf{R1CS}} \to \mathbf{R}_{\mathsf{acc}}$

Input: P receives $(\mathsf{pp}, \mathbf{x}; \mathbf{W}) \in \mathbf{R}_{\mathsf{R1CS}}$. V receives $(\mathsf{pp}, \mathbf{x})$.

```
1: P samples $s \leftarrow \mathbb{F}$, computes the commitment $\overline{\mathbf{W}} := \mathsf{Com}(\mathsf{pp}_{\mathsf{Com}}, \mathbf{W}, s)$ and sends it to V.
2: V samples a random vector $\mathbf{r} \leftarrow \mathbb{F}^{\log m}$ and sends $\mathbf{r}$ to P.
3: P and V set $v = 0, u = 1$, and output $\mathbb{x}_{\mathsf{acc}}$ where $\mathbb{x}_{\mathsf{acc}} = (\mathbf{x}, v, u, \overline{\mathbf{W}}, \mathbf{r})$. Additionally, P outputs
   $\mathbb{w}_{\mathsf{acc}} = (\mathbf{W}, \mathbf{E} = \mathbf{0}, s)$.
```

Lemma 4.1. *Protocol 5 is a reduction of knowledge from $\mathbf{R}_{\mathsf{R1CS}}$ to $\mathbf{R}_{\mathsf{acc}}$.*

Proof. Public reducibility. We construct a deterministic function φ, with input $(\mathsf{pp}, \mathbb{x}, \tau)$, that outputs $\mathbb{x}'$. Suppose $\mathbb{x} = (\mathbf{x})$ and $\tau = (\overline{\mathbf{W}}, \mathbf{r})$, otherwise the function φ aborts. The function φ takes the values $\overline{\mathbf{W}}$ and $\mathbf{r}$ from the transcript τ, then outputs $(\mathbf{x}, v = 0, u = 0, \overline{\mathbf{W}}, \mathbf{r})$.

Let $\mathcal{A}, \mathsf{P}^*$ be PPT adversaries and let $(\mathbb{x}, \mathsf{st}) \leftarrow \mathcal{A}(\mathsf{pp})$. Let τ be a transcript of the interaction between P^* and V, with input $(\mathsf{pp}, \mathbb{x}, \mathsf{st})$, following Protocol 5. Let $(\mathbb{x}'; \mathbb{w}')$ be the output of this interaction. It is not hard to see that $\varphi(\mathsf{pp}, \mathbb{x}, \tau)$ is the output that V obtains from the interaction with P^*, following Protocol 5, after having received $\overline{\mathbf{W}}$ and sampled $\mathbf{r}$.

Perfect completeness. Let $\mathcal{A}$ be a PPT adversary and let $(\mathbb{x}, \mathbb{w}) \leftarrow \mathcal{A}(\mathsf{pp})$ be such that $(\mathsf{pp}, \mathbb{x}; \mathbb{w}) \in \mathbf{R}_{\mathsf{R1CS}}$. Write $\mathbb{x} = (\mathbf{x})$ and $\mathbb{w} = (\mathbf{W})$. Let $\mathbb{x}' = (\mathbf{x}, v = 0, u = 0, \overline{\mathbf{W}}, \mathbf{r})$ and $\mathbb{w}' = (\mathbf{W}, \mathbf{E} = \mathbf{0}, s)$ be the outputs of V and P after honestly executing Protocol 5 with inputs $(\mathsf{pp}, \mathbb{x}; \mathbb{w})$. We have that $(\mathsf{pp}, \mathbb{x}'; \mathbb{w}')$ is in $\mathbf{R}_{\mathsf{acc}}$. Indeed, the relation $(A \cdot \mathbf{Z}) \circ (B \cdot \mathbf{Z}) = u \cdot (C \cdot \mathbf{Z}) + \mathbf{E}$ is satisfied by the choice of $\mathbf{E} = \mathbf{0}$, $u = 1$, and by the fact that $(\mathsf{pp}, \mathbb{x}; \mathbb{w}) \in \mathbf{R}_{\mathsf{R1CS}}$. Moreover, since $\mathbf{E} = 0$, also $\mathsf{mle}[\mathbf{E}](\mathbf{r}) = v = 0$.

Knowledge soundness. Let $\mathcal{A}$ and P^* be expected polynomial-time adversaries. Let $(\mathbb{x}, \mathsf{st}) \leftarrow \mathcal{A}(\mathsf{pp})$, with $\mathbb{x} = (\mathbf{x})$.

Fix the notation $\langle \mathsf{P}^*, \mathsf{V} \rangle = \langle \mathsf{P}^*, \mathsf{V} \rangle(\mathsf{pp}, \mathbb{x}, \mathsf{st})$, i.e. $\langle \mathsf{P}^*, \mathsf{V} \rangle$ is interactive protocol in which P^* and V interact following Protocol 3, with inputs $\mathsf{pp}, \mathbb{x}$ and st. We also look at $\langle \mathsf{P}^*, \mathsf{V} \rangle$ as a random variable modelling the output of such interaction. Let $\varepsilon_{\mathsf{total}} = \Pr[(\mathsf{pp}, \langle \mathsf{P}^*, \mathsf{V} \rangle(\mathsf{pp}, \mathbb{x}, \mathsf{st})) \in \mathbf{R}_{\mathsf{acc}}]$.

The extractor Ext proceeds as follows. Ext receives $(\mathsf{pp}, \mathbb{x}, \mathsf{st})$ as inputs. Then:

- **Step 1.** Ext runs the protocol $\langle \mathsf{P}^*, \mathsf{V} \rangle$ once. Let $(\mathsf{pp}, \mathbb{x}^{(1)}; \mathbb{w}^{(1)})$ be the output of this interaction. If $(\mathsf{pp}, \mathbb{x}^{(1)}; \mathbb{w}^{(1)}) \notin \mathbf{R}_{\mathsf{acc}}$, then Ext aborts. Otherwise, $(\mathsf{pp}, \mathbb{x}^{(1)}; \mathbb{w}^{(1)}) \in \mathbf{R}_{\mathsf{acc}}$. Say we have
$$\mathbb{x}^{(1)} = (\mathbf{x}, v^{(1)} = 0, u^{(1)} = 1, \overline{\mathbf{W}}, \mathbf{r}^{(1)}), \quad \mathbb{w}^{(1)} = (\mathbf{W}^{(1)}, \mathbf{E}^{(1)}, s^{(1)}).$$
If $\mathbf{E}^{(1)} = \mathbf{0}$, then Ext terminates and outputs $\mathbf{W}^{(1)}$.

- **Step 2.** Next, Ext repeatedly runs $\langle \mathsf{P}^*, \mathsf{V} \rangle$, keeping always the same first message sent by P^* to be $\overline{\mathbf{W}}$. To do so, Ext rewinds P^* only to the point where P^* has already sent $\overline{\mathbf{W}}$.

  As soon as Ext obtains an output $(\mathsf{pp}, \mathbb{x}^{(2)}; \mathbb{w}^{(2)}) \in \mathbf{R}_{\mathsf{acc}}$, Ext terminates and outputs $\mathbf{W}^{(1)}$.

Let $\mathcal{E}_{\overline{\mathbf{W}}}$ be the event that P^*'s first message in Step 1 of Ext is $\overline{\mathbf{W}}$. Fix one such first message $\overline{\mathbf{W}}$, and let ε be the probability that $(\mathsf{pp}, \langle \mathsf{P}^*, \mathsf{V} \rangle(\mathsf{pp}, \mathbb{x}, \mathsf{st})) \in \mathbf{R}_{\mathsf{acc}}$. We next prove that, conditioned on $\mathcal{E}_{\overline{\mathbf{W}}}$, Ext runs in expected polynomial time and outputs a valid witness for $(\mathbb{x})$ with probability $\varepsilon - \mathsf{negl}(\lambda)$. For readability purposes, we avoid for now referring to $\overline{\mathbf{W}}$ in our notation. In what follows, unless stated otherwise, we consider all probabilities and events referring to Ext as conditioned on $\mathcal{E}_{\overline{\mathbf{W}}}$.

First of all, we prove that Ext terminates in expected polynomial time. Notice that Ext always runs $\langle \mathsf{P}^*, \mathsf{V} \rangle$ once in Step 1. Let $\mathcal{E}$ be the event that Ext does not abort in

Step 1, which happens with probability ε. Denote by Z the random variable (over Ext's random coins) representing the number of times Ext runs the interaction $\langle \mathsf{P}^*, \mathsf{V} \rangle$ in Step 2 (if $\mathbf{E}^{(1)} = \mathbf{0}$ and, thus, Ext terminated, then $Z = 0$). Let Z_{total} be the random variable (over Ext's random coins) representing the total number of times $\langle \mathsf{P}^*, \mathsf{V} \rangle$ is run when executing Ext. We have

$$\mathbb{E}[Z_{\mathsf{total}}] = 1 + \Pr[\mathcal{E}] \cdot \mathbb{E}[Z \mid \mathcal{E}] \leq 1 + \varepsilon \cdot \frac{1}{\varepsilon} = 2. \tag{9}$$

Hence, Ext runs in expected polynomial time, and it does not abort with probability at least ε.

Let $\mathcal{E}_{\mathsf{binding}}$ be the event that Ext, on inputs $(\mathsf{pp}_{\mathsf{Com}}, \mathbb{x}, \mathsf{st})$ is not able to break the binding property of the commitment scheme Com, i.e. $\mathcal{E}_{\mathsf{binding}}$ is the event that at no point Ext has computed two vectors $\mathbf{U}_1, \mathbf{U}_2$ and elements s_1, s_2 such that $\mathsf{Com}(\mathsf{pp}_{\mathsf{Com}}, \mathbf{U}_1, s_1) = \mathsf{Com}(\mathsf{pp}_{\mathsf{Com}}, \mathbf{U}_2, s_2)$. We have $\mathsf{Pr}[\mathcal{E}_{\mathsf{binding}}] = 1 - \mathsf{negl}(\lambda)$. Let $\mathcal{E}_{\mathsf{zero}}$ be the event that both $\mathcal{E}$ occurs and the vector $\mathbf{E}^{(1)}$ output at the end of Step 1 satisfies $\mathbf{E}^{(1)} = \mathbf{0}$. Similarly, let $\mathcal{E}_{\mathsf{nonzero}}$ be the event that $\mathcal{E}$ occurs and $\mathbf{E}^{(1)} \neq \mathbf{0}$. Now, conditioning on $\mathcal{E}$ occurring, these two events are complementary and mutually exclusive. Hence, by the law of total expectation,

$$\begin{aligned}
&\mathbb{E}[Z \mid \mathcal{E}] = \mathbb{E}[Z \mid \mathcal{E}_{\mathsf{nonzero}} \text{ or } \mathcal{E}_{\mathsf{zero}}] \\
=&\mathbb{E}[Z \mid \mathcal{E}_{\mathsf{nonzero}}]\mathsf{Pr}[\mathcal{E}_{\mathsf{nonzero}}] + \mathbb{E}[Z \mid \mathcal{E}_{\mathsf{zero}}]\mathsf{Pr}[\mathcal{E}_{\mathsf{zero}}] \\
=&\mathbb{E}[Z \mid \mathcal{E}_{\mathsf{nonzero}}]\varepsilon_1 + \mathbb{E}[Z \mid \mathcal{E}_{\mathsf{zero}}](1 - \varepsilon_1),
\end{aligned} \tag{10}$$

where $\varepsilon_1 = \mathsf{Pr}[\mathcal{E}_{\mathsf{nonzero}}]$. Note that $\mathbb{E}[Z \mid \mathcal{E}_{\mathsf{zero}}] = 0$, because if, at the end of Step 1, $\mathbf{E}^{(1)} = \mathbf{0}$, then the extractor terminates. We now make the following claim:

Claim 4.2. $\mathbb{E}[Z \mid \mathcal{E}_{\mathsf{nonzero}}, \mathcal{E}_{\mathsf{binding}}]^{-1} = \mathsf{negl}(\lambda)$.

Assume Claim 4.2 is true for now. We next argue that $\varepsilon_1 = \mathsf{negl}(\lambda)$. Note that this will complete the proof (barring the proof of Claim 4.2)that Ext runs in PPT time and outputs a valid witness with probability $\varepsilon - \mathsf{negl}(\lambda)$, conditioned on P^*'s first message being $\overline{\mathbf{W}}$, i.e. conditioned on the event $\mathcal{E}_{\mathbf{W}}$. Indeed, if Ext does not abort and $\mathbf{E}^{(1)} = \mathbf{0}$, then clearly $(\mathsf{pp}, \mathbb{x}^{(1)}; \mathbb{w}^{(1)}) \in \mathbf{R}_{\mathsf{R1CS}}$. Further, we have already argued that Ext runs in expected polynomial time, and that it does not abort with probability at least ε. If, additionally, the probability that Ext does not abort and $\mathbf{E}^{(1)} \neq \mathbf{0}$ is negligible, i.e. $\varepsilon_1 = \mathsf{negl}(\lambda)$, then we conclude that Ext is a PPT algorithm that outputs a valid witness for $\mathbb{x}$ with probability $\varepsilon - \mathsf{negl}(\lambda)$.

We now prove that $\varepsilon_1 = \mathsf{negl}(\lambda)$, assuming that Claim 4.2 is true. Indeed, plugging (10) into (9), and using $\mathbb{E}[Z \mid \mathcal{E}_{\mathsf{zero}}] = 0$, we obtain

$$2 \geq \mathbb{E}[Z_{\mathsf{total}}] = 1 + \varepsilon\mathbb{E}[Z \mid \mathcal{E}] = 1 + \varepsilon\varepsilon_1\mathbb{E}[Z \mid \mathcal{E}_{\mathsf{nonzero}}] \tag{11}$$

Using again the law of total expectation,

$$\begin{aligned}
&\mathbb{E}[Z \mid \mathcal{E}_{\mathsf{nonzero}}] \\
=&\mathbb{E}[Z \mid \mathcal{E}_{\mathsf{nonzero}}, \mathcal{E}_{\mathsf{binding}}]\mathsf{Pr}[\mathcal{E}_{\mathsf{binding}}] + \mathbb{E}[Z \mid \mathcal{E}_{\mathsf{nonzero}}, \neg\mathcal{E}_{\mathsf{binding}}]\mathsf{Pr}[\neg\mathcal{E}_{\mathsf{binding}}] \\
\geq&\mathbb{E}[Z \mid \mathcal{E}_{\mathsf{nonzero}}, \mathcal{E}_{\mathsf{binding}}]\mathsf{Pr}[\mathcal{E}_{\mathsf{binding}}].
\end{aligned} \tag{12}$$

Note that $\varepsilon_1 \le \varepsilon$, since ε_1 is the probability that Ext does not abort at Step 1, which occurs with probability ε, and, additionally, $\mathbf{E}^{(1)} \neq 0$. Hence from (11), (12), and Claim 4.2 we obtain

$$\varepsilon_1^2 \le \varepsilon_1\varepsilon \le \mathbb{E}[Z \mid \mathcal{E}_{\mathsf{nonzero}}, \mathcal{E}_{\mathsf{binding}}]^{-1} \mathsf{Pr}[\mathcal{E}_{\mathsf{binding}}]^{-1} = \mathsf{negl}(\lambda),$$

where the last equality follows from Remark 3.1 and the fact that $\mathbb{E}[Z \mid \mathcal{E}_{\mathsf{nonzero}}, \mathcal{E}_{\mathsf{binding}}]^{-1} = \mathsf{negl}(\lambda)$ and $\mathsf{Pr}[\mathcal{E}_{\mathsf{binding}}] = 1 - \mathsf{negl}(\lambda)$. This implies that $\varepsilon_1 = \mathsf{negl}(\lambda)$, as needed. It only remains to prove Claim 4.2.

Proof of Claim 4.2. Assume $\mathcal{E}_{\mathsf{binding}}$ holds, i.e. Ext does not break the binding property of the commitment scheme Com. Assume further we run Ext up to Step 1 and $\mathcal{E}_{\mathsf{nonzero}}$ holds. Then Ext does not abort. Let $\mathbb{x}^{(1)} = (\mathbf{x}, v^{(1)} = 0, u^{(1)} = 1, \overline{\mathbf{W}}, \mathbf{r}^{(1)})$, $\mathbb{w}^{(1)} = (\mathbf{W}^{(1)}, \mathbf{E}^{(1)}, s^{(1)})$ be the output of $\langle \mathsf{P}^*, \mathsf{V}\rangle$ at the end of Step 1. By assumption $\mathbf{E}^{(1)} \neq \mathbf{0}$.

By construction of Ext, during Step 2, Ext successively repeats an experiment Ξ, until Ξ is successful. The experiment Ξ consists in running $\langle \mathsf{P}^*, \mathsf{V}\rangle$, and Ξ is successful if the output $(\mathbb{x}^{(2)}, \mathbb{w}^{(2)})$ of $\langle \mathsf{P}^*, \mathsf{V}\rangle$ is in $\mathbf{R}_{\mathsf{acc}}$. Importantly, note that the random challenge $\mathbf{r}^{(2)}$ sent by V during this experiment is uniformly random and independent of $\mathbf{E}^{(1)}$.

Let $\mathbb{x}^{(2)} = (\mathbf{x}, v^{(2)} = 0, u^{(2)} = 1, \overline{\mathbf{W}}, \mathbf{r}^{(2)})$, $\mathbb{w}^{(2)} = (\mathbf{W}^{(2)}, \mathbf{E}^{(2)}, s^{(2)})$ be an output of $\langle \mathsf{P}^*, \mathsf{V}\rangle$ obtained after running the experiment Ξ, not necessarily successfully. We argue that if $(\mathsf{pp}, \mathbb{x}^{(2)}, \mathbb{w}^{(2)}) \in \mathbf{R}_{\mathsf{acc}}$, then $\mathsf{mle}[\mathbf{E}^{(1)}](\mathbf{r}^{(2)}) = 0$.

Indeed, assume $(\mathsf{pp}, \mathbb{x}^{(2)}, \mathbb{w}^{(2)}) \in \mathbf{R}_{\mathsf{acc}}$, and set $\mathbf{Z}^{(i)} = (\mathbf{W}^{(i)}, \mathbf{x}, 1)$ for $i = 1, 2$. Let $\mathbf{E}^{(i)} = (A \cdot \mathbf{Z}^{(i)}) \circ (B \cdot \mathbf{Z}^{(i)}) - (C \cdot \mathbf{Z}^{(i)})$. Since $(\mathsf{pp}, \mathbb{x}^{(1)}; \mathbb{w}^{(1)})$ and $(\mathsf{pp}, \mathbb{x}^{(2)}; \mathbb{w}^{(2)})$ are in $\mathbf{R}_{\mathsf{acc}}$, we have

$$\mathsf{Com}(\mathsf{pp}_{\mathsf{Com}}, \mathbf{W}^{(1)}, s^{(1)}) = \overline{\mathbf{W}} = \mathsf{Com}(\mathsf{pp}_{\mathsf{Com}}, \mathbf{W}^{(2)}, s^{(2)}).$$

Since we assumed $\mathcal{E}_{\mathsf{binding}}$ holds, it must hold that $\mathbf{W}^{(1)} = \mathbf{W}^{(2)}$, and hence also $\mathbf{Z}^{(1)} = \mathbf{Z}^{(2)}$. Using again that $(\mathsf{pp}, \mathbb{x}^{(1)}; \mathbb{w}^{(1)})$ and $(\mathsf{pp}, \mathbb{x}^{(2)}; \mathbb{w}^{(2)})$ are in $\mathbf{R}_{\mathsf{acc}}$, we have that, for $i = 1, 2$,

$$\mathbf{E}^{(i)} = (A \cdot \mathbf{Z}^{(i)}) \circ (B \cdot \mathbf{Z}^{(i)}) - (C \cdot \mathbf{Z}^{(i)}). \tag{13}$$

Using twice Eq. (13), the fact that $\mathbf{Z}^{(1)} = \mathbf{Z}^{(2)}$, and that $\mathsf{mle}[\mathbf{E}^{(2)}](\mathbf{r}^{(2)}) = 0$, we obtain

$$\begin{aligned} \mathsf{mle}[\mathbf{E}^{(1)}](\mathbf{r}^{(2)}) &= \mathsf{mle}[(A \cdot \mathbf{Z}^{(1)}) \circ (B \cdot \mathbf{Z}^{(1)}) - (C \cdot \mathbf{Z}^{(1)})](\mathbf{r}^{(2)}) \\ &= \mathsf{mle}[(A \cdot \mathbf{Z}^{(2)}) \circ (B \cdot \mathbf{Z}^{(2)}) - (C \cdot \mathbf{Z}^{(2)})](\mathbf{r}^{(2)}) = \mathsf{mle}[\mathbf{E}^{(2)}](\mathbf{r}^{(2)}) = 0, \end{aligned} \tag{14}$$

as required.

Hence, for Ξ to be successful, it is necessary that $\mathbf{r}^{(2)}$ is a root of $\mathsf{mle}[\mathbf{E}^{(1)}](X)$. Since $\mathbf{E}^{(1)} \neq \mathbf{0}$, we have $\mathsf{mle}[\mathbf{E}^{(1)}](X) \neq 0$. Then, by Schwartz-Zippel lemma, and because $\mathbf{r}^{(2)}$ is sampled uniformly at random after $\mathbf{E}^{(1)}$ is determined, the probability that Ξ is successful is at most $|\mathbb{F}|^{-1} = \mathsf{negl}(\lambda)$. Since $\mathbb{E}[Z \mid \mathcal{E}_{\mathsf{nonzero}}] = \mathsf{Pr}[\Xi \text{ is successful}]^{-1}$, we conclude that $\mathbb{E}[Z \mid \mathcal{E}_{\mathsf{nonzero}}]^{-1} = \mathsf{negl}(\lambda)$. This completes the proof of Claim 4.2. □

Recall that $\mathcal{E}_{\overline{\mathbf{W}}}$ denotes the event that P^*'s first message at Step 1 of Ext is $\overline{\mathbf{W}}$. We have so far proved that, conditioned on $\mathcal{E}_{\overline{\mathbf{W}}}$, Ext runs in expected polynomial time and outputs a valid witness for $\mathbb{x}$ with probability $\varepsilon - \mathsf{negl}(\lambda)$, where ε is the probability that $(\mathsf{pp}, \langle \mathsf{P}^*, \mathsf{V}\rangle) \in \mathbf{R}_{\mathsf{R1CS}}$, conditioned on P^*'s first message being $\overline{\mathbf{W}}$. Let us denote now ε by $\varepsilon_{\overline{\mathbf{W}}}$. Since the aforementioned negligible function $\mathsf{negl}(\lambda)$ also depends on $\overline{\mathbf{W}}$, we denote it by $\nu_{\overline{\mathbf{W}}}(\lambda)$.

Clearly, it follows that Ext runs in expected polynomial time, regardless of what is P^*'s first message in Step 1. We prove that, also, Ext outputs a valid witness with probability $\varepsilon_{\mathsf{total}} - \mathsf{negl}(\lambda)$, no matter what is P^*'s first message.

Consider the function

$$\nu(\lambda) = \max_{\overline{\mathbf{W}}}\{\nu_{\overline{\mathbf{W}}}(\lambda)\},$$

where the maximum is taken over the (finite) set of all possible commitments $\overline{\mathbf{W}}$. Note that $\nu(\lambda) = \mathsf{negl}(\lambda)$. Now, by the law of total expectation and what we have proved so far,

$$\begin{aligned}
\varepsilon_{\mathsf{total}} \geq & \Pr[(\mathsf{pp}; \mathbb{x}, \mathsf{Ext}(\mathsf{pp}, \mathbb{x}, \mathsf{st}) \in \mathbf{R}_{\mathsf{R1CS}}] \\
= & \sum_{\overline{\mathbf{W}}} \Pr[(\mathsf{pp}; \mathbb{x}, \mathsf{Ext}(\mathsf{pp}, \mathbb{x}, \mathsf{st})) \in \mathbf{R}_{\mathsf{R1CS}} \mid \mathcal{E}_{\overline{\mathbf{W}}}] \cdot \Pr[\mathcal{E}_{\overline{\mathbf{W}}}] \\
= & \sum_{\overline{\mathbf{W}}} (\varepsilon_{\overline{\mathbf{W}}} - \nu_{\overline{\mathbf{W}}}(\lambda)) \cdot \Pr[\mathcal{E}_{\overline{\mathbf{W}}}] \geq \sum_{\overline{\mathbf{W}}} (\varepsilon_{\overline{\mathbf{W}}} - \nu(\lambda)) \cdot \Pr[\mathcal{E}_{\overline{\mathbf{W}}}] \\
= & \left(\sum_{\overline{\mathbf{W}}} \varepsilon_{\overline{\mathbf{W}}} \cdot \Pr[\mathcal{E}_{\overline{\mathbf{W}}}] \right) - \left(\nu(\lambda) \sum_{\overline{\mathbf{W}}} \Pr[\mathcal{E}_{\overline{\mathbf{W}}}] \right) \\
= & \varepsilon_{\mathsf{total}} - \nu(\lambda) \cdot 1 = \varepsilon_{\mathsf{total}} - \mathsf{negl}(\lambda),
\end{aligned}$$

where the summations run over all possible commitments $\overline{\mathbf{W}}$. This completes the proof of the lemma. □

Remark 4.3 (On the knowledge soundness proof of Lemma 4.5)**.** The proof of knowledge soundness in Lemma 4.1 may look at first glance as being more complex than it needs to be. Here we informally discuss some difficulties in it. First, it is intuitively convincing that the knowledge soundness of Protocol 5 follows from the Schwartz-Zippel lemma. As such, at first glance, it seems like the tree extraction lemma (Lemma 3.5) is a perfect tool for proving that Protocol 5 is knowledge sound: one would simpy need to take a tree of accepting transcripts of arity large enough that would force the output error vector $\mathbf{E}$[5] to be $\mathbf{0}$. However, it is not possible to make this argument work if we restrict (as we must) to polynomially sized trees.

The next natural approach is to directly describe an extractor Ext that executes a single run of $\langle \mathsf{P}^*, \mathsf{V} \rangle$. If the output of this interaction is a valid instance-witness $(\mathbf{x}, v, u, \overline{\mathbf{W}}.\mathbf{r}; \mathbf{W}, \mathbf{E})$ for $\mathbf{R}_{\mathsf{acc}}$, then Ext simply outputs $\mathbf{W}$. One hopes that Schwartz-Zippel lemma yields then that $\mathbf{E} = \mathbf{0}$ e.w.n.p. However, this argument does not work because $\mathbf{E}$ is output at the end of $\langle \mathsf{P}^*, \mathsf{V} \rangle$, once the random evaluation point $\mathbf{r}$ is already known. Hence, a priori, $\mathbf{r}$ is not independent of $\mathbf{E}$. It does not seem feasible to use the commitment to $\mathbf{W}$ in order to argue that P^* chose $\mathbf{E}$ before knowing $\mathbf{r}$, at least if we only rely on the binding property of the commitment scheme (and we do not want to add extra assumptions).

The next natural approach is to have the extractor execute $\langle \mathsf{P}^*, \mathsf{V} \rangle$ twice, and only produce an output if both interaction outputs belong to $\mathbf{R}_{\mathsf{acc}}$: the binding property of the commitment scheme forces P^* to use the same $\mathbf{W}$ and $\mathbf{E}$ in both outputs, so now the challenge in the second run of $\langle \mathsf{P}^*, \mathsf{V} \rangle$ is indeed independent of $\mathbf{E}$, e.w.n.p. This is mostly correct, but the issue is that now Ext has a success probability of ε^2, which is too low.

[5] $\mathbf{E}$ is the same, e.w.n.p. for all transcripts, due to the binding property of the commitment to $\mathbf{W}$, and due to $\mathbf{E} = (A \cdot \mathbf{Z}) \circ (B \cdot \mathbf{Z}) - uC \cdot \mathbf{Z}$ being uniquely determined by $\mathbf{Z} = (\mathbf{W}, \mathbf{x}, 1)$.

We thus naturally arrive at the extractor as described in our proof (or a similar one). One may be tempted to use a simpler argument to the one in our proof, and say that the challenge $\mathbf{r}_{\mathsf{last}}$ in the output of the last run of $\langle \mathsf{P}^*, \mathsf{V} \rangle$ is independent of $\mathbf{E}$, and then use Schwartz-Zippel lemma to conclude that $\mathbf{E} = \mathbf{0}$. The problem with this argument is that it is not clear whether the random variable counting the number of times that $\langle \mathsf{P}^*, \mathsf{V} \rangle$ is run is independent of $\mathbf{E}$ (here by $\mathbf{E}$ we mean the error term output in Step 1 of our extractor). Since $\mathbf{r}_{\mathsf{last}}$ depends on this random variable, it is unclear then whether $\mathbf{r}_{\mathsf{last}}$ is independent of $\mathbf{E}$. Our proof avoids making this assumption.

4.2 Reduction to common evaluation point

In this section we describe Mova's reductions of knowledge from $\mathbf{R}_{\mathsf{acc}} \times \mathbf{R}_{\mathsf{acc}}$ to $\mathbf{R}_{\mathsf{equal}}$. Recall that $\mathbf{R}_{\mathsf{equal}}$ was defined as the set of pairs of instance-witness tuples from $\mathbf{R}_{\mathsf{acc}}$ with the same evaluation point. More precisely,

$$\mathbf{R}_{\mathsf{equal}} = \left\{ (\mathsf{pp}, (\mathbb{x}_1, \mathbb{x}_2); (\mathbb{w}_1; \mathbb{w}_2)) \;\middle|\; \begin{array}{l} (\mathsf{pp}, \mathbb{x}_i; \mathbb{w}_i) \in \mathbf{R}_{\mathsf{acc}}, \;\; i = 1, 2, \\ \mathbb{x}_1.\mathbf{r} = \mathbb{x}_2.\mathbf{r} \end{array} \right\}.$$

Our reduction of knowledge (see Protocol 6) is based on a classic technique based on first computing a line $\ell : \mathbb{F} \to \mathbb{F}^{\log n}$ passing through the points $\mathbf{x}_1.\mathbf{r}, \mathbf{x}_2.\mathbf{r}$, and then using the composition univariate polynomials $\mathsf{mle}[\mathbf{E}_1] \circ \ell$ and $\mathsf{mle}[\mathbf{E}_2] \circ \ell$ in a clever way. The technique, which we call *point-vs-line*, can be found in Section 4.5.2 of [Tha22].

Protocol 6 Mova's reduction of knowledge $\mathbf{R}_{\mathsf{acc}} \times \mathbf{R}_{\mathsf{acc}} \to \mathbf{R}_{\mathsf{equal}}$

Input: P receives $(\mathsf{pp}, (\mathbb{x}_1, \mathbb{x}_2); (\mathbb{w}_1, \mathbb{w}_2)) \in \mathbf{R}_{\mathsf{acc}}$ as input, for $i = 1, 2$. V receives $(\mathsf{pp}, (\mathbb{x}_1, \mathbb{x}_2))$ as input. Let $\mathbb{x}_i = (\mathbf{x}_i, v_i, u_i, \overline{\mathbf{W}}_i, \mathbf{r}_i)$, $\mathbb{w}_i = (\mathbf{W}_i, \mathbf{E}_i, s_i)$, $i = 1, 2$.

1: P and V define the linear function $\ell : \mathbb{F} \to \mathbb{F}^{\log m}$ satisfying $\ell(0) = \mathbf{r}_1$ and $\ell(1) = \mathbf{r}_2$. Then P sends V the polynomials $h_1(X), h_2(X)$, of degree at most $\log m$, of the form

$$h_i(X) := \mathsf{mle}[\mathbf{E}_i] \circ \ell(X), \quad i = 1, 2.$$

2: V checks that $h_1(0) = v_1$ and $h_2(1) = v_2$, and aborts if this check fails. Then V samples a random challenge $\beta \leftarrow \mathbb{F}$ and sends β to P.

3: P and V set $v_1' = h_1(\beta)$, $v_2' = h_2(\beta)$ and $\mathbf{r}' = \ell(\beta)$.

4: P and V output

$$\mathbb{x}_i' = (\mathbf{x}_i, v_i', u_i, \overline{\mathbf{W}}_i, \mathbf{r}'), \quad i = 1, 2.$$

In addition, P outputs the witnesses $\mathbb{w}_1', \mathbb{w}_2'$, that are defined as

$$\mathbb{w}_i' = (\mathbf{W}_i, \mathbf{E}_i, s_i), \quad i = 1, 2.$$

Lemma 4.4. *Protocol 6 is a reduction of knowledge.*

Proof. The proof is a combination of the ideas used in the proof of Lemma 4.1, together with the proof of the classic point-vs-line argument we have alluded to before. Due to its length, we defer it to Section 7.1. □

4.3 Folding two instances in $\mathbf{R}_{\mathsf{acc}}$ with the same evaluation point

As the last piece of Mova, in Protocol 7 we describe a reduction of knowledge from $\mathbf{R}_{\mathsf{equal}}$ to $\mathbf{R}_{\mathsf{acc}}$.

Protocol 7 Mova's reduction of knowledge $\mathbf{R}_{\mathsf{equal}} \to \mathbf{R}_{\mathsf{acc}}$

Input: P receives $(\mathsf{pp}, \mathbb{x}_i; \mathbb{w}_i) = (\mathsf{pp}, \mathbf{x}_i, v_i, u_i, \overline{\mathbf{W}}_i, \mathbf{r}; \mathbf{W}_i, \mathbf{E}_i, s_i) \in \mathbf{R}_{\mathsf{acc}}$ as input, for $i = 1, 2$. V receives $(\mathsf{pp}, \mathbb{x}_i)$ as input, for $i = 1, 2$.

1: P computes $t := \mathsf{mle}[\mathbf{T}](\mathbf{r})$, where

$$\mathbf{T} := (A \cdot \mathbf{Z}_1) \circ (B \cdot \mathbf{Z}_2) + (A \cdot \mathbf{Z}_2) \circ (B \cdot \mathbf{Z}_1) - u_1 \cdot (C \cdot \mathbf{Z}_2) - u_2 \cdot (C \cdot \mathbf{Z}_1),$$

and $\mathbf{Z}_i := (\mathbf{W}_i, \mathbf{x}_i, u_i)$ for $i = 1, 2$, and sends it to V.

2: V samples a random challenge $\alpha \leftarrow \mathbb{F}$ and sends α to P.

3: Both P and V output $\mathbb{x}_{\mathsf{acc}} = (\mathbf{x}, v, u, \overline{\mathbf{W}}, \mathbf{r})$ where

$$\begin{aligned} \mathbf{x} &= \mathbf{x}_1 + \alpha \mathbf{x}_2, \\ v &= v_1 + \alpha t + \alpha^2 v_2, \\ u &= u_1 + \alpha u_2, \\ \overline{\mathbf{W}} &= \overline{\mathbf{W}}_1 + \alpha \overline{\mathbf{W}}_2. \end{aligned} \tag{15}$$

Additionally, P outputs $\mathbb{w}_{\mathsf{acc}} = (\mathbf{W}, \mathbf{E}, s)$, where

$$\begin{aligned} \mathbf{W} &= \mathbf{W}_1 + \alpha \mathbf{W}_2, \\ \mathbf{E} &= \mathbf{E}_1 + \alpha \mathbf{T} + \alpha^2 \mathbf{E}_2, \\ s &= s_1 + \alpha s_2. \end{aligned} \tag{16}$$

Lemma 4.5. *Protocol 7 is a reduction of knowledge from $\mathbf{R}_{\mathsf{equal}}$ to $\mathbf{R}_{\mathsf{acc}}$.*

Proof. Public reducibility. We construct a deterministic function φ, with input $(\mathsf{pp}, \mathbb{x}_1, \mathbb{x}_2, \tau)$, that outputs $\mathbb{x}'$. Let $\mathbb{x}_i = (\mathbf{x}_i, v_i, u_i, \overline{\mathbf{W}}_i, \mathbf{r})$, $i = 1, 2$, and let $\tau = (t, \alpha)$ (if the input has another form the function aborts). The function φ simply outputs $(\mathbf{x}, v, u, \overline{\mathbf{W}}, \mathbf{r})$, where $\mathbf{x}, v, u, \overline{\mathbf{W}}$ are computed as in Eq. (15).

Let $\mathcal{A}^*, \mathsf{P}^*$ be PPT adversaries and let $(\mathbb{x}_1, \mathbb{x}_2, \mathsf{st}) \leftarrow \mathcal{A}(\mathsf{pp})$. Let τ be a transcript of the interaction between P^* and V, with input $(\mathsf{pp}, \mathbb{x}_1, \mathbb{x}_2, \mathsf{st})$, following Protocol 7. Let $(\mathbb{x}'; \mathbb{w}')$ be the output of this interaction. It is not hard to see that the output of $\varphi(\mathsf{pp}, \mathbb{x}_1, \mathbb{x}_2, \tau)$ coincides with $(\mathbb{x}'; \mathbb{w}')$.

Perfect completeness. Let $\mathcal{A}$ be a PPT adversary $\mathcal{A}$ and let $(\mathbb{x}_1, \mathbb{x}_2; \mathbb{w}_1, \mathbb{w}_2), \leftarrow \mathcal{A}(\mathsf{pp})$ such that $(\mathsf{pp}, \mathbb{x}_1, \mathbb{x}_2; \mathbb{w}_1, \mathbb{w}_2) \in \mathbf{R}_{\mathsf{equal}}$. Write $\mathbb{x}_i = (\mathbf{x}_i, v_i, u_i, \overline{\mathbf{W}}_i, \mathbf{r})$, $\mathbb{w}_i = (\mathbf{W}_i, \mathbf{E}_i, s_i)$, for $i = 1, 2$. Set $\mathbf{Z}_i = (\mathbf{W}_i, \mathbf{x}_i, u_i)$ for $i = 1, 2$.

Let $\mathbb{x} = (\mathbf{x}, v, u, \overline{\mathbf{W}}, \mathbf{r})$ and $\mathbb{w} = (\mathbf{W}, \mathbf{E}, s)$ be the outputs of P and V after honestly following Protocol 7 with inputs $(\mathsf{pp}, \mathbb{x}_1, \mathbb{x}_2; \mathbb{w}_1, \mathbb{w}_2)$. To prove that $(\mathsf{pp}, \mathbb{x}; \mathbb{w}) \in \mathbf{R}_{\mathsf{acc}}$ it suffices to check three properties. First of all that $\mathsf{Com}(\mathsf{pp}_{\mathsf{Com}}, \mathbf{W}, s) = \overline{\mathbf{W}}$, then that $\mathsf{mle}[\mathbf{E}](\mathbf{r}) = v$ and finally that

$$(A \cdot \mathbf{Z}) \circ (B \cdot \mathbf{Z}) = u \cdot (C \cdot \mathbf{Z}) + \mathbf{E}. \tag{17}$$

The first property is immediate by the definition of $\overline{\mathbf{W}}$ and $\mathbf{W}$ in Eqs. (15) and (16) respectively, and using that the commitment scheme Com is additively homomorphic. Similarly, using that $\mathsf{mle}[\mathbf{E}_i](\mathbf{r}) = v_i$ for $i = 1, 2$, that $t = \mathsf{mle}[\mathbf{T}](\mathbf{r})$ and the definition of $\mathbf{E}$ in Eq. (16), we directly obtain that $\mathsf{mle}[\mathbf{E}](\mathbf{r}) = v$.

We now want to prove Eq. (17). The proof of this equality is analogous to the proof of Lemma 6 of [KST21]. Expanding the terms of Eq. (17), we obtain

$$(A \cdot (\mathbf{Z}_1 + \alpha\mathbf{Z}_2)) \circ (B \cdot (\mathbf{Z}_1 + \alpha\mathbf{Z}_2)) = (u_1 + \alpha u_2) \cdot (C \cdot (\mathbf{Z}_1 + \alpha\mathbf{Z}_2)) + \mathbf{E}$$

and by distributing and reordering we obtain

$$\begin{aligned}\mathbf{E} =& (A \cdot \mathbf{Z}_1) \circ (B \cdot \mathbf{Z}_1) + \alpha[(A \cdot \mathbf{Z}_1) \circ (B \cdot \mathbf{Z}_2) + (A \cdot \mathbf{Z}_2) \circ (B \cdot \mathbf{Z}_1)] \\ &+ \alpha^2 (A \cdot \mathbf{Z}_2) \circ (B \cdot \mathbf{Z}_2) - u_1 \cdot (C \cdot \mathbf{Z}_1) - \alpha(u_1 \cdot (C \cdot \mathbf{Z}_2) + u_2 \cdot (C \cdot \mathbf{Z}_1)) \\ &- \alpha^2 u_2 \cdot (C \cdot \mathbf{Z}_2).\end{aligned} \tag{18}$$

By construction, we have the equality

$$\mathbf{T} = (A \cdot \mathbf{Z}_1) \circ (B \cdot \mathbf{Z}_2) + (A \cdot \mathbf{Z}_2) \circ (B \cdot \mathbf{Z}_1) - u_1 \cdot (C \cdot \mathbf{Z}_2) - u_2 \cdot (C \cdot \mathbf{Z}_1).$$

Now by hypothesis $((\mathsf{pp}, \mathbb{x}_1; \mathbb{w}_1), (\mathsf{pp}, \mathbb{x}_2; \mathbb{w}_2)) \in \mathbf{R}_{\mathsf{equal}}$, and in particular

$$\begin{aligned}(A \cdot \mathbf{Z}_1) \circ (B \cdot \mathbf{Z}_1) - u_1 \cdot (C \cdot \mathbf{Z}_1) = \mathbf{E}_1, \\ (A \cdot \mathbf{Z}_2) \circ (B \cdot \mathbf{Z}_2) - u_2 \cdot (C \cdot \mathbf{Z}_2) = \mathbf{E}_2.\end{aligned}$$

Using these equalities, via direct substitution in Eq. (18) we have that Eq. (17) is true if and only if

$$\mathbf{E} = \mathbf{E}_1 + \alpha\mathbf{T} + \alpha^2\mathbf{E}_2,$$

which is satisfied by the definition of $\mathbf{E}$ in Eq. (16).

<u>Knowledge soundness</u>. The proof is similar to the knowledge soundness proof of Nova [KST21]. We prove knowledge soundness by means of Lemma 3.5.

Precisely, let $(\mathbb{x}_1, \mathbb{x}_2)$ be an instance, where $\mathbb{x}_1 = (\mathbf{x}_1, v_1, u_1, \overline{\mathbf{W}}_1, \mathbf{r})$ and $\mathbb{x}_2 = (\mathbf{x}_2, v_2, u_2, \overline{\mathbf{W}}_2, \mathbf{r})$. Suppose we are given three transcripts $\tau^{(1)}, \tau^{(2)}, \tau^{(3)}$ and corresponding outputs

$$(\mathbb{x}^{(1)}, \mathbb{w}^{(1)}), (\mathbb{x}^{(2)}, \mathbb{w}^{(2)}), (\mathbb{x}^{(3)}, \mathbb{w}^{(3)})$$

of the interaction between P and V for the input $(\mathsf{pp}, \mathbb{x}_1, \mathbb{x}_2)$. Assume the three transcripts all share the same first message t, so that, for $i = 1, 2, 3$, $\tau^{(i)} = (t, \alpha^{(i)})$. For each $i = 1, 2, 3$, assume $(\mathsf{pp}, \mathbb{x}^{(i)}; \mathbb{w}^{(i)}) \in \mathbf{R}_{\mathsf{acc}}$, and let

$$(\mathbb{x}^{(i)}, \mathbb{w}^{(i)}) = (\mathbf{x}^{(i)}, v^{(i)}, u^{(i)}, \overline{\mathbf{W}}^{(i)}, \mathbf{r}; \mathbf{W}^{(i)}, \mathbf{E}^{(i)}, s^{(i)})$$

be the output of the interaction between P and V at the end of

Assume further that the three challenges $\alpha^{(1)}, \alpha^{(2)}, \alpha^{(3)}$ are pairwise different. These three trancripts and outputs form a tree of depth 2 and arity 3 (in the sense of Definition 3.5). We describe a PPT extractor Ext that, for any such tree, outputs a valid witness $\mathbb{w}_1, \mathbb{w}_2$ for $\mathbb{x}_1, \mathbb{x}_2$, e.w.n.p.

Before we proceed, we introduce the following terminology: given $k, r \geq 0$, vectors $\mathbf{U}^{(1)}, \ldots, \mathbf{U}^{(k)} \in \mathbb{F}^r$, and pairwise field elements $\beta_1, \ldots, \beta_k$, by *interpolation* of the tuples $(\beta^{(1)}, \mathbf{U}^{(1)}), \ldots, (\beta^{(k)}, \mathbf{U}^{(k)})$ we mean the process of finding vectors $\mathbf{V}_1, \ldots, \mathbf{V}_k \in \mathbb{F}^r$ such that $\mathbf{V}_1 + \beta_i \mathbf{V}_2 + \ldots + \beta_i^{k-1} \mathbf{V}_k = \mathbf{U}^{(i)}$ for all $i \in [k]$. This is achieved by, for each coordinate $j \in [r]$, using standard interpolation on the points $(\beta^{(1)}, \mathbf{U}^{(1)}[j]), \ldots, (\beta^{(k)}, \mathbf{U}^{(k)}[j])$, and then taking $\mathbf{V}_1[j], \ldots \mathbf{V}_k[j]$ be the coefficients of the resulting interpolating polynomial.

We are ready to describe the extractor. First, by interpolating $(\alpha^{(1)}, \mathbf{W}^{(1)}), (\alpha^{(2)}, \mathbf{W}^{(2)})$, Ext finds two vectors $\mathbf{W}_1^*, \mathbf{W}_2^*$ such that

$$\mathbf{W}_1^* + \alpha^{(i)} \mathbf{W}_2^* = \mathbf{W}^{(i)} \quad \text{for} \quad i = 1, 2. \tag{19}$$

(note that here we use $\alpha^{(1)} \neq \alpha^{(2)}$). Similarly, by interpolating the points $(\alpha^{(1)}, s^{(1)}), (\alpha^{(2)}, s^{(2)})$, Ext constructs s_1^*, s_2^* satisfying

$$s_1^* + \alpha^{(i)} s_2^* = s^{(i)} \tag{20}$$

for $i = 1, 2$. Again via interpolation, but this time using

$$(\alpha^{(1)}, \mathbf{E}^{(1)}), (\alpha^{(2)}, \mathbf{E}^{(2)}), (\alpha^{(3)}, \mathbf{E}^{(3)}),$$

Ext obtains vectors $\mathbf{E}_1^*, \mathbf{T}^*$ and $\mathbf{E}_2^*$ satisfying

$$\mathbf{E}_1^* + \alpha^{(i)} \mathbf{T}^* + (\alpha^{(i)})^2 \mathbf{E}_2^* = \mathbf{E}^{(i)} \quad \text{for} \quad i = 1, 2, 3. \tag{21}$$

(Similarly as before, here we have used that the challenges $\alpha^{(i)}$ are pairwise different). The extractor algorithm outputs $\mathbb{w}_1^* = (\mathbf{W}_1^*, \mathbf{E}_1^*, s_1^*)$ and $\mathbb{w}_2^* = (\mathbf{W}_2^*, \mathbf{E}_2^*, s_2^*)$. To complete the proof, it suffices to check that $(\mathsf{pp}, \mathbb{x}_1, \mathbb{x}_2; \mathbb{w}_1^*, \mathbb{w}_2^*) \in \mathbf{R}_{\mathsf{equal}}$.

We claim that $\mathsf{Com}(\mathsf{pp}_{\mathsf{Com}}, \mathbf{W}_i^*, s_i^*) = \overline{\mathbf{W}}_i$ for $i = 1, 2$. Indeed, by Eqs. (19) and (20) and the fact that the final output of the transcripts $\tau^{(1)}, \tau^{(2)}$ is in $\mathbf{R}_{\mathsf{equal}}$, we have for $i = 1, 2$ that

$$\begin{aligned}
&\mathsf{Com}(\mathsf{pp}_{\mathsf{Com}}, \mathbf{W}_1^*, s_1^*) + \alpha^{(i)} \mathsf{Com}(\mathsf{pp}_{\mathsf{Com}}, \mathbf{W}_2^*, s_2^*) \\
&= \mathsf{Com}(\mathsf{pp}_{\mathsf{Com}}, \mathbf{W}_1^* + \alpha^{(i)} \mathbf{W}_2^*, s_1^* + \alpha^{(i)} s_2^*) \\
&= \mathsf{Com}(\mathsf{pp}_{\mathsf{Com}}, \mathbf{W}^{(i)}, s^{(i)}) = \overline{\mathbf{W}}^{(i)} \\
&= \overline{\mathbf{W}}_1 + \alpha^{(i)} \overline{\mathbf{W}}_2.
\end{aligned}$$

The linear polynomials $\mathsf{Com}(\mathsf{pp}_{\mathsf{Com}}, \mathbf{W}_1^*, s_1^*) + X \mathsf{Com}(\mathsf{pp}_{\mathsf{Com}}, \mathbf{W}_2^*, s_2^*)$ and $\overline{\mathbf{W}}_1 + X \overline{\mathbf{W}}_2$ take the same value on the two distinct points $\alpha^{(1)}$ and $\alpha^{(2)}$. Therefore they must be the same polynomial, and so

$$\begin{aligned}
\mathsf{Com}(\mathsf{pp}_{\mathsf{Com}}, \mathbf{W}_1^*, s_1^*) &= \overline{\mathbf{W}}_1, \\
\mathsf{Com}(\mathsf{pp}_{\mathsf{Com}}, \mathbf{W}_2^*, s_2^*) &= \overline{\mathbf{W}}_2.
\end{aligned} \tag{22}$$

Using Eq. (22) we prove that Eq. (19) is valid also for $i = 3$. Indeed

$$\begin{aligned}
&\mathsf{Com}(\mathsf{pp}_{\mathsf{Com}}, \mathbf{W}_1^* + \alpha^{(3)} \mathbf{W}_2^*, s_1^* + \alpha^{(3)} s_2^*) \\
&= \mathsf{Com}(\mathsf{pp}_{\mathsf{Com}}, \mathbf{W}_1^*, s_1^*) + \alpha^{(3)} \mathsf{Com}(\mathsf{pp}_{\mathsf{Com}}, \mathbf{W}_2^*, s_2^*)
\end{aligned}$$

$$\begin{aligned}&= \overline{\mathbf{W}}_1 + \alpha^{(3)}\overline{\mathbf{W}}_2\\&= \overline{\mathbf{W}}^{(3)}\\&= \mathsf{Com}(\mathsf{pp}_{\mathsf{Com}}, \mathbf{W}^{(3)}, s^{(3)}).\end{aligned}$$

By the binding property of Com, e.w.n.p., we have

$$\mathbf{W}_1^* + \alpha^{(3)}\mathbf{W}_2^* = \mathbf{W}^{(3)}. \tag{23}$$

We now prove that $\mathsf{mle}[\mathbf{E}_1^*](\mathbf{r}) = v_1$ and $\mathsf{mle}[\mathbf{E}_2^*](\mathbf{r}) = v_2$. Indeed, using Eq. (21) and the fact that the outputs in $(\mathsf{pp}; \mathbb{x}^{(i)}, \mathbb{w}^{(i)}) \in \mathbf{R}_{\mathsf{acc}}$ for $i = 1, 2, 3$, we have that

$$\begin{aligned}&\mathsf{mle}[\mathbf{E}_1^*](\mathbf{r}) + \alpha^{(i)}\mathsf{mle}[\mathbf{T}^*](\mathbf{r}) + (\alpha^{(i)})^2\mathsf{mle}[\mathbf{E}_2^*](\mathbf{r})\\&= \mathsf{mle}[\mathbf{E}_1^* + \alpha^{(i)}\mathbf{T}^* + (\alpha^{(i)})^2\mathbf{E}_2^*](\mathbf{r})\\&= \mathsf{mle}[\mathbf{E}^{(i)}](\mathbf{r}) = v^{(i)} = v_1 + \alpha^{(i)}t + (\alpha^{(i)})^2 v_2\end{aligned}$$

for $i = 1, 2, 3$. Again, we look at both the left and right-hand side above as two polynomials of degree 2 evaluated at $\alpha^{(i)}$. Since these polynomials take the same values evaluated at the three distinct values $\alpha^{(1)}, \alpha^{(2)}, \alpha^{(3)}$, we obtain that they are the same polynomial, and in particular $\mathsf{mle}[\mathbf{E}_1^*](\mathbf{r}) = v_1$ and $\mathsf{mle}[\mathbf{E}_2^*](\mathbf{r}) = v_2$.

It remains to prove that $(A \cdot \mathbf{Z}_i^*) \circ (B \cdot \mathbf{Z}_i^*) - u \cdot (C \cdot \mathbf{Z}_i^*) = \mathbf{E}_i^*$, where $\mathbf{Z}_i^* := (\mathbf{W}_i^*, \mathbf{x}_i, u_i)$ for $i = 1, 2$. Set $\mathbf{Z}^{(i)} := (\mathbf{W}^{(i)}, \mathbf{x}^{(i)}, u^{(i)})$ for $i = 1, 2, 3$. Since $(\mathsf{pp}, \mathbb{x}^{(i)}; \mathbb{w}^{(i)}) \in \mathbf{R}_{\mathsf{acc}}$, we have that for $i = 1, 2, 3$,

$$(A \cdot \mathbf{Z}^{(i)}) \circ (B \cdot \mathbf{Z}^{(i)}) - u \cdot (C \cdot \mathbf{Z}^{(i)}) = \mathbf{E}^{(i)}. \tag{24}$$

By definition, $\mathbf{x}^{(i)} = \mathbf{x}_1 + \alpha^{(i)}\mathbf{x}_2$ and $u^{(i)} = u_1 + \alpha^{(i)}u_2$ for $i = 1, 2, 3$. Together with Eqs. (19) and (23) we obtain that $\mathbf{Z}^{(i)} = \mathbf{Z}_1^* + \alpha^{(i)}\mathbf{Z}_2^*$ for $i = 1, 2, 3$, e.w.n.p. Expanding Eq. (24), and using Eq. (21), we obtain for $i = 1, 2, 3$, e.w.n.p.,

$$\begin{aligned}&(A \cdot \mathbf{Z}_1^*) \circ (B \cdot \mathbf{Z}_1^*) + \alpha^{(i)}[(A \cdot \mathbf{Z}_1^*) \circ (B \cdot \mathbf{Z}_2^*) + (A \cdot \mathbf{Z}_2^*) \circ (B \cdot \mathbf{Z}_1^*)]\\&+(\alpha^{(i)})^2(A \cdot \mathbf{Z}_2^*) \circ (B \cdot \mathbf{Z}_2^*) - u_1 \cdot (C \cdot \mathbf{Z}_1^*) - \alpha^{(i)}[u_1 \cdot (C \cdot \mathbf{Z}_2^*) + u_2 \cdot (C \cdot \mathbf{Z}_1^*)]\\&-(\alpha^{(i)})^2 u_2 \cdot (C \cdot \mathbf{Z}_2^*) = \mathbf{E}_1^* + \alpha^{(i)}\mathbf{T}^* + (\alpha^{(i)})^2\mathbf{E}_2^*.\end{aligned}$$

Again, we see both the left hand side and the right hand side of the equality as two polynomials of degree evaluated at $\alpha^{(i)}$. Since $\alpha^{(1)}, \alpha^{(2)}, \alpha^{(3)}$ are pairwise different, the two polynomials must coincide coefficient-wise. Hence, e.w.n.p.,

$$(A \cdot \mathbf{Z}_1^*) \circ (B \cdot \mathbf{Z}_1^*) - u \cdot (C \cdot \mathbf{Z}_1^*) = \mathbf{E}_1^*$$

$$(A \cdot \mathbf{Z}_2^*) \circ (B \cdot \mathbf{Z}_2^*) - u \cdot (C \cdot \mathbf{Z}_2^*) = \mathbf{E}_2^*$$

This concludes the proof that $(\mathsf{pp}, \mathbb{x}_1, \mathbb{w}_1^*), (\mathsf{pp}, \mathbb{x}_2, \mathbb{w}_2^*) \in \mathbf{R}_{\mathsf{acc}}$, e.w.n.p. $\square$

4.4 Putting everything together

Protocol 8 describes the Mova folding scheme in its entirety. It is a simple composition of the previous schemes Protocols 5 to 7.

Protocol 8 Complete Mova's reduction of knowledge $\mathbf{R}_{\mathsf{R1CS}} \times \mathbf{R}_{\mathsf{acc}} \to \mathbf{R}_{\mathsf{acc}}$

Input: P receives $(\mathsf{pp}, \mathbb{x}_1; \mathbb{w}_1) \in \mathbf{R}_{\mathsf{R1CS}}$ and $(\mathsf{pp}, \mathbb{x}_2; \mathbb{w}_2) \in \mathbf{R}_{\mathsf{acc}}$ as input. V receives $(\mathsf{pp}, \mathbb{x}_1, \mathbb{x}_2)$ as input.

```
1: P and V follow Protocol 5 with input $(\mathsf{pp}, \mathbb{x}_1; \mathbb{w}_1)$ and $(\mathsf{pp}, \mathbb{x}_1)$, respectively. Let $(\mathbb{x}'_1; \mathbb{w}'_1)$ be the
   output of this interaction.
2: P and V follow Protocol 6 with input $(\mathsf{pp}, \mathbb{x}'_1, \mathbb{x}_2; \mathbb{w}'_1, \mathbb{w}_2)$. Let $(\mathbb{x}_1^{(2)}; \mathbb{w}_1^{(2)}), (\mathbb{x}_2^{(2)}; \mathbb{w}_2^{(2)})$ be the
   output of this interaction.
3: P and V follow Protocol 7, with input $(\mathsf{pp}, \mathbb{x}_1^{(2)}; \mathbb{w}_1^{(2)}), (\mathsf{pp}, \mathbb{x}_2^{(2)}; \mathbb{w}_2^{(2)})$. Let $(\mathbb{x}^{(3)}; \mathbb{w}^{(3)})$ be the
   output of this interaction.
   Finally, P and V output $\mathbb{x}^{(3)}$, and P additionally outputs $\mathbb{w}^{(3)}$.
```

Lemma 4.6. *Protocol 8 is a reduction of knowledge from* $\mathbf{R}_{\mathsf{R1CS}} \times \mathbf{R}_{\mathsf{acc}}$ *to* $\mathbf{R}_{\mathsf{acc}}$.

Proof. Consider the trivial reduction of knowledge Π_{trivial} from $\mathbf{R}_{\mathsf{acc}}$ to $\mathbf{R}_{\mathsf{acc}}$ in which P and V perform no rounds of interaction and simply output their inputs.

By Lemma 4.1, Step 1 is a reduction of knowledge Π_1 from $\mathbf{R}_{\mathsf{R1CS}}$ to $\mathbf{R}_{\mathsf{acc}}$. Step 1 followed by Step 2 is equivalent to the protocol one obtains by composing Π_1 and Π_{trivial} in parallel, yielding a reduction of knowledge $\mathbf{R}_{\mathsf{R1CS}} \times \mathbf{R}_{\mathsf{acc}} \to \mathbf{R}_{\mathsf{acc}} \times \mathbf{R}_{\mathsf{acc}}$, and then running Step 2, which is a reduction of knowledge from $\mathbf{R}_{\mathsf{acc}} \times \mathbf{R}_{\mathsf{acc}}$ to $\mathbf{R}_{\mathsf{equal}}$. By the Parallel and Sequential Composition Theorems 3.3 and 3.4, and by Lemmas 4.1 and 4.4, we obtain that Step 1 followed by Step 2 is a reduction of knowledge from $\mathbf{R}_{\mathsf{R1CS}} \times \mathbf{R}_{\mathsf{acc}}$ to $\mathbf{R}_{\mathsf{equal}}$.

Finally, using again the Sequential Composition Theorem 3.3 and Lemma 4.6, we obtain that sequentially performing the above reduction of knowledge, and then running Step 3 results in a reduction of knowledge from $\mathbf{R}_{\mathsf{R1CS}} \times \mathbf{R}_{\mathsf{acc}}$ to $\mathbf{R}_{\mathsf{acc}}$. □

5 Performance

In this section we present and compare the concrete costs of Mova, Nova [KST21], and Hypernova [KS23b] when applied to the same R1CS structure. The costs are displayed in Table 3. We defer the explanation of how these costs were derived to Section 5.2.

We evaluate the scenario in which the Prover and Verifier are folding an instance-witness pair from $\mathbf{R}_{\mathsf{R1CS}}$ and an instance-witness pair from $\mathbf{R}_{\mathsf{acc}}$.

We consider different scenarios depending on whether the R1CS witness vector $\mathbf{W}$ has "small" or "large" entries. Here, by "small" we mean that entries in $\mathbf{W}$ all belong to the range $\{0, \ldots, |\mathbf{W}| - 1\}$. By "large" we mean that $\mathbf{W}$ has entries sampled randomly in $\mathbb{F}$. In many practical applications, $\mathbf{W}$ does contain small entries, and in this case the cost of committing to it is much lower than if $\mathbf{W}$ contains large entries. Namely, if $\mathbf{W}$ contains small entries, then the cost of committing to $\mathbf{W}$ is roughly equivalent to the cost of computing $|\mathbf{W}|$ group operations [STW23b], which in turn is roughly equivalent to the cost of $15|\mathbf{W}|$ field multiplications [zka]. In Table 7 we provide benchmarks that show that this is, roughly, indeed the case. We refer to [STW23b] for further discussion on this topic.

[6]For the Prover's costs, we only display the number of field multiplications, as these can be up to an order of magnitude more expensive than field additions. For the Verifier's costs, we display the number of additions and multiplications because, in many applications such as IVC or PCD, V's algorithm will be arithmetised and proved recursively.

	P	V	Round(s)
Nova [KST21]	$3n + 5m\,\mathbb{F}$ $2\,\mathbb{G}$ ops., $2\,\mathbb{G}$ exp. Com. vector of m $\mathbb{F}$-elements Com. $\mathbf{W}$	$2\ell\,\mathbb{F}$ $2\,\mathbb{G}$ ops., $2\,\mathbb{G}$ exp.	1
Hypernova [KS23b]	$6n + 14m + O(\sqrt{m})\ \mathbb{F}$ $1\,\mathbb{G}$ op., $1\,\mathbb{G}$ exp. Com. $\mathbf{W}$	$2\ell + O(\log(m))\ \mathbb{F}$ $1\,\mathbb{G}$ op., $1\,\mathbb{G}$ exp.	$\log(m) + O(1)$
Mova (this work)	$3n + 12m + 3\log(m)\ \mathbb{F}$ $1\,\mathbb{G}$ op., $1\,\mathbb{G}$ exp. Com. $\mathbf{W}$	$2\ell + 7\log(m) + 5\,\mathbb{F}$ $1\,\mathbb{G}$ op., $1\,\mathbb{G}$ exp.	3

Table 3: Comparison of the concrete costs of Mova, Nova, and Hypernova. In all cases, we consider the same size bounds m, n, ℓ, meaning that R1CS matrices are in $\mathbb{F}^{m\times m}$ with $n = \Omega(m)$ nonzero entries, and witnesses are in $\mathbb{F}^{m-\ell-1}$.
The symbol $\mathbb{F}$ in the Prover P column indicates the number of field *multiplications*. In the Verifier V column, $\mathbb{F}$ indicates the number of field *multiplications and additions*[6]. We write n group operations (resp. exponentiations) as $n\,\mathbb{G}$ ops. (resp. $n\,\mathbb{G}$ exp.). "Com. vector m elements in $\mathbb{F}$" indicates that a vector of size m with arbitrarily sized entries in $\mathbb{F}$ must be committed. "Com. $\mathbf{W}$" means that the R1CS witness vector $\mathbf{W}$ must be committed.
In Mova and Nova, the cost of computing $\mathbf{T}$ is of $3n + 2m$ field multiplications plus some field additions (see Eq. (25)). If A, B, C are binary matrices, this cost can be replaced by $2m$ (cf. Section 5.2). In Hypernova, computing all the necessary matrix-vector products costs $6n + m$ field multiplications plus some field additions (cf. Section 5.2). If A, B, C are binary, then this cost is only m.

In Table 4 we display the relative speed-up of Mova's Prover with respect to Nova's and Hypernova's Prover. We make this comparison by taking Table 3 and setting different specific choices of parameters.

We assume the commitment scheme is the Pedersen or the KZG scheme over the Pallas curve. We estimate the costs of computing a vector with arbitraily large entries (such as Nova's cross term $\mathbf{T}$) using Table 1 in [Hab22]. For example, the cost of committing to a vector of size $m = 2^{16}$ is estimated to be $349m$ field multiplications in [Hab22]. When $m = 2^{20}$, the same source estimates the cost to be roughly $2^8 m$ field multiplications. When $\mathbf{W}$ has small entries, we estimate the cost of committing to $\mathbf{W}$ as explained above.

As we see in Section 5.2, each major step in Mova's Prover has a cost of no more than, roughly, $2^3 m$ field multiplications (assuming the matrices A, B, C are not very complex), not counting the cost of committing to the R1CS witness $\mathbf{W}$. The dominating costs of Mova are no longer the commitment to $\mathbf{T}$ as in Nova, but either the execution of the point-vs-line reduction of knowledge (Protocol 6), which costs $\approx 4m$ multiplications (Section 5.2), or the computation of $\mathbf{T}$ itself, which costs $\approx 3n$ multiplications, where $n = \Omega(m)$ is a bound on the number of nonzero entries in each matrix A, B, C (cf. Eq. (25)).

In Section 5.1 we present our benchmarks for the Prover's runtime in the three schemes. Then, in Section 5.2 we describe how the costs in Table 3 were computed.

m	n	Binary R1CS matrices?	$\mathbf{W}$ entries	$\frac{\text{Nova Prover}}{\text{Mova Prover}}$	$\frac{\text{Hypernova Prover}}{\text{Mova Prover}}$
2^{16}	m	Yes	Small	13.667	1.074
2^{16}	m	No	Small	12.400	1.167
2^{16}	$3m$	Yes	Small	13.667	1.074
2^{16}	$3m$	No	Small	10.500	1.306
2^{20}	m	Yes	Small	10.222	1.074
2^{20}	m	No	Small	9.300	1.167
2^{20}	$3m$	Yes	Small	10.222	1.074
2^{20}	$3m$	No	Small	7.917	1.306
2^{16}	m	Yes	Large	1.864	1.005
2^{16}	m	No	Large	1.857	1.013
2^{16}	$3m$	Yes	Large	1.864	1.005
2^{16}	$3m$	No	Large	1.844	1.027
2^{20}	m	Yes	Large	2.779	1.014
2^{20}	m	No	Large	2.741	1.035
2^{20}	$3m$	Yes	Large	2.779	1.014
2^{20}	$3m$	No	Large	2.671	1.074

Table 4: Concrete comparison of Mova's Prover with Nova and Hypernova's Prover. The two last columns compare the *approximate* equivalent field *multiplication* work of the Provers of both protocols. Larger numbers are better. The third column indicates whether the R1CS $m \times m$ matrices A, B, C are binary or not, i.e. whether all their entries belong to $\{0, 1\}$. The 4th column indicates whether $\mathbf{W}$ contains small or large entries. By small and large we mean that all entries are sampled randomly in the sets $\{0, \ldots, |\mathbf{W}| - 1\}$ or $\mathbb{F}$, respectively. As commitment scheme we use the KZG commitment scheme over the Pallas curve. As per Table 1 in [Hab22], the cost of committing to a witness $\mathbf{W}$ with large entries, or to the cross-term $\mathbf{T}$, is estimated as approximately $349m$ and $2^8 m$ field multiplications, for $m = 2^{16}$ and $m = 2^{20}$, respectively. When $\mathbf{W}$ contains small entries, we estimate the cost of committing to $\mathbf{W}$ as $15|\mathbf{W}|$ field multiplications. This is equivalent to the cost of 1 group multiplication per entry in $\mathbf{W}$ [zka, STW23b].

5.1 Experimental evaluation

In this section we evaluate the performance of our Mova implementation against the Nova and Hypernova implementations in the Sonobe [St] framework. For cryptographic operations, we used Pedersen commitments [Ped91] over the Pallas curve [Hop], and Poseidon [Arka] for cryptographic hashing.

We set the R1CS matrices A, B, C as the identity matrices[7]. We populated the vector $\mathbf{Z}_1$ with random numbers of size from 1 to 384 bits, and $\mathbf{Z}_2$ with random numbers of size from 1 to 20 bits. For Nova and Mova, we chose u to be a random field element. We computed $\mathbf{T}$ naively through the formula $\mathbf{T} = A \cdot \mathbf{Z}_2 \circ B \cdot \mathbf{Z}_1 + A \cdot \mathbf{Z}_1 \circ B \cdot \mathbf{Z}_2 - u_1 C \mathbf{Z}_2 - u_2 C \mathbf{Z}_1$. In

[7]We leave running benchmarks with more complex matrices as future work. See Remark 5.1 for a discussion on how using more complex A, B, C would affect the benchmarks

Section 5.2 we describe a more efficient approach which is roughly twice faster than the naive method. In the previous Tables 3 and 4 we used the optimised method.

The concluding numbers are the results of the average time of 100 runs measured on a 16 GB memory, 16-core Intel i7-11800H laptop CPU restricted to a single core. Table 5 presents the benchmarking results, not including the cost of committing to the R1CS witness vector $\mathbf{W}$. Within parentheses we indicate the costs of the main steps that are unique to each folding scheme. In Table 6 we display the costs of the main common operations in the three schemes.

Table 7 shows the costs of committing to a vector $\mathbf{V}$ of different sizes and with either small entries, or large entries. Finally, Table 8 displays the benchmark results for Mova, Nova, and Hypernova, including the costs of committing to the R1CS witness $\mathbf{W}$. We consider two cases, depending on whether $\mathbf{W}$ contains small or large entries.

The code used for the experiments is publicly available on GitHub[8].

m	Mova (pt-vs-line + MLE eval.)	Nova (Commit to $\mathbf{T}$)	Hypernova (Sumcheck)
2^{16}	36.0216 ms (10.4999 ms)	400.4406 ms (375.5840 ms)	150.0280 ms (106.7538 ms)
2^{20}	761.5190 ms (223.4527 ms)	5470.6681 ms (4913.5817 ms)	3195.2789 ms (1975.6348 ms)

Table 5: Comparison of Prover runtimes for a single fold across Nova, Mova, and Hypernova, not including the cost of commiting to the witness vector (cf. Table 8 for those costs). In all cells, the time without parentheses indicates the total Proving runtime. In Mova, the time within parentheses indicates the duration of the point-vs-line step (Protocol 6) plus the evaluation time of $\mathsf{mle}[\mathbf{T}]$ at a point $\mathbf{r}$. For Nova, it shows the time taken to commit to $\mathbf{T}$, and in Hypernova, it reflects the duration of the sumcheck. We remark that Hypernova's implementation does not leverage certain optimizations and thus its runtime does not reflect the concrete costs from Table 3 (cf. Remark 5.2)

m	Computing $\mathbf{T}$	Miscellaneous
2^{16}	20.7591 ms	4.0797 ms
2^{20}	416.5537 ms	84.1495 ms

Table 6: The two main operations that Nova and Mova Provers have in common (besides committing to $\mathbf{W}$), namely computing the cross-term $\mathbf{T}$, and performing miscellaneous operations such as computing the resulting folded witness and error vectors, and computing their commitments by using the homomorphicity of the commitment scheme.

Remark 5.1 (On the complexity of the R1CS matrices A, B, C)**.** The benchmarks were run taking the R1CS matrices A, B, C to be the identity matrix. The complexity of these matrices affects the cost of computing the cross-term $\mathbf{T}$, which is a common cost to Mova and Nova, and also affects Hypernova's costs (see Section 5.2). However, unlike in Nova and Hypernova, this computation represents the dominating cost of Mova's Prover work, and so

[8] https://github.com/NethermindEth/sonobe/tree/paper

m	Entries in $\{0, \dots, m-1\}$	Entries in $\mathbb{F}$
2^{16}	28.4877 ms	375.5840 ms
2^{20}	554.5014 ms	4913.5817 ms

Table 7: Time taken to commit a vector $\mathbf{V}$ with size $|\mathbf{V}| = m$ where $m = 2^{16}$ or $m = 2^{20}$, and so that either $\mathbf{V}$ contains random entries within the range $\{0, \dots, |\mathbf{V}| - 1\}$, or random entries in $\mathbb{F}$. Here we used the Pedersen commitment over the Pallas curve. The Hypernova library we used [St] does not make use of known optimisations such as [DT24].

m	$\mathbf{W}$ entries	Mova	Nova	Hypernova
2^{16}	Small	64.5093 ms	428.9283 ms	178.5157 ms
2^{16}	Large	411.6056 ms	776.0246 ms	525.6120 ms
2^{20}	Small	1316.0204 ms	6025.1695 ms	3749.7803 ms
2^{20}	Large	5675.1007 ms	10384.2498 ms	8108.8606 ms

Table 8: Benchmark of Mova, Nova and Hypernova's Prover runtimes including the cost of committing to the R1CS witness $\mathbf{W}$. We consider two scenarios, one where the entries in $\mathbf{W}$ are "small", and one where the entries are "large". As usual, by "small" we mean that the entries are randomly sampled in the range $\{0, \dots, |\mathbf{W}| - 1\}$, and by "large" that the entries are sampled in $\mathbb{F}$. The benchmarks follow the same configuration as those in Table 5.

using more complex matrices A, B, C will result in a less favorable comparison for Mova. Table 4 presents cost comparisons for matrices A, B, C of varying sparseness.

Remark 5.2 (On Hypernova's benchmark results)**.** As we discuss later in Section 5.2, when estimating the concrete costs of Hypernova's Prover in Table 3, we use [DT24] to bound the cost of the sumcheck protocol as $10m$. However, for our benchmarks, we used the less optimised framework [St]. This explains that the benchmarked Hypernova Prover is significantly more expensive than what Table 3 suggest.

5.2 Computing the concrete costs of Mova, Nova, and Hypernova

In this section we detail how the concrete costs in Table 3 were derived. We first look at the concrete field operation cost in Nova and Mova, then we analyse the concrete field operation cost of the point-vs-line reduction (Protocol 6), and finally we explain how we obtained the concrete field operation cost of the Hypernova protocol applied to R1CS.

Concrete costs of Mova and Nova In both Nova and Mova, the Prover has to compute the cross term

$$\mathbf{T} = A\mathbf{Z}_1 \circ B\mathbf{Z}_2 + A\mathbf{Z}_2 \circ B\mathbf{Z}_1 - u_1 C\mathbf{Z}_2 - u_2 C\mathbf{Z}_1$$

However, note the equality:

$$A(\mathbf{Z}_1 + \mathbf{Z}_2) \circ B(\mathbf{Z}_1 + \mathbf{Z}_2) - (u_1 + u_2)C(\mathbf{Z}_1 + \mathbf{Z}_2) = \mathbf{E}_1 + \mathbf{T} + \mathbf{E}_2$$

This means that the Prover can compute the expression on the left hand side of this equality, and then subtract the error terms. In general, for an $m \times m$ matrix M with non-zero rows, the multiplication of M with a vector in $\mathbb{F}^m$ costs $\sum_{i=1}^m (k_i^M - 1) = n - m$ field additions and $\sum_{i=1}^m k_i^M = n$ field multiplications, where for all $i \in [m]$, $k_i^M > 0$ is the number of non-zero entries of row i, and $n = \Omega(m)$ is the number of non-zero entries of M. This is because each entry in the matrix-vector product is the result of k_i^M multiplications, and $k_i^M - 1$ additions. To compute the cross-term $\mathbf{T}$, apart from the matrix-vector products, the Prover needs to perform the following operations: compute the vector $\mathbf{Z}_1 + \mathbf{Z}_2$; perform a Hadamard product between vectors of length m; perform a multiplication of a vector of length m by the constant $u_1 + u_2$; and perform three subtractions of a vector of length m. Therefore, the computation of the cross-term can be done in:

$$\begin{aligned} 4m + \sum_{i=1}^m ((k_i^A - 1) + (k_i^B - 1) + (k_i^C - 1)) = 3n + m, & \quad \text{field additions, and} \\ 2m + \sum_{i=1}^m (k_i^A + k_i^B + k_i^C) = 3n + 2m, & \quad \text{field multiplications,} \end{aligned} \tag{25}$$

where k_i^A is the number of non-zero entries in the i-th row of A (similarly for k_i^B, k_i^C). If all entries in A, B, C are either 0 or 1, then the cost of the matrix-vector products is reduced (there is no need for multiplications), and the total cost is reduced to

$$3n + m \text{ additions}, \quad 2m \text{ multiplications.} \tag{26}$$

Additionally, both Mova and Nova's Prover has to update the witness and error vectors in the last step of the protocol. This costs $3m$ field multiplications (required to compute $\alpha\mathbf{W}_2, \alpha\mathbf{T}, \alpha^2\mathbf{E}_2$) and $3m$ field additions. We do not count other negligible costs.

This suffices to count the cost of Nova's Prover and Verifier. The additional field operation work for Mova's Prover and Verifier consists of:

- The field operation work in the reduction $\mathbf{R}_{\mathsf{acc}} \times \mathbf{R}_{\mathsf{acc}} \to \mathbf{R}_{\mathsf{equal}}$. This is the point-vs-line reduction of Section 4.2. The costs are computed below.
- The evaluation of the MLE of the cross-term at a random vector of field elements. This can be done in at most $2m$ field additions and $3m$ field multiplications (see Lemma 3.8 in [Tha22]).

Concrete costs of the point-vs-line reduction of knowledge (Protocol 6) In the point-vs-line reduction (Section 4.2), the field operation costs are as follows:

- The Prover needs to compute the polynomials $h_1(X), h_2(X)$. In Section 6 we describe an algorithm that, in general, allows to compute each h_i with $4m - 2\log m - 4$ field multiplications and $5m - 2\log m - 5$ field additions.

  In the application where the Prover and Verifier fold an instance-witness pair from $\mathbf{R}_{\mathsf{R1CS}}$ with an instance-witness pair from $\mathbf{R}_{\mathsf{acc}}$, the Prover's costs for computing $h_1(X)$ are 0. This is because, if P is honest, then $h_1(X) = \mathsf{mle}[\mathbf{E}_1] \circ \ell(X) = 0$ because $\mathbf{E}_1 = \mathbf{0}$ is the zero vector. Hence, the total field multiplication cost of computing $h_1(X), h_2(X)$ in Mova (Protocol 8) is $4m - 2\log m - 4$.

  We note that if the Prover and Verifier were folding two instance-witness pairs from $\mathbf{R}_{\mathsf{acc}}$, then this cost would double, since then both $h_1(X), h_2(X)$ are nonzero polynomials.

- The Verifier needs to compute $h_1(0), h_2(1)$, the first of which can be read off as the constant coefficient of h_1, and the second of which is the sum of the coefficients of h_2. In total, this costs $\log(m) + 1$ field additions and no field multiplications.
- The Prover and Verifier need to compute the evaluations of h_1, h_2, ℓ at β. By using Horner's method, computing $h_1(\beta), h_2(\beta)$ costs at most $4(\log(m) + 1)$ field multiplications. Computing $\ell(\beta)$ takes at most $\log(m)$ field additions and $\log(m)$ field multiplications.

All in all, we obtain the concrete cost in Table 9 for the point-vs-line reduction.

	P	V	Rounds
Point-versus-line (Protocol 6)	$4m + 3\log(m)\ \mathbb{F}$	$7\log(m) + 5\ \mathbb{F}$	1

Table 9: Concrete cost of the point-versus-line method, in field multiplications for P, and field additions and multiplications for V. Rounds indicates the number of communication rounds in Protocol 6.

Concrete costs of Hypernova We next discuss the costs of Hypernova. We adopt the notation from [KS23b], and we refer to this reference for a full description of the Hypernova protocol.

Hypernova has the Prover perform a sumcheck on the function:

$$g(\mathbf{X}) := \left(\sum_{j \in [3]} \gamma^j \cdot L_j(\mathbf{X}) \right) + \gamma^4 \cdot Q(\mathbf{X})$$

where $\gamma \in \mathbb{F}$ is random, and the L_j, Q are defined as:

$$L_1(\mathbf{X}) := \widetilde{\mathsf{eq}}(\mathbf{r}_x, \mathbf{X}) \cdot \left(\sum_{\mathbf{y} \in \mathbb{B}^{\log(m)}} \widetilde{A}(\mathbf{X}, \mathbf{y}) \cdot \tilde{z}_1(\mathbf{y}) \right)$$

$$L_2(\mathbf{X}) := \widetilde{\mathsf{eq}}(\mathbf{r}_x, \mathbf{X}) \cdot \left(\sum_{\mathbf{y} \in \mathbb{B}^{\log(m)}} \widetilde{B}(\mathbf{X}, \mathbf{y}) \cdot \tilde{z}_1(\mathbf{y}) \right)$$

$$L_3(\mathbf{X}) := \widetilde{\mathsf{eq}}(\mathbf{r}_x, \mathbf{X}) \cdot \left(\sum_{\mathbf{y} \in \mathbb{B}^{\log(m)}} \widetilde{C}(\mathbf{X}, \mathbf{y}) \cdot \tilde{z}_1(\mathbf{y}) \right)$$

$$Q(\mathbf{X}) := \widetilde{\mathsf{eq}}(\beta, \mathbf{X}) \cdot \left(\left(\sum_{\mathbf{y} \in \mathbb{B}^{\log(m)}} \widetilde{A}(\mathbf{X}, \mathbf{y}) \cdot \tilde{z}_2(\mathbf{y}) \right) \cdot \left(\sum_{\mathbf{y} \in \mathbb{B}^{\log(m)}} \widetilde{B}(\mathbf{X}, \mathbf{y}) \cdot \tilde{z}_2(\mathbf{y}) \right) - \left(\sum_{\mathbf{y} \in \mathbb{B}^{\log(m)}} \widetilde{C}(\mathbf{X}, \mathbf{y}) \cdot \tilde{z}_2(\mathbf{y}) \right) \right)$$

Here $\widetilde{z}_1 = (\mathbb{w}_1, u, \mathbb{x}_1), \widetilde{z}_2 = (\mathbb{w}_2, 1, \mathbb{x}_2)$ are concatenatations of the witnesses for the linearised R1CS instance and the R1CS instance (respectively) with a field element (either u or 1), and the public inputs.

In [DT24], Dao and Thaler bound the Prover's number of field multiplications when performing the sumcheck on a polynomial of the form $Q(\mathbf{X})$. They bound this cost as $5m$. By applying their optimisations to the remaining term $\gamma \cdot L_1(\mathbf{X}) + \gamma^2 \cdot L_2(\mathbf{X}) + \gamma^3 \cdot L_3(\mathbf{X})$, the Prover can perform the sumcheck on this term in $4m + O(\sqrt{m})$ field multiplications, with the constant being such that the $O(\sqrt{m})$ term is well below m. All in all, we estimate that the Prover performs no more than $10m$ field multiplications in total for the sumcheck involving g.

The costs above assume the Prover knows all the evaluations of the sums in the definition of L_1, L_2, L_3, Q.We next explain how Prover can compute these.

Note that for any $\mathbf{x} \in \mathbb{B}^{\log(m)}$, the sum $\sum_{\mathbf{y} \in \mathbb{B}^{\log(m)}} \widetilde{A}(\mathbf{x}, \mathbf{y}) \cdot \tilde{z}_1(\mathbf{y})$ is equal to the $\mathbf{x}$-th coordinate (the coordinate whose index has binary representation $\mathbf{x}$) of the product $A \cdot z_1$, and similarly for B, C and z_2. Therefore, the Prover needs to compute $A \cdot z_1, B \cdot z_1, C \cdot z_1, A \cdot z_2, B \cdot z_2, C \cdot z_2$. We estimate the cost of these as follows: to compute $A \cdot z_1$, one needs

$$\sum_{i=1}^{m} (k_i^A - 1) = n - m$$

field additions, where k_i^A is the number of nonzero entries in the i-th row of A, and

$$\sum_{i=1}^{m} k_i^A = n$$

field multiplications. Recall $n = \Omega(m)$ is the number of nonzero entries in the R1CS matrices. The rest of matrix-vector multiplications can be computed similarly. Overall, the total field operations to compute the necessary matrix vector quantities is $6(n - m)$ field additions, and $6n$ field multiplications. If A, B, C are binary matrices, then there are no multiplications.

The Prover also needs the quantities:

$$\sigma_1 := \sum_{\mathbf{y} \in \mathbb{B}^{\log(m)}} \widetilde{A}(\mathbf{r}_{x'}, \mathbf{y}) \cdot \widetilde{z}_1(\mathbf{y})$$
$$\sigma_2 := \sum_{\mathbf{y} \in \mathbb{B}^{\log(m)}} \widetilde{B}(\mathbf{r}_{x'}, \mathbf{y}) \cdot \widetilde{z}_1(\mathbf{y})$$
$$\sigma_3 := \sum_{\mathbf{y} \in \mathbb{B}^{\log(m)}} \widetilde{C}(\mathbf{r}_{x'}, \mathbf{y}) \cdot \widetilde{z}_1(\mathbf{y})$$
$$\theta_1 := \sum_{\mathbf{y} \in \mathbb{B}^{\log(m)}} \widetilde{A}(\mathbf{r}_{x'}, \mathbf{y}) \cdot \widetilde{z}_2(\mathbf{y})$$
$$\theta_2 := \sum_{\mathbf{y} \in \mathbb{B}^{\log(m)}} \widetilde{B}(\mathbf{r}_{x'}, \mathbf{y}) \cdot \widetilde{z}_2(\mathbf{y})$$
$$\theta_3 := \sum_{\mathbf{y} \in \mathbb{B}^{\log(m)}} \widetilde{C}(\mathbf{r}_{x'}, \mathbf{y}) \cdot \widetilde{z}_2(\mathbf{y})$$

where $\mathbf{r}_{x'}$ is determined during the course of the sumcheck protocol. The Prover already has all σ_i's at the end of sumcheck. It also knows the quantity $\theta_1 \cdot \theta_2 - \theta_3$, so it may only compute two out of the three θ_i. Because at this point, the Prover already has the products $A \cdot z_2$ and so on, each of the two θ_i can be obtained by computing the evaluation of the MLE of the corresponding matrix vector product at the random point $\mathbf{r}_{x'}$. This can be separated into computing the evaluations of $\widetilde{\mathsf{eq}}(\mathbf{r}_{x'}, \mathbf{x})$ for all $\mathbf{x} \in \mathbb{B}^{\log(m)}$, and then performing two inner products with the dense representations of the corresponding matrix-vector products. The former costs no more than m field multiplications, and the latter costs $2m$ field multiplications in total. Finally there is the added cost of updating the witness, public inputs, and commitments among other things. Barring negligible costs, this amounts to m field multiplications and m field additions.

The Verifier needs to perform the sumcheck, which costs $c \log(m)$ field operations (counting both additions and multiplications) for some very small c. The Verifier also computes the quantity:

$$\sum_{j\in[3]} \gamma^j \cdot e_1 \cdot \sigma_j + \gamma^4 \cdot e_2 \cdot (\theta_1 \cdot \theta_2 - \theta_3)$$

where $e_1 := \widetilde{\mathsf{eq}}(\mathbf{r}_x, \mathbf{r}_{x'})$ and $e_2 := \widetilde{\mathsf{eq}}(\beta, \mathbf{r}_{x'})$. Computing e_1, e_2 costs at most $5\log(m)$ field operations, so this costs at most $5\log(m) + 13$ field operations. Then, there is also the cost to update the commitment, and the public inputs among other things. We do not make the constant hidden in the asymptotics explicit in Table 3, since we believe the Verifier cost is comparable to Mova.

6 An efficient algorithm for composing a multilinear polynomial with a line

In Section 4.2, we described a reduction of knowledge that uses the point-vs-line method as a subroutine. In the point-vs-line method, the Prover needs to compute univariate polynomials of the form $\mathsf{mle}[\mathbf{E}] \circ \ell$, where $\mathbf{E} \in \mathbb{F}^m$ and $\ell : \mathbb{F} \to \mathbb{F}^m$ is a line in $\mathbb{F}^m$ (a polynomial of the form $\mathbf{r}_0 + X(\mathbf{r}_1 - \mathbf{r}_0)$ for $\mathbf{r}_0, \mathbf{r}_1 \in \mathbb{F}^m$). If done naively, this could easily cost $O(m\mathsf{polylog}(m))$ field multiplications. In this section we provide a cost analysis of an algorithm that allows to compose a line and a $\log m$-variate MLE over a field $\mathbb{F}$ in $O(m)$ time and $O(m)$ space. We will be using the MLE point evaluation algorithm first proposed by Vu et al. in [VSBW13] (cf. [Tha22] for a detailed exposition) and implemented in the arkworks algebra library [Arkb]. Our observation is that the algorithm can be used for point values from any commutative ring $\mathbb{F} \subseteq \mathcal{R}$. This is because using the addition, subtraction and multiplication in $\mathcal{R}$ leaves the algorithm unchanged, while division is not used at all. Therefore, we can evaluate the MLE on the "point" $\mathbf{r}_0 + (\mathbf{r}_1 - \mathbf{r}_0)X \in \mathbb{F}[X]^{\log m}$ using the $O(m)$ evaluation algorithm. The algorithm is described in full in Algorithm 1. The result is similar to Claim 4.4 in [EG23].

We identify the boolean hypercube points $b \in \mathbb{B}^n$ with the binary integers they represent in little endian form, i.e. $(1, 1, 0, 1)$ is equivalent to $0b1011$ (cf. [Arkb]).

Algorithm 1 MLE-after-line composition

```
Input: MLE evaluations $[E_b]_{b \in \mathbb{B}^m}$ and line points $\mathbf{r}_0, \mathbf{r}_1 \in \mathbb{F}^{\log m}$.
Output: The coefficients of the univariate polynomial $\mathsf{mle}[\mathbf{E}](\mathbf{r}_0 + (\mathbf{r}_1 - \mathbf{r}_0)X)$.
 1: poly[0..(m - 1)] ← $[E_b]_{b \in \mathbb{B}^m}$; // Treat each $E_b \in \mathbb{F}$ as a constant polynomial $\in \mathbb{F}[X]$
 2: for $i = 1$ to $\log m$ do
 3:     $\ell_i \leftarrow \mathbf{r}_0[i-1] + (\mathbf{r}_1[i-1] - \mathbf{r}_0[i-1])X$; // compute the line coordinate polynomial
 4:     for $b = 0$ to $2^{\log m - i} - 1$ do
 5:         left ← poly[$2 \cdot b$];
 6:         right ← poly[$2 \cdot b + 1$];
 7:         poly[$b$] ← left + $\ell_i$ · (right − left);
 8:     end for
 9: end for
10: return poly[0];
```

A natural question is whether this algorithm is still linear in space and time. Now addition can cost up to $O(\log m)$ field operations and multiplication up to $O(\log^2 m)$ field operations, assuming univariate polynomial multiplication is done naively. Will it bring a (poly)logarithmic factor to the time complexity? In addition, one atomic storage now can contain up to $\log m$ field elements, will it also lead to increase in space complexity? Fortunately, the answer is no to both questions!

Lemma 6.1. *Algorithm 1 can be executed in at most $5m - 2\log m - 5$ field additions, $4m - 2\log m - 4$ field multiplications, and with a total memory consumption of at most $2m + 2\log m - 2$ field elements, given that the univariate polynomial multiplication algorithm is the naive quadratic one.*

Proof. Let us analyse the time complexity first. We will count field operations required to perform the algorithm.

Let us count how many multiplications μ each iteration of the loop 2-9 does and compute the degree d of the auxiliary polynomials stored in the array poly.

$\underline{i = 1}$: At the beginning, all the auxiliary polynomials are field elements. We do $2^{\log m - 1} = m/2$ (the loop 4-8) multiplications of field elements by a linear polynomial $\mathbf{r}_0[0] + (\mathbf{r}_1[0] - \mathbf{r}_0[0])X$ (line 7). That is $2 \cdot m/2 = 2 \cdot 1 \cdot m/2$ multiplications. The degrees of $m/2$ resulting auxiliary polynomials is 1 now. So

$$\mu_1 = 2 \cdot 1 \cdot m/2 \text{ and } d_1 = 1.$$

$\underline{i = 2}$: At the beginning, all the auxiliary polynomials are degree $d_1 = 1$. We do $2^{\log m - 2} = m/4$ multiplications of linear polynomial by a linear polynomial $\mathbf{r}_0[1] + (\mathbf{r}_1[1] - \mathbf{r}_0[1])X$. That is $2 \cdot 2 \cdot m/4$ multiplications. The degrees of $m/4$ resulting auxiliary polynomials is 2 now (linear by linear). So

$$\mu_2 = 2 \cdot 2 \cdot m/4 \text{ and } d_2 = 2.$$

$\underline{i = i}$: At the beginning, all the auxiliary polynomials are degree $d_{i-1} = i - 1$. We do $m/2^i$ multiplications of degree-$(i-1)$ polynomials by a linear polynomial $\mathbf{r}_0[i-1] + (\mathbf{r}_1[i-1] -$

$\mathbf{r}_0[i-1])X$. That is $2 \cdot i \cdot m/2^i$ multiplications. The degrees of $m/2^{i+1}$ resulting auxiliary polynomials is i now ($i-1$ by linear). So

$$\mu_i = 2 \cdot i \cdot m/2^i \text{ and } d_i = i.$$

Summing up $\mu_1, \ldots, \mu_{\log m}$ we get the expression for the number of multiplications

$$\sum_{i=1}^{\log m} \mu_i = \sum_{i=1}^{\log m} \frac{i \cdot m}{2^{i-1}} = m \sum_{i=1}^{\log m} \frac{i}{2^{i-1}}.$$

Using the method of differentiation of geometric progression we compute the sum

$$m \sum_{i=1}^{\log m} \frac{i}{2^{i-1}} = m \cdot \left(4 - \frac{2\log m + 4}{m}\right) = 4m - 2\log m - 4.$$

The number of additions on the ith iteration can be derived similarly. let us focus on the line 7 of the algorithm. We do $d_{i-1}+1$ additions in $\mathsf{right} - \mathsf{left}$. Then we need d_{i-1} additions for the multiplication by the linear polynomial ℓ_i. And finally to add to left we do $d_{i-1}+1$ additions as well. That sums up to $3d_{i-1}+2$. Knowing that $d_{i-1} = i-1$ we get $3i-1$ additions. And that happens $m/2^i$ times in the loop 4-8. This finally gives $m(3i-1)/2^i$. In addition to that, at each of the $\log m$ iterations of the loop 2-9 we compute the polynomial ℓ_i at line 3. The only field operation we do for that is $\mathbf{r}_1[i-1] - \mathbf{r}_0[i-1]$, which is an addition (we do not distinguish addition and subtraction). Thus we arrive to the following sum for the number of additions

$$\sum_{i=1}^{\log m} \left(\frac{m \cdot (3i-1)}{2^i} + 1\right).$$

Using the same summation techniques we obtain the number of additions:

$$5m - 2\log m - 5.$$

Summing the two formulas together the number of field operations is

$$9m - 4\log m - 9.$$

Regarding the memory, we have variables ℓ_i, left and right that are always there and that occupy in total $3\log m$. The array poly occupies by the end of the algorithm

$$\frac{m}{2} \sum_{i=1}^{\log m} \frac{i}{2^{i-1}} = 2m - \log m - 2$$

field memory cells. Hence the total memory consumption is

$$2m + 2\log m - 2.$$

$\square$

7 Deferred proofs

We end the paper with a proof that Protocol 6 is a reduction of knowledge.

7.1 Proof of Lemma 4.4

Public reducibility. We construct a deterministic function φ that, with input $(\mathsf{pp}, \mathbb{x}_1, \mathbb{x}_2, \tau)$, outputs $(\mathbb{x}_1', \mathbb{x}_2')$. Let $\tau = ((h_1(X), h_2(X)), \beta)$ (otherwise abort). Define $v_1' := h_1(\beta)$, $v_2' := h_2(\beta)$ and $\mathbf{r}' = (\mathbb{x}_2.\mathbf{r} - \mathbb{x}_1.\mathbf{r})\beta + \mathbb{x}_1.\mathbf{r}$ and notice that these values are polynomial-time computable from $\mathbb{x}_1, \mathbb{x}_2$ and τ. We have φ output $(\mathbb{x}_1', \mathbb{x}_2')$, where $\mathbb{x}_1' = (\mathbb{x}_1.\mathbf{x}, v_1', \mathbb{x}_1.u, \mathbb{x}_1.\overline{\mathbf{W}}, \mathbf{r}')$ and $\mathbb{x}_2' = (\mathbb{x}_2.\mathbf{x}, v_2', \mathbb{x}_2.u, \mathbb{x}_2.\overline{\mathbf{W}}, \mathbf{r}')$.

Let $\mathcal{A}, \mathsf{P}^*$ be PPT adversaries and let $(\mathbb{x}_1, \mathbb{x}_2, \mathsf{st}) \leftarrow \mathcal{A}(\mathsf{pp})$. Let τ be the transcript of the interaction between P^* and V, with input $(\mathsf{pp}, \mathbb{x}_1, \mathbb{x}_2, \mathsf{st})$ following Protocol 6. Let $(\mathbb{x}_1', \mathbb{x}_2'; \mathbb{w}_1', \mathbb{w}_2')$, be the output of the interaction. The output of $\varphi(\mathsf{pp}, \mathbb{x}_1, \mathbb{x}_2)$ is by construction equal to $(\mathbb{x}_1', \mathbb{x}_2')$, as required.

Perfect completeness. Let $\mathcal{A}$ be a PPT adversary and let $(\mathbb{x}_1; \mathbb{w}_1), (\mathbb{x}_1; \mathbb{w}_2) \leftarrow \mathcal{A}(\mathsf{pp})$ be such that $(\mathsf{pp}, \mathbb{x}_i; \mathbb{w}_i) \in \mathbf{R}_{\mathsf{acc}}$ for both $i = 1, 2$.

Write $(\mathsf{pp}, \mathbb{x}_i; \mathbb{w}_i) = (\mathsf{pp}, \mathbf{x}_i, v_i, u_i, \overline{\mathbf{W}}_i, \mathbf{r}_i; \mathbf{W}_i, \mathbf{E}_i, s_i) \in \mathbf{R}_{\mathsf{acc}}$, for $i = 1, 2$. Assume P and V honestly follow Protocol 6 with inputs $(\mathsf{pp}, \mathbb{x}_i; \mathbb{w}_i)$ and $(\mathsf{pp}, \mathbb{x}_i)$, $i = 1, 2$. We argue that the output of the interaction $\langle \mathsf{P}, \mathsf{V} \rangle$ is in $\mathbf{R}_{\mathsf{equal}}$ and V accepts. The case when $\mathbf{r}_1 = \mathbf{r}_2$ is immediate. Assume $\mathbf{r}_1 \neq \mathbf{r}_2$. By definition of $h_1(X), h_2(X)$ (the first message sent by P) and ℓ, we have

$$h_1(0) = \mathsf{mle}[\mathbf{E}_1] \circ \ell(0) = \mathsf{mle}[\mathbf{E}_1](\mathbf{r}_1) = v_1.$$

In exactly the same way, we obtain $h_2(1) = v_2$. Hence, V does not abort at the end of Protocol 6. Let $(\mathbb{x}_1', \mathbb{x}_2'; \mathbb{w}_1, \mathbb{w}_2)$ be the output of the interaction between P and V. Write $\mathbb{x}_i' = (\mathbb{x}_i, v_i', u_i', \overline{\mathbf{W}}_i, \mathbf{r}_i')$ and $\mathbb{w}_i' = (\mathbf{W}_i, \mathbf{E}_i, s_i)$, $i = 1, 2$. We now argue that $(\mathsf{pp}, \mathbb{x}_1', \mathbb{x}_2'; \mathbb{w}_1', \mathbb{w}_2') \in \mathbf{R}_{\mathsf{equal}}$. Indeed, by construction, $\mathbf{r}_1' = \mathbf{r}_2' = \mathbf{r}'$ for some point $\mathbf{r}'$. Hence, it suffices to check that $(\mathsf{pp}, \mathbb{x}_i'; \mathbb{w}_i') \in \mathbf{R}_{\mathsf{acc}}$ for both $i = 1, 2$. We show this for $(\mathsf{pp}, \mathbb{x}_1'; \mathbb{w}_1')$, the reasoning for $(\mathsf{pp}, \mathbb{x}_2'; \mathbb{w}_2')$ being analogous. Since $(\mathsf{pp}, \mathbb{x}_1; \mathbb{w}_1) \in \mathbf{R}_{\mathsf{acc}}$, we have $(A \cdot \mathbf{Z}_1) \circ (B \cdot \mathbf{Z}_1) = u_1 \cdot (C \cdot \mathbf{Z}_1) + \mathbf{E}_1$, and $\mathsf{Com}(\mathsf{pp}_{\mathsf{Com}}, \mathbf{W}_1, s_1) = \overline{\mathbf{W}}_1$. Hence, it only remains to check that $\mathsf{mle}[\mathbf{E}_1](\mathbf{r}') = v_1'$. However, it is clear that this is the case after inspecting Step 3 in Protocol 6. More precisely, we have $v_1' = h_1(\beta) = \mathsf{mle}[\mathbf{E}_1] \circ \ell(\beta) = \mathsf{mle}[\mathbf{E}_1](\mathbf{r}')$.

Knowledge soundness. The proof follows the same structure as the knowledge soundness proof in Lemma 4.1. Some of its arguments are a word-by-word repetiton of reasonings used in said proof.

Let $\mathcal{A}$ and P^* be expected polynomial-time adversaries. Let $(\mathbb{x}_1, \mathbb{x}_2, \mathsf{st}) \leftarrow \mathcal{A}(\mathsf{pp})$. Write $(\mathbb{x}_i, \mathbb{w}_i) = (\mathbf{x}_i, \overline{\mathbf{W}}_i, v_i, u_i, \mathbf{r}_i; \mathbf{W}_i, \mathbf{E}_i, s_i)$ for $i = 1, 2,$. Assume $\mathbf{r}_1 \neq \mathbf{r}_2$. Let ℓ be the linear function $\ell : \mathbb{F} \to \mathbb{F}^{\log m}$ satisfying $\ell(0) = \mathbf{r}_1$ and $\ell(1) = \mathbf{r}_2$.

Fix the notation $\langle \mathsf{P}^*, \mathsf{V} \rangle = \langle \mathsf{P}^*, \mathsf{V} \rangle(\mathsf{pp}, \mathbb{x}_1, \mathbb{x}_2, \mathsf{st})$, i.e. $\langle \mathsf{P}^*, \mathsf{V} \rangle$ is interactive protocol in which P^* and V interact following Protocol 3, with inputs $\mathsf{pp}, \mathbb{x}_1, \mathbb{x}_2$ and st. We also look at $\langle \mathsf{P}^*, \mathsf{V} \rangle$ as a random variable modelling the output of such interaction.

Let $\varepsilon_{\mathsf{total}}$ be the probability that P^*, with inputs $(\mathsf{pp}, \mathbb{x}_1, \mathbb{x}_2, \mathsf{st})$, succeeds in obtaining an output in $\mathbf{R}_{\mathsf{equal}}$, i.e. $\varepsilon_{\mathsf{total}} = \Pr[(\mathsf{pp}, \mathbb{x}, \langle \mathsf{P}^*, \mathsf{V} \rangle(\mathsf{pp}, \mathbb{x}_1, \mathbb{x}_2, \mathsf{st})) \in \mathbf{R}_{\mathsf{equal}}]$.

We define the extractor Ext as follows. Ext receives $(\mathsf{pp}, \mathbb{x}_1, \mathbb{x}_2, \mathsf{st})$ as inputs. Then:

1. Ext runs the protocol $\langle \mathsf{P}^*, \mathsf{V}\rangle$ once. Let $(\mathsf{pp}, \mathbb{x}_1^{(1)}, \mathbb{x}_2^{(1)}; \mathbb{w}_1^{(1)}, \mathbb{w}_2^{(1)})$ be the output of this interaction. If V rejects the interaction, or if $(\mathsf{pp}, \mathbb{x}_1^{(1)}, \mathbb{x}_2^{(1)}; \mathbb{w}_1^{(1)}, \mathbb{w}_2^{(1)}) \notin \mathbf{R}_{\mathsf{equal}}$, then Ext aborts.

   Otherwise, say we have

   $$(\mathsf{pp}, \mathbb{x}_i^{(1)}; \mathbb{w}_i^{(1)}) = (\mathsf{pp}, \mathbf{x}_i, v_i^{(1)}, u_i, \overline{\mathbf{W}}_i, \mathbf{r}^{(1)}; \mathbf{W}_i^{(1)}, \mathbf{E}_i^{(1)}, s_i^{(1)})$$

   for $i = 1, 2$, where $\mathbf{r}^{(1)} = \ell(\beta^{(1)})$ and $\beta^{(1)}$ is the Verifier's challenge. Let $h_1(X), h_2(X)$ be the Prover's first message of the interaction.

   Let $\mathcal{E}_{\mathsf{lines}}$ be the event that Ext does not abort in Step 1 and both $h_1(X) = \mathsf{mle}[E_1^{(1)}] \circ \ell(X)$ and $h_2(X) = \mathsf{mle}[E_2^{(1)}] \circ \ell(X)$. Let $\mathcal{E}_{\mathsf{bad-lines}}$ be the event that Ext does not abort in Step 1 but either $h_1(X) \neq \mathsf{mle}[E_1^{(1)}] \circ \ell(X)$ or $h_2(X) \neq \mathsf{mle}[E_2^{(1)}] \circ \ell(X)$.

   If $\mathcal{E}_{\mathsf{lines}}$ holds, then Ext terminates and outputs $\mathbb{w}_i^{(1)} = (\mathbf{W}_i^{(1)}, \mathbf{E}_i^{(1)}, s_i^{(1)})$ for $i = 1, 2$.

2. Next, Ext repeatedly runs $\langle \mathsf{P}^*, \mathsf{V}\rangle$, keeping always the same first message sent by P^* to be $(h_1(X), h_2(X))$. To do so, Ext rewinds P^* only to the point where P^* has already sent $(h_1(X), h_2(X))$.

   As soon as Ext obtains an output

   $$(\mathsf{pp}, \mathbb{x}_1^{(2)}, \mathbb{x}_2^{(2)}; \mathbb{w}_1^{(2)}, \mathbb{w}_2^{(2)}) \in \mathbf{R}_{\mathsf{equal}}$$

   and V does not reject, Ext terminates and outputs $\mathbb{w}_i^{(1)} = (\mathbf{W}_i^{(1)}, \mathbf{E}_i^{(1)}, s_i^{(1)})$ for $i = 1, 2$.

Let $\mathcal{E}_{h_1,h_2}$ be the event that P^*'s first message in Step 1 of Ext is $(h_1(X), h_2(X))$. Fix one such first message $(h_1(X), h_2(X))$, and let ε be the probability that $(\mathsf{pp}, \langle \mathsf{P}^*, \mathsf{V}\rangle(\mathsf{pp}, \mathbb{x}_1, \mathbb{x}_2, \mathsf{st})) \in \mathbf{R}_{\mathsf{acc}}$. We next prove that, conditioned on $\mathcal{E}_{h_1,h_2}$, Ext runs in expected polynomial time and outputs a valid witness for $(\mathbb{x}_1, \mathbb{x}_2)$ with probability $\varepsilon - \mathsf{negl}(\lambda)$. For readability purposes, we avoid referring to (h_1, h_2) in our notation. In what follows, unless stated otherwise, we consider all probabilities and events referring to Ext as conditioned on $\mathcal{E}_{h_1,h_2}$.

First of all we prove that Ext terminates in expected polynomial time. The argument is analogous to that in the proof of Lemma 4.1. Let $\mathcal{E}$ be the event that Ext does not abort in Step 1. We have $\mathsf{Pr}[\mathsf{Ext}] = \varepsilon$. Denote by Z the random variable (over Ext's random coins) representing the number of times Ext runs the interaction $\langle \mathsf{P}^*, \mathsf{V}\rangle$ in Step 2 (if Ext terminated or aborts in Step 1, then $Z = 0$). Let Z_{total} be the random variable (over Ext's random coins) representing the total number of times $\langle \mathsf{P}^*, \mathsf{V}\rangle$ is run when executing Ext. We have

$$\mathbb{E}[Z_{\mathsf{total}}] = 1 + \Pr[\mathcal{E}] \cdot \mathbb{E}[Z \mid \mathcal{E}] \leq 1 + \varepsilon \cdot \frac{1}{\varepsilon} = 2. \tag{27}$$

Hence, Ext runs in expected polynomial time, and it does not abort with probability at least ε.

Let $\mathcal{E}_{\mathsf{binding}}$ be the event that Ext, on inputs $(\mathsf{pp}_{\mathsf{Com}}, \mathbb{x}_1, \mathbb{x}_2, \mathsf{st})$ is not able to break the binding property of the commitment scheme Com, i.e. $\mathcal{E}_{\mathsf{binding}}$ is the event that at no point

Ext has computed two vectors $\mathbf{U}_1, \mathbf{U}_2$ and elements s_1, s_2 such that $\mathsf{Com}(\mathsf{pp}_{\mathsf{Com}}, \mathbf{U}_1, s_1) = \mathsf{Com}(\mathsf{pp}_{\mathsf{Com}}, \mathbf{U}_2, s_2)$. We have $\mathsf{Pr}[\mathcal{E}_{\mathsf{binding}}] = 1 - \mathsf{negl}(\lambda)$.

Now, conditioning on $\mathcal{E}$ occurring, since the events $\mathcal{E}_{\mathsf{lines}}$ and $\mathcal{E}_{\mathsf{bad-lines}}$ are complementary and mutually exclusive, by the law of total expectation,

$$\begin{aligned} &\mathbb{E}[Z \mid \mathcal{E}] = \mathbb{E}[Z \mid \mathcal{E}_{\mathsf{lines}} \text{ or } \mathcal{E}_{\mathsf{bad-lines}}] \\ =&\mathbb{E}[Z \mid \mathcal{E}_{\mathsf{lines}}]\mathsf{Pr}[\mathcal{E}_{\mathsf{lines}}] + \mathbb{E}[Z \mid \mathcal{E}_{\mathsf{bad-lines}}]\mathsf{Pr}[\mathcal{E}_{\mathsf{bad-lines}}] \\ =&\mathbb{E}[Z \mid \mathcal{E}_{\mathsf{bad-lines}}]\varepsilon_1 + \mathbb{E}[Z \mid \mathcal{E}_{\mathsf{lines}}](1 - \varepsilon_1), \end{aligned} \tag{28}$$

where $\varepsilon_1 = \mathsf{Pr}[\mathcal{E}_{\mathsf{bad-lines}}]$ is the probability that Ext does not abort in Step 1 and $\mathcal{E}_{\mathsf{lines}}$ does not hold. Note that $\mathbb{E}[Z \mid \mathcal{E}_{\mathsf{lines}}] = 0$, because if, at the end of Step 1, $\mathcal{E}_{\mathsf{lines}}$ holds, then the extractor terminates. We now make the following claim:

Claim 7.1. $\mathbb{E}[Z \mid \mathcal{E}_{\mathsf{bad-lines}}, \mathcal{E}_{\mathsf{binding}}]^{-1} = \mathsf{negl}(\lambda)$.

Assume Claim 7.1 is true for now. We now claim that if Ext does not abort in Step 1 and $\mathcal{E}_{\mathsf{lines}}$ holds, then both $(\mathsf{pp}, \mathbb{x}_1; \mathbb{w}_1^{(1)})$ and $(\mathsf{pp}, \mathbb{x}_2; \mathbb{w}_2^{(1)})$ belong to $\mathbf{R}_{\mathsf{acc}}$. Indeed, since Ext does not abort, we know that $(\mathsf{pp}, \mathbb{x}_1^{(1)}, \mathbb{x}_2^{(1)}; \mathbb{w}_1^{(1)}, \mathbb{w}_2^{(1)}) \in \mathbf{R}_{\mathsf{equal}}$. Further, $h_1(0) = v_1$ and $h_2(1) = v_2$, because V did not reject. Since $\mathcal{E}_{\mathsf{lines}}$ holds, this implies that

$$\begin{aligned} \mathsf{mle}[\mathbf{E}_1^{(1)}](\mathbf{r}_1) = \mathsf{mle}[\mathbf{E}_1^{(1)}] \circ \ell(0) = h_1(0) = v_1 \\ \mathsf{mle}[\mathbf{E}_2^{(1)}](\mathbf{r}_2) = \mathsf{mle}[\mathbf{E}_2^{(1)}] \circ \ell(1) = h_2(1) = v_2. \end{aligned} \tag{29}$$

Finally, since $(\mathsf{pp}, \mathbb{x}_i^{(1)}; \mathbb{w}_i^{(1)}) \in \mathbf{R}_{\mathsf{acc}}$ for both $i = 1, 2$, we conclude that $(\mathsf{pp}, \mathbb{x}_i; \mathbb{w}_i^{(1)}) \in \mathbf{R}_{\mathsf{acc}}$ as well, due to Eq. (29) and $\mathbb{x}_i, \mathbb{x}_i^{(1)}$ only differing in the evaluation points $\mathbf{r}_i, \mathbf{r}^{(1)}$, and evaluation values $v_i, v_i^{(1)}$.

We next argue that $\varepsilon_1 = \mathsf{negl}(\lambda)$. Note that this will complete the proof (barring the proof of Claim 7.1) that Ext runs in PPT time and outputs a valid witness with probability $\varepsilon - \mathsf{negl}(\lambda)$, conditioned on P^*'s first message being $(h_1(X), h_2(X))$, i.e. conditioned on the event $\mathcal{E}_{h_1,h_2}$. .

Further, we have already argued that Ext runs in expected polynomial time, and that it does not abort with probability at least ε. If, additionally, the probability that Ext does not abort and $\neg\mathcal{E}_{\mathsf{lines}}$ is negligible, then we conclude that Ext is a PPT algorithm that outputs a valid witness for $\mathbb{x}_1, \mathbb{x}_2$ with probability $\varepsilon - \mathsf{negl}(\lambda)$.

We now prove that $\varepsilon_1 = \mathsf{negl}(\lambda)$, assuming that Claim 7.1 is true. Indeed, plugging (28) into (27), and using both Claim 7.1 and $\mathbb{E}[Z \mid \mathcal{E}_{\mathsf{lines}}] = 0$, we obtain

$$2 \geq \mathbb{E}[Z_{\mathsf{total}}] = 1 + \varepsilon\mathbb{E}[Z \mid \mathcal{E}] = 1 + \varepsilon\varepsilon_1\mathbb{E}[Z \mid \mathcal{E}_{\mathsf{bad-lines}}] \tag{30}$$

Using again the law of total expectation,

$$\begin{aligned} &\mathbb{E}[Z \mid \mathcal{E}_{\mathsf{bad-lines}}] \\ =&\mathbb{E}[Z \mid \mathcal{E}_{\mathsf{bad-lines}}, \mathcal{E}_{\mathsf{binding}}]\mathsf{Pr}[\mathcal{E}_{\mathsf{binding}}] + \mathbb{E}[Z \mid \mathcal{E}_{\mathsf{bad-lines}}, \neg\mathcal{E}_{\mathsf{binding}}]\mathsf{Pr}[\neg\mathcal{E}_{\mathsf{binding}}] \\ \geq&\mathbb{E}[Z \mid \mathcal{E}_{\mathsf{bad-lines}}, \mathcal{E}_{\mathsf{binding}}]\mathsf{Pr}[\mathcal{E}_{\mathsf{binding}}]. \end{aligned} \tag{31}$$

Note that $\varepsilon_1 \leq \varepsilon$, since ε_1 is the probability that Ext does not abort at Step 1, which occurs with probability ε, and, additionally, $\mathcal{E}_{\mathsf{bad-lines}}$ holds. Hence from (30), (31), and Claim 7.1 we obtain

$$\varepsilon_1^2 \leq \varepsilon_1 \varepsilon \leq \mathbb{E}[Z \mid \mathcal{E}_{\mathsf{bad-lines}}, \mathcal{E}_{\mathsf{binding}}]^{-1} \mathsf{Pr}[\mathcal{E}_{\mathsf{binding}}]^{-1} = \mathsf{negl}(\lambda),$$

where the last equality follows from Remark 3.1 and the fact that $\mathbb{E}[Z \mid \mathcal{E}_{\mathsf{nonzero}}, \mathcal{E}_{\mathsf{binding}}]^{-1} = \mathsf{negl}(\lambda)$ and $\mathsf{Pr}[\mathcal{E}_{\mathsf{binding}}] = 1 - \mathsf{negl}(\lambda)$. This implies that $\varepsilon_1 = \mathsf{negl}(\lambda)$, as needed.

Now it only remains to prove Claim 7.1.

Proof of Claim 7.1. Assume $\mathcal{E}_{\mathsf{binding}}$ holds, i.e. Ext does not break the binding property of the commitment scheme Com. Assume we run Ext up to Step 1 and $\mathcal{E}_{\mathsf{bad-lines}}$ holds, so in particular Ext does not abort. Let $(\mathsf{pp}, \mathbb{x}_1^{(1)}, \mathbb{x}_2^{(1)}; \mathbb{w}_1^{(1)}, \mathbb{w}_2^{(1)})$ be the output of $\langle \mathsf{P}^*, \mathsf{V} \rangle$ obtained at the end of Step 1. By definition, this output belongs to $\mathbf{R}_{\mathsf{equal}}$.

By construction of Ext, during Step 2, Ext successively repeats an experiment Ξ, until Ξ is successful. The experiment Ξ consists in running $\langle \mathsf{P}^*, \mathsf{V} \rangle$, and Ξ is successful if the output $(\mathsf{pp}, \mathbb{x}_1^{(2)}, \mathbb{x}_2^{(2)}; \mathbb{w}_1^{(2)}, \mathbb{w}_2^{(2)})$ of $\langle \mathsf{P}^*, \mathsf{V} \rangle$ is in $\mathbf{R}_{\mathsf{equal}}$. Importantly, note that the random challenge $\beta^{(2)}$ sent by V during this experiment is uniformly random and independent of $(\mathsf{pp}, \mathbb{x}_1^{(1)}, \mathbb{x}_2^{(1)}; \mathbb{w}_1^{(1)}, \mathbb{w}_2^{(1)})$. Suppose one such output $(\mathsf{pp}, \mathbb{x}_1^{(2)}, \mathbb{x}_2^{(2)}; \mathbb{w}_1^{(2)}, \mathbb{w}_2^{(2)})$ is in $\mathbf{R}_{\mathsf{acc}}$. Write

$$(\mathsf{pp}, \mathbb{x}_i^{(2)}; \mathbb{w}_i^{(2)}) = (\mathsf{pp}, \mathbf{x}_i, v_i^{(2)}, u_i, \overline{\mathbf{W}}_i, \mathbf{r}^{(2)}; \mathbf{W}_i^{(2)}, \mathbf{E}_i^{(2)}, s_i^{(2)}), \quad i = 1, 2.$$

We next prove that $\mathbf{W}_i^{(1)} = \mathbf{W}_i^{(2)}$ and that $\mathbf{E}_i^{(1)} = \mathbf{E}_i^{(2)}$ for both $i = 1, 2$. Indeed, since in particular $(\mathsf{pp}, \mathbb{x}_i^{(1)}; \mathbb{w}_i^{(1)})$ and $(\mathsf{pp}, \mathbb{x}_i^{(2)}; \mathbb{w}_i^{(2)})$ are in $\mathbf{R}_{\mathsf{acc}}$, we have

$$\mathsf{Com}(\mathsf{pp}_{\mathsf{Com}}, \mathbf{W}_i^{(1)}, s_i^{(1)}) = \overline{\mathbf{W}}_i = \mathsf{Com}(\mathsf{pp}_{\mathsf{Com}}, \mathbf{W}_i^{(2)}, s_i^{(2)}) \quad \text{for } i = 1, 2. \tag{32}$$

Since we assume $\mathcal{E}_{\mathsf{binding}}$ holds, we obtain $\mathbf{W}_1^{(1)} = \mathbf{W}_1^{(2)}$ and $\mathbf{W}_2^{(1)} = \mathbf{W}_2^{(2)}$. Setting $\mathbf{Z}_i^{(1)} = (\mathbf{W}_i^{(1)}, \mathbf{x}_i, 1)$ and $\mathbf{Z}_i^{(2)} = (\mathbf{W}_i^{(2)}, \mathbf{x}_i, 1)$, it also follows that $\mathbf{Z}_i^{(1)} = \mathbf{Z}_i^{(2)}$ for both $i = 1, 2$. To show that $\mathbf{E}_i^{(1)} = \mathbf{E}_i^{(2)}$ for both $i = 1, 2$, we use that $(\mathsf{pp}, \mathbb{x}_1^{(1)}; \mathbb{w}_1^{(1)})$ and $(\mathsf{pp}, \mathbb{x}_1^{(2)}; \mathbb{w}_1^{(2)})$ are in $\mathbf{R}_{\mathsf{acc}}$, and that $\mathbf{Z}_1^{(1)} = \mathbf{Z}_1^{(2)}$:

$$\begin{aligned} \mathbf{E}_1^{(1)} &= (A \cdot \mathbf{Z}_1^{(1)}) \circ (B \cdot \mathbf{Z}_1^{(1)}) - u_1 \cdot (C \cdot \mathbf{Z}_1^{(1)}) \\ &= (A \cdot \mathbf{Z}_1^{(2)}) \circ (B \cdot \mathbf{Z}_1^{(2)}) - u_1 \cdot (C \cdot \mathbf{Z}_1^{(2)}) = \mathbf{E}_1^{(2)}. \end{aligned} \tag{33}$$

An analogous argument shows that $\mathbf{E}_2^{(1)} = \mathbf{E}_2^{(2)}$.

We now prove that $\beta^{(2)}$ is a root of $h_1(X) - \mathsf{mle}[\mathbf{E}_1^{(1)}] \circ \ell(X)$ and of $h_2(X) - \mathsf{mle}[\mathbf{E}_2^{(1)}] \circ \ell(X)$. Recall that the univariate polynomials $h_1(X), h_2(X)$ constitute P^*'s first message in all runs of $\langle \mathsf{P}^*, \mathsf{V} \rangle$. Since we assumed that

$$(\mathsf{pp}, \mathbb{x}_1^{(2)}, \mathbb{x}_2^{(2)}; \mathbb{w}_1^{(2)}, \mathbb{w}_2^{(2)}) \in \mathbf{R}_{\mathsf{equal}} \tag{34}$$

and V did not abort, we have $h_1(0) = v_1$ and $h_2(1) = v_2$, and $v_1^{(2)} = h_1(\beta^{(2)})$, $v_2^{(2)} = h_2(\beta^{(2)})$. Further, due to (34), $v_i^{(2)} = \mathsf{mle}[\mathbf{E}_i^{(2)}](\mathbf{r}^{(2)})$ for $i = 1, 2$. Hence, using that $\mathbf{E}_1^{(1)} = \mathbf{E}_1^{(2)}$, and $\mathbf{E}_2^{(1)} = \mathbf{E}_2^{(2)}$ we have that that

$$\begin{aligned} h_i(\beta^{(2)}) &= v_i^{(2)} = \mathsf{mle}[\mathbf{E}_i^{(2)}](\mathbf{r}^{(2)}) = \mathsf{mle}[\mathbf{E}_i^{(1)}](\mathbf{r}^{(2)}) \\ &= \mathsf{mle}[\mathbf{E}_i^{(1)}](\ell(\beta^{(2)})) = (\mathsf{mle}[\mathbf{E}_i^{(1)}] \circ \ell)(\beta^{(2)}) \end{aligned} \tag{35}$$

for $i = 1, 2$, as claimed. Hence, for Ξ to be successful, it is necessary that $\beta^{(2)}$ is a root of both degree $\leq \log(m)$ univariate polynomials $h_1(X) - \mathsf{mle}[\mathbf{E}_1^{(1)}] \circ \ell(X)$ and $h_2(X) - \mathsf{mle}[\mathbf{E}_2^{(1)}] \circ \ell(X)$. Since we assume $\mathcal{E}_{\mathsf{bad-lines}}$ holds, we have that at least one of these polynomials is not the zero polynomial. Then, by Schwartz-Zippel lemma, and because $\beta^{(2)}$ is sampled uniformly at random after $\mathbf{E}^{(1)}$, $\ell(X)$, and $h_1(X), h_2(X)$ are determined, the probability that Ξ is successful is at most $\log m/|\mathbb{F}| = \mathsf{negl}(\lambda)$. $\square$

Recall we have only proved so far that, conditioned on P^*'s first message in Step 1 of Ext being $(h_1(X), h_2(X))$, we have that Ext runs in expected polynomial time and outputs a valid witness for $\mathbb{x}$ with probability $\varepsilon - \mathsf{negl}(\lambda)$, where ε is the probability that $(\mathsf{pp}, \langle \mathsf{P}^*, \mathsf{V} \rangle) \in \mathbf{R}_{\mathsf{R1CS}}$, conditioned on P^*'s first message being $(h_1(X), h_2(X))$.

We have to prove that, without conditioning on any P^*'s first message, Ext is a PPT algorithm that outputs a valid witness with probability $\varepsilon_{\mathsf{total}} - \mathsf{negl}(\lambda)$. At this point, this follows word by word as the last part of the proof of Lemma 4.1.

Finally, recall we assumed that the evaluation points $\mathbf{r}_1, \mathbf{r}_2$ were different. If $\mathbf{r}_1 = \mathbf{r}_2$, then it is trivial to build a PPT extractor that outputs a valid witness with probability $\varepsilon_{\mathsf{total}}$. $\square$

8 Acknowledgements

We thank Ariel Gabizon, Matteo Pintonello, Srinath Setty, Lev Soukhanov, and Justin Thaler for very helpful discussions, and Togzhan Barakbayeva and Matthew Klein for invaluable help in preparing the experimental evaluations of Mova.

This work was supported by the Ethereum Foundation ZK grant FY24-1491.

9 References

[Arka] Arkworks_contributors. Ark-crypto-primitives. URL: https://github.com/arkworks-rs/crypto-primitives.

[Arkb] Arkworks_contributors. Arkworks zkSNARK ecosystem/algebra. URL: https://github.com/arkworks-rs/algebra/tree/dcf73a5f9610ba9d16a3c8e0de0b3835e5e5d5e4.

[Awe] Awesome_folding_contributors. Awesome-folding. URL: https://github.com/lurk-lab/awesome-folding?tab=readme-ov-file#other-resources-podcasts-etc.

[BC23] Benedikt Bünz and Binyi Chen. Protostar: Generic efficient accumulation/folding for special sound protocols. Cryptology ePrint Archive, Paper 2023/620, 2023. https://eprint.iacr.org/2023/620. URL: https://eprint.iacr.org/2023/620.

[BC24a] Dan Boneh and Binyi Chen. Latticefold: A lattice-based folding scheme and its applications to succinct proof systems. Cryptology ePrint Archive, Paper

2024/257, 2024. https://eprint.iacr.org/2024/257. URL: https://eprint.iacr.org/2024/257.

[BC24b] Benedikt Bünz and Jessica Chen. Proofs for deep thought: Accumulation for large memories and deterministic computations. Cryptology ePrint Archive, Paper 2024/325, 2024. URL: https://eprint.iacr.org/2024/325.

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

[BCMS20] Benedikt Bünz, Alessandro Chiesa, Pratyush Mishra, and Nicholas Spooner. Proof-carrying data from accumulation schemes. Cryptology ePrint Archive, Paper 2020/499, 2020. URL: https://eprint.iacr.org/2020/499.

[BCS21] Jonathan Bootle, Alessandro Chiesa, and Katerina Sotiraki. Sumcheck arguments and their applications. In *Advances in Cryptology–CRYPTO 2021: 41st Annual International Cryptology Conference, CRYPTO 2021, Virtual Event, August 16–20, 2021, Proceedings, Part I 41*, pages 742–773. Springer, 2021.

[BGH19] Sean Bowe, Jack Grigg, and Daira Hopwood. Recursive proof composition without a trusted setup. Cryptology ePrint Archive, Paper 2019/1021, 2019. URL: https://eprint.iacr.org/2019/1021.

[CT10] Alessandro Chiesa and Eran Tromer. Proof-carrying data and hearsay arguments from signature cards. In *ICS*, pages 310–331. Tsinghua University Press, 2010.

[DT24] Quang Dao and Justin Thaler. More optimizations to sum-check proving. Cryptology ePrint Archive, Paper 2024/1210, 2024. https://eprint.iacr.org/2024/1210. URL: https://eprint.iacr.org/2024/1210.

[EG23] Liam Eagen and Ariel Gabizon. Protogalaxy: Efficient protostar-style folding of multiple instances. Cryptology ePrint Archive, Paper 2023/1106, 2023. https://eprint.iacr.org/2023/1106. URL: https://eprint.iacr.org/2023/1106.

[Hab22] Ulrich Haböck. Multivariate lookups based on logarithmic derivatives. Cryptology ePrint Archive, Paper 2022/1530, 2022. https://eprint.iacr.org/2022/1530. URL: https://eprint.iacr.org/2022/1530.

[Hop] Daira Hopwood. Crate ark_pallas. URL: https://docs.rs/ark-pallas/latest/ark_pallas/.

[KP23] Abhiram Kothapalli and Bryan Parno. Algebraic reductions of knowledge. In *Annual International Cryptology Conference*, pages 669–701. Springer, 2023.

[KS23a] Abhiram Kothapalli and Srinath Setty. Cyclefold: Folding-scheme-based recursive arguments over a cycle of elliptic curves. *Cryptology ePrint Archive*, 2023.

[KS23b] Abhiram Kothapalli and Srinath Setty. Hypernova: Recursive arguments for customizable constraint systems. Cryptology ePrint Archive, Paper 2023/573, 2023. https://eprint.iacr.org/2023/573. URL: https://eprint.iacr.org/2023/573.

[KST21] Abhiram Kothapalli, Srinath Setty, and Ioanna Tzialla. Nova: Recursive zero-knowledge arguments from folding schemes. Cryptology ePrint Archive, Paper 2021/370, 2021. https://eprint.iacr.org/2021/370. URL: https://eprint.iacr.org/2021/370.

[NDC$^+$24] Wilson Nguyen, Trisha Datta, Binyi Chen, Nirvan Tyagi, and Dan Boneh. Mangrove: A scalable framework for folding-based snarks. *Cryptology ePrint Archive*, 2024.

[Ped91] Torben Pryds Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *Annual international cryptology conference*, pages 129–140. Springer, 1991.

[Sou24] Lev Soukhanov. Warpfold: Wrongfield arithmetic for protostar folding. *Cryptology ePrint Archive*, 2024.

[St] Sonobe-team. Sonobe. URL: https://github.com/privacy-scaling-explorations/sonobe.

[STW23a] Srinath Setty, Justin Thaler, and Riad Wahby. Customizable constraint systems for succinct arguments. *Cryptology ePrint Archive*, 2023.

[STW23b] Srinath Setty, Justin Thaler, and Riad Wahby. Unlocking the lookup singularity with lasso. Cryptology ePrint Archive, Paper 2023/1216, 2023. https://eprint.iacr.org/2023/1216. URL: https://eprint.iacr.org/2023/1216.

[Tha22] Justin Thaler. Proofs, arguments, and zero-knowledge. *Foundations and Trends® in Privacy and Security*, 4(2–4):117–660, 2022.

[Val08] Paul Valiant. Incrementally verifiable computation or proofs of knowledge imply time/space efficiency. pages 1–18, 03 2008. doi:10.1007/978-3-540-78524-8_1.

[VSBW13] Victor Vu, Srinath Setty, Andrew J. Blumberg, and Michael Walfish. A hybrid architecture for interactive verifiable computation. In *2013 IEEE Symposium on Security and Privacy*, pages 223–237, 2013. doi:10.1109/SP.2013.48.

[ZGGX23] Tianyu Zheng, Shang Gao, Yu Guo, and Bin Xiao. Kilonova: Non-uniform pcd with zero-knowledge property from generic folding schemes. *Cryptology ePrint Archive*, 2023.

[zka] zkalc. zkalc is a cryptographic calculator! URL: https://zka.lc/.

[ZZD23] Zibo Zhou, Zongyang Zhang, and Jin Dong. Proof-carrying data from multi-folding schemes. *Cryptology ePrint Archive*, 2023.