Publicly Verifiable Non-Interactive Arguments for Delegating Computation

Omer Paneth[*] Guy N. Rothblum[†]

December 4, 2014

Abstract

We construct publicly verifiable non-interactive arguments that can be used to delegate polynomial time computations. These computationally sound proof systems are completely non-interactive in the common reference string model. The verifier's running time is nearly-linear in the input length, and poly-logarithmic in the complexity of the delegated computation. Our protocol is based on graded encoding schemes, introduced by Garg, Gentry and Halevi (Eurocrypt 2012). Security is proved under a falsifiable and arguably simple cryptographic assumption about graded encodings. All prior publicly verifiable non-interactive argument systems were based on non-falsifiable knowledge assumptions. Our new result builds on the beautiful recent work of Kalai, Raz and Rothblum (STOC 2014), who constructed *privately verifiable* 2-message arguments. While building on their techniques, our protocols avoid no-signaling PCPs, and we obtain a simplified and modular analysis.

As a second contribution, we also construct a publicly verifiable non-interactive argument for (logspace-uniform) computations of bounded depth. The verifier's complexity grows with the depth of the circuit. This second protocol is *adaptively* sound, and its security is based on a falsifiable assumption about the hardness of a search problem on graded encodings (a milder cryptographic assumption). This result builds on the interactive proof of Goldwasser, Kalai and Rothblum (STOC 2008), using graded encodings to construct a non-interactive version of their protocol.

[*]Boston University. Email: omer@bu.edu. Supported by the Simons award for graduate students in theoretical computer science and an NSF Algorithmic foundations grant 1218461.

[†]Stanford University. Email: rothblum@alum.mit.edu. Part of this work was done at Microsoft Research, Silicon Valley. This work was supported by the DARPA PROCEED program. Opinions, findings and conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect the views of DARPA.

Contents

1 Introduction

The power of efficiently verifiable proof systems is a foundational issue in the study of computation. A central goal is constructing proof systems that can be used by a powerful prover to convince a weak verifier of the correctness of a complex computational statement. Beyond its foundational importance in the theory computation, this question has real-world applications, such as *delegating computation*. In this setting, a powerful server (playing the role of the prover) can run a complex computation for a much weaker client (playing the role of the verifier), and provide a proof of the output's correctness. For such a proof to be applicable to the delegation scenario, we require: (i) super-efficient verifier: in particular, verification should not require re-excuting the computation, and (ii) poly-time proof: generating the proof should not require much more computational resources than just running the computation.

A similar question was raised by Babai, Lund, Fortnow and Szegedy [BFLS91] in the PCP setting. Kilian [Kil92] and Micali [Mic94] gave the first candidate scheme for delegating computation. The question re-emerged in the theoretical literature in the work of Goldwasser, Kalai and Rothblum [GKR08], and has since become become the focus of a rich body of research spanning both theory and systems. See the recent survey by Walfish and Blumberg [WB13].

We focus on proof systems for proving the membership of an input x in a language $\mathcal{L}$. The "holy grail" for delegating computations is *fully non-interactive proofs*, comprised of a single message sent from the prover to the verifier, as in classic NP proofs. Unfortunately, there are serious barriers to constructing such proofs for delegating general deterministic computations (in particular, even without the poly-time proof property, they imply general non-deterministic speedups for deterministic computations). Instead, we focus on computationally sound proofs in the common reference string model, a prevalent model in the cryptographic literature:

1. We require only *computational soundness*. Soundness is only required to hold against *efficient* cheating provers. Computationally sound proof systems are commonly called *argument systems*.

2. We use a (public) *common reference string* (CRS), generated in advance by a trusted authority (or by the verifier herself). This CRS can be used (repeatedly) by different parties to verify proofs.

Under these relaxations, we maintain the requirements for super-efficient verification (including the time to generate the CRS) and poly-time proofs. Both the prover and the verifier have access to the CRS, but neither has access to the secret coins used to generate the CRS. We refer to such a system as a *(publicly verifiable) non-interactive argument (for delegating computation)*. For the remainder of this work, we use the term *non-interactive argument* as shorthand. (We note that past work has occasionally referred to 2-message protocols with private-key verification as non-interactive arguments. Unless explicitly stated otherwise, all non-interactive arguments in this work are publicly verifiable.) Such non-interactive argument are especially attractive for delegating computation, as any untrusted server can simply use the CRS to generate proofs and send them off (non-interactively and asynchronously), to be verified at the clients' convenience. The fundamental question that motivates our work is:

Do there exist publicly verifiable *non-interactive arguments for delegating computations?*

Past work, starting with [Mic94], constructed publicly-verifiable non-interactive arguments under *non-falsifiable* assumptions (and even for non-deterministic computations).[1] In a beautiful recent work, Kalai, Raz and Rothblum [KRR14] constructed a *privately verifiable* 2-message argument for delegating polynomial-time computations, assuming a sub-exponentially secure PIR scheme. Private verifiability means that only the verifier, who generated the challenge message using secret coins, can verify that a proof is valid. Our focus, on the other hand, is on *public verifiability*.

[1]A "falsifiable" assumption [Nao03] is one that can be efficiently refuted. Falsifiability is a basic "litmus test" for cryptographic assumptions.

This work. We construct non-interactive arguments for delegating computations based on *graded encoding schemes* (see Garg, Gentry and Halevi [GGH13]). Security is proved under *simple and falsifiable polynomial-time hardness assumptions* about graded encodings. We obtain:

- A (publicly verifiable) non-interactive argument that can be used to delegate *any deterministic polynomial-time computation.*
- A (publicly verifiable) non-interactive argument for delegating *bounded-depth computations*. While this second protocol doesn't support as rich a family of computations as the first protocol, we prove its security under a milder hardness assumption. Moreover, it can be shown to be *adaptively secure*: security holds even against an adversary who chooses the input x as a function of the CRS. To the best of our knowledge, this is the first construction of a non-interactive delegation scheme that has adaptive soundness under a falsifiable polynomial-time hardness assumption.

While our main focus is on the public verifiability property, we also note that no prior constructions of non-interactive arguments (publicly verifiable or privately verifiable) were known to be secure under polynomial-time hardness assumptions, even with non-adaptive soundness.

We elaborate briefly on graded encodings (GE), which provide cryptographic hard-to-invert encodings for the elements of a certain ring. The hard-to-invert (probabilistic) encodings support homomorphic evaluation of low-degree algebraic expressions over the encoded elements. The GE also allows *zero-tests*: testing whether a low-degree expression evaluates to zero. Evaluating expressions whose degree exceeds a set threshold is assumed to be hard.

At a (very) high level, both of our constructions rely heavily on techniques from the interactive proof and PCP literature. These techniques are used to obtain a low-degree (non-cryptographic) *algebraic encoding* of the delegated computation. Using the Graded Encoding Scheme, the CRS specifies hard-to-invert *cryptographic encryptions* of queries into the algebraically encoded computation. On one hand, the honest prover can (homomorphically) compute encrypted answers to the encrypted queries because the algebraic encoding of the computation is low degree, and the GE supports low-degree operations on the encrypted queries. On the other hand, the GE's security prevents a cheating prover from learning information about the queries and breaking soundness. The public verification procedure runs a low-degree procedure on the encrypted queries from the CRS and on the proof. The check returns 0 when it succeeds, and the verifier can test for this using the GE zero-test procedure.

Our first constructions is greatly inspired by the recent work of [KRR14], who achieved *privately verifiable* non-interactive arguments (albeit using only PIR schemes, rather than GEs). We remark that our analyses are significantly simpler than [KRR14]. We also attempt to abstract the key ideas inspired by their proof, and provide a modular presentation.

Organization. We proceed with informal statements and a discussion of our main results in Sections 1.1 and 1.2. We give an overview of graded encodings and our cryptographic assumptions in Section 1.3. Further related work is discussed in Section 1.4. We provide a technical overview of some key ideas and techniques in Section 2. The remainder of the paper gives the results in full detail.

1.1 Non-Interactive Arguments for P

Our main result is a non-interactive argument for delegating polynomial-time computations.

Informal Theorem 1.1 (See Theorem 4.5). *Assume the existence of a Graded Encoding Scheme satisfying Assumption 1.4. Let $\mathcal{L}$ be a language that is computable by a Turing Machine in time $t(n)$. Let κ be a security parameter and $n = \mathrm{poly}(\kappa)$ an input length. There exists a (publicly verifiable) non-interactive argument for $\mathcal{L}$ where:*

1. *The prover runs in time* $\mathrm{poly}(t(n), \kappa)$.

2. *The verifier runs in time* $n \cdot \mathrm{poly}(\log(t(n)), \kappa)$.

3. *The time to generate the CRS, as well as the lengths of the CRS and the proof, are* $\mathrm{poly}(\log(t(n)), \kappa)$.

We elaborate on Assumption 1.4 in Section 1.3.1. We emphasize that we only assume polynomial time hardness. Soundness is *non-adaptive*: the input x is specified independently of the CRS. As is usually the case for argument systems, we can use complexity leveraging to make the protocol adaptively secure, but this requires assuming sub-exponential time hardness.

Background. One can view our construction as inspired by a tantalizing idea of Aiello *et al* [ABOR00] for constructing *private-key* non-interactive arguments using a PIR scheme or Fully Homomorphic Encryption (FHE). Their idea was to have the verifier encrypt queries to a PCP using the PIR scheme or FHE. The honest prover could answer according to the honest PCP proof and convince the verifier. The hope was that since a cheating prover couldn't tailor its answer to one query depending on other queries's values, the non-interactive argument would inherit the PCP's soundness. Dwork *et al.* [DLN+04] showed that this intuition for security was quite problematic (see below). Nonetheless, a beautiful series of works [KR09, KRR13, KRR14] has taken inspiration from the [ABOR00] idea to great effect.

Semantically secure queries and public verifiability.

convince the verifier on a *simulated* CRS, generated to encrypt any arbitrary collection of queries. This is because the encryption is semantically secure, and the prover's success is publicly observable even without knowing the secret key. We leverage this powerful guarantee in the security proof. It greatly simplifies our analysis (compared with [KRR14]), and in particular it allows us to avoid no-signaling PCPs. In fact, we avoid even the construction of standard PCPs and do not use low-degree tests. We do note, however, that our construction and security proof draw extensively on those of [KRR14]. Most significantly, their "augmented circuit transformation" plays a key role in our proof. As an additional contribution, our security proof abstracts some of the main ideas and steps in the [KRR14] construction, and is (arguably) simpler and more modular.

See Section 2.1 for a taste of the techniques, and Section 4 for the full details.

1.2 Non-Interactive Arguments for Bounded-Depth Computations

Our second contribution is a protocol for bounded-depth computations. We follow [GKR08] and focus our attention on languages that are computable by log-space uniform circuits of bounded depth. For this more restricted (but still very rich) family of computations, we obtain a protocol with improved security. First, we base security on the hardness of a search problem on GEs, rather than on a decisional assumption. Second, the resulting scheme is *adaptively sound*: even a cheating prover who first sees the CRS, and picks the input x adaptively as a function of the CRS, cannot make the verifier accept when $x \notin \mathcal{L}$.

Informal Theorem 1.2 (See Theorem 5.13)**.** *Assume the existence of a Graded Encoding Scheme where Assumption 1.6 holds. Let $\mathcal{L}$ be a language that is computable by a* $\log$*-space uniform circuit ensemble of depth $D(n)$ and size $S(n)$. Let κ be the security parameter and $n = \mathrm{poly}(\kappa)$ an input length. There exists a (publicly verifiable) non-interactive argument for $\mathcal{L}$ with adaptive soundness, where:*

1. *The prover runs in time* $\mathrm{poly}(S(n), \kappa)$.
2. *The verifier runs in time* $n \cdot \mathrm{poly}(D(n), \kappa)$.
3. *The time to generate the CRS , as well as the lengths of the CRS and the proof, are* $\mathrm{poly}(D(n), \kappa)$.

We elaborate on Assumption 1.6 in Section 1.3.2. Our starting point is the (information theoretically secure) interactive proof of [GKR08]. At a high level, the CRS contains hard-to-invert cryptographic encodings of the GKR verifier's interactive queries. If these queries were sent to a cheating prover in the clear and in advance, it would be easy to find convincing answers that make the GKR verifier accept. We show, however, that any cheating prover who can make the GKR verifier accept using only these *cryptographic encodings* of the queries, must be breaking the security of the GE.

We emphasize that, in this second protocol, the cryptographic encodings *are not semantically secure*. In particular, we cannot argue (as above) that the prover cannot distinguish the *real* CRS from a *simulated* CRS tailored to include specific queries. Instead, we make extensive use of the GKR protocol's algebraic structure to show that finding a cheating proof necessitates solving a hard search problem on GEs. This allows us to achieve *adaptive* security based on a milder hardness assumption.

1.3 Graded Encoding Schemes and Security Assumptions

Graded (multi-linear) encoding schemes were introduced by Garg, Gentry and Halevi [GGH13]. By now there are several candidate constructions [GGH13, CLT13, GGH14]. They come in symmetric and asymmetric variants, and we use both types. The delegation protocol for polynomial-time computations uses symmetric GEs, and the protocol for bounded-depth computation uses asymmetric GEs. We give a brief overview on GEs and give informal statements of our cryptographic assumptions. See Section 3.1 for details on GEs and our assumptions.

A brief overview. A GE is specified by a degree parameter δ and public parameters pp, which define an underlying ring R (for us R is always $\mathbb{Z}_p$ for some prime p exponential in the security parameter). The GE includes a collection Λ of *levels* for encodings. For symmetric GEs, the levels are integers and $\Lambda = [0, \delta]$. For asymmetric GEs, there is an additional *dimension* parameter κ, and the levels are vectors of integers in $\Lambda = [0, \delta]^\kappa$. In both cases, any ring element $\alpha \in R$ can be encoded in any level $\mathbf{v} \in \Lambda$. Such an encoding is denoted by $[\alpha]_{\mathbf{v}}$ (there are many encodings for each ring elements in each level).

Using GEs we can perform: (i) Additions and subtractions on encodings in the same level, where $[\alpha]_{\mathbf{v}} \pm [\beta]_{\mathbf{v}} = [\alpha \pm \beta]_{\mathbf{v}}$, (ii) a zero test, taking an input encoding $[\alpha]_{\mathbf{v}}$ in any level in Λ, and outputting whether $\alpha = 0$, (iii) multiplications, where $[\alpha]_{\mathbf{v}} \cdot [\beta]_{\mathbf{u}} = [\alpha \cdot \beta]_{\mathbf{v}+\mathbf{u}}$. In the asymmetric case, the summation of levels is performed pointwise. If $(\mathbf{v} + \mathbf{u}) \notin \Lambda$, then multiplication fails, (iv) sampling the encoding of a uniformly random ring element encoded in different (and non-zero) levels of Λ.[2]

1.3.1 Curve Encryption Assumption over Symmetric GEs

Our non-interactive arguments for P use symmetric GEs, and a "curve encryption" primitive. An *m-dimensional curve of degree δ* over the ring R is a function $\gamma : R \to R^m$, which can be written as $\gamma(t) = \sum_{i\in[0,\delta]} \vec{\alpha_i} \cdot t^i$, where for each i we have $\vec{\alpha_i} = (\alpha_{i,1}, \ldots, \alpha_{i,m}) \in R^m$. We also consider *encoded curves*, where the items in each $\vec{\alpha}$ are encoded:

$$[\gamma]_\ell = \left\{ [\alpha_{i,j}]_\ell \right\}_{i\in[0,\delta], j\in[m]} .$$

The security requirement is that a level-1 encoding of a curve that passes through a given point $\mathbf{w} \in \{0,1\}^m$, but is otherwise a random curve, completely hides $\mathbf{w}$. We assume this for curves of the maximal degree δ that the GE supports.

Definition 1.3 (Curve Encryption)**.** *A curve encryption scheme is given by a procedure* Curve *such that:*

- *Functionality. For $\mathbf{w} \in \{0,1\}^m$, taking $([\gamma]_1, [t]_0) \leftarrow \mathsf{Curve}(\mathsf{pp}, \mathbf{w})$, we get that γ is a random m-dimensional curve of degree δ such that $\gamma(t) = \mathbf{w}$.*
- *Semantic Security. For every two points $\mathbf{w}^0, \mathbf{w}^1 \in \{0,1\}^m$, taking:*

$$(\left[\gamma^b\right]_1, \left[t^b\right]_0) \leftarrow \mathsf{Curve}(\mathsf{pp}, \mathbf{w}^b),$$

for every $b \in \{0,1\}$, no PPT adversary can distinguish the distributions of $\left[\gamma^0\right]_1$ and $\left[\gamma^1\right]_1$ (without seeing $\left[t^0\right]_0$ or $\left[t^1\right]_0$).

Definition 1.3 specifies the (distribution of) encoded values, but it does not specify how the encodings are sampled. Indeed, curve encryption can be instantiated via several different constructions, which sample the encodings in a variety of ways. We consider two instantiations below. The first is a candidate that we conjecture to be secure in all known secure GEs. The second instantiation can be proven secure based on a simple and natural decisional base-group assumption over GEs that support re-randomization.

Candidate construction. We provide a candidate construction for curve encryption that we conjecture to be secure for all known GE candidates.

Assumption 1.4. *The construction given in Section 3.2.3 is a secure curve encryption (according to Definition 1.3).*

[2]Random sampling in arbitrary encoding levels is not a completely standard operation, but to the best of our knowledge, can be instantiated in known schemes without allowing any known attacks. See Remark 3.8.

This construction can be shown to be secure in the generic GE model, see Brakerski and Rothblum [BR14]. To the best of our knowledge, no non-generic attacks on GEs are known, unless the adversary can use re-randomization parameters or non-trivial encodings of 0. Neither of these is released by our candidate. Even if this candidate is broken by future attacks, curve encryption remains a simple, plausible and falsifiable GE primitive.

Candidate construction from the Polynomial Resolution assumption. We also construct curve encryption where security can be based on a simple and natural decisional base-group assumption on GEs that support re-randomization. We call this the "Polynomial Resolution (PR) Assumption", see below and in Section 3.2.2. We note that several recent attacks on known GE candidates [GGH13, CHL$^+$14, GHMS14, BWZ14, CLT14] leverage re-randomization to break security. In particular, these attacks also rule out the security of our polynomial resolution assumption in all known candidates that allow re-randomization. Still, this state of affairs seems quite fluid: more attacks may be discovered, as may further secure candidates. We are hopeful that secure candidates that allow re-randomization will emerge, and we view our security proof based on the PR assumption as an important contribution in showing how to directly instantiate the curve encryption primitive from a simple and natural assumption on GEs that allow re-randomization.

In a nutshell, the Polynomial Resolution assumption states that for a GE with degree bound δ, it is hard to evaluate a random (univariate) encoded polynomial of degree δ on an encoded input. The encodings of all $\delta + 1$ coefficients $(\alpha_0, \ldots, \alpha_\delta)$ and of the evaluation point t are given in level 1. Note that evaluating the polynomial at t requires total degree $(\delta + 1)$ (whereas the GE only supports degree δ). Moreover, we make a decisional assumption: we assume that it is difficult to distinguish a random encoding of the correct evaluation point (in level 1) from the encoding of a uniformly random and independent ring element z.

Assumption 1.5 (Polynomial Resolution (PR) Assumption, see Assumption 3.4)**.** *There exists a symmetric GE with degree bound δ s.t. the following two distributions are computationally indistinguishable:*

$$\left([\alpha_0]_1, \ldots, [\alpha_\delta]_1, [t]_1, \left[y = \sum_{i=0}^{\delta} \alpha_i \cdot t^i \right]_1 \right) \overset{c}{\approx} ([\alpha_0]_1, \ldots, [\alpha_\delta]_1, [t]_1, [z]_1),$$

Where $\alpha_0, \ldots, \alpha_\delta, t, z$ are independent and uniformly random ring elements, and where all the encodings above are random level-1 encodings of their respective ring elements.

1.3.2 Hardness Assumption over Asymmetric GEs

Our non-interactive arguments for bounded-depth computations use asymmetric GEs and a new hardness assumption. Roughly speaking, we assume that given an encoding $[\alpha]_{\mathbf{e}_i}$ of a random element, where $\mathbf{e}_i$ is the i-th unit vector, it is hard to find encodings $[\beta]_{\mathbf{v}}, [\gamma]_{\mathbf{u}}$ s.t. $(\beta \cdot \alpha^j = \gamma)$, $\mathbf{v}[i] = 0$, $\mathbf{u}[i] < j$.

We note that a milder assumption would simply say that it is hard to find an encoding $[\gamma]_{\mathbf{u}}$ such that both $(\alpha^j = \gamma)$ and $(\mathbf{u}[i] < j)$. We make the adversary's job easier by allowing it to use an encoding of β, but we insist that the i-th dimension of this encoding's level be 0.

We conjecture that the assumption holds over all known GE candidates [GGH13, CLT13, GGH14, BWZ14]. Note that the adversary never receives re-randomization parameters, so we are not susceptible to the attacks in [GGH13, CHL$^+$14, GHMS14, BWZ14, CLT14]. Note also that the assumption holds in the generic GE model.

Assumption 1.6 (see Assumption 3.10)**.** *For an asymmetric GE specified by* pp*, with κ dimensions, for any PPT* Adv *and any $i \in [\kappa]$, $j \in [0, \delta]$, let $[\alpha]_{\mathbf{e}_i}$ be a random level-$\mathbf{e}_i$ encoding:*

$$\Pr_{\mathsf{pp}, [\alpha]_{\mathbf{e}_i}} \left[\; [\beta]_{\mathbf{v}}, [\gamma]_{\mathbf{u}} \leftarrow \mathsf{Adv}(\mathsf{pp}, [\alpha]_{\mathbf{e}_i}) \quad ; \quad \left(\beta \cdot \alpha^j = \gamma\right), (\beta \neq 0), (\mathbf{v}[i] = 0), (\mathbf{u}[i] < j) \; \right] = \mathrm{negl} \;,$$

where $\mathbf{u}, \mathbf{v} \in \Lambda$ *(and thus the relation* $\left(\beta \cdot \alpha^j = \gamma\right)$ *can be tested efficiently, and the assumption is falsifiable).*

1.4 Further Related Work

There are several *interactive* schemes for delegating computation. Starting with the *interactive arguments* of Kilian (using [Kil92] (as generalized by Micali [Mic94]), which apply even to non-deterministic computations. Goldwasser, Kalai and Rothblum [GKR08] constructed *interactive proofs* for bounded-depth deterministic computations. Rothblum, Vadhan and Wigderson [RVW13] constructed *interactive proofs of proximity*, with sublinear time verification, for bounded-depth computations.

A line of works that we discuss extensively above constructs *privately verifiable* two-message arguments for delegating computation assuming subexponentially hard PIR or FHE schemes [KR09, KRR13, KRR14]. Several recent works construct publicly-verifiable non-interactive arguments for delegating (even non-deterministic) computation *under non-falsifiable assumptions* (see above), see for example [Gro10, Lip12, DFH12, GGPR13, BCI$^+$13, BCCT13] and more. Gentry and Wichs [GW11] showed that there are barriers to constructing *succinct* non-interactive arguments for non-deterministic computations. These barriers are not applicable to delegation schemes, since those focus on deterministic computations.

A separate line of works consider non-interactive delegation schemes in a *pre-processing model*, where the verifier is efficient in an amortized sense [GGP10, CKV10, AIK10, PRV12]. Another promising body of research, starting with [CTY11], focuses on practical efficiency and concrete implementations of delegation schemes.

Related and independent works. A recent sequence of works [BGT14, CHJV14, LP14, KLW14], all independent of our work, construct publicly-verifiable delegation schemes based on *indistinguishability obfuscation*. [GLSW14] showed that indistinguishability obfuscation can be based on the sub-exponential hardness of the Multilinear Subgroup Elimination problem in GEs. The work of [KLW14] obtains non-interactive protocols for general polynomial-time computations and can be based on the sub-exponentially hard Subgroup Elimination. The works [BGT14, CHJV14, LP14] obtain non-interactive delegation protocols for bounded-space computations based on the sub-exponentially hard Subgroup Elimination. The protocols of [BGT14, LP14] can be based on *polynomial* hardness of Subgroup Elimination, a falsifiable assumption, but this comes at the price of adding an additional message to the protocol: They get a two-message protocol, where the first message depends on the input x, rather than a non-interactive scheme in the CRS model.

Comparing with our work, our first result is a fully non-interactive protocol in the CRS model for *any polynomial-time computation*, rather than bounded space as in [BGT14, CHJV14, LP14]. Security is based on a falsifiable polynomial-time hardness assumption, whereas [KLW14] need sub-exponential hardness. We do note, however, that the assumptions are incomparable. On one hand, Indistinguishability Obfuscation is a more richly studied primitive than our new assumptions. On the other hand, it is a "heavy hammer" in terms of the concrete assumptions required and the expensive constructions. Comparing with our second result, the classes of depth-bounded and space-bounded computations are incomparable. In terms of security, our protocol achieves adaptive soundness. All other works require sub-exponential hardness assumptions to achieve adaptive soundness. For security, we only assume the hardness of a natural search problem on GEs. This assumption is arguably more plausible than the security of indistinguishability obfuscators or of decisional hardness assumption in GEs.

2 Technical Overview

We give an overview of some of the techniques and insights behind Informal Theorems 1.1 and 1.2.

2.1 Non-Interactive Arguments for P

In this section we give an overview of our techniques that give non-interactive arguments for polynomial-time computations. A key component of this result, is generating *locally consistent partial assignments* for a set of algebraic constraints. We outline this contribution below. See Section 4 for full details.

Globally consistent assignments ([BFLS91]). The following proof verification primitive is a cornerstone of PCP constructions. Let R be a ring, and $m \in \mathbb{N}$. Let $f : R^{3m+3} \to R$ be an easy-to-compute low-degree arithmetic circuit. A prover claims to know a function $X : \{0,1\}^m \to \{0,1\}$ such that:

$$\forall w_1, w_2, w_3 \in \{0,1\}^m, \; f(w_1, w_2, w_3, X(w_1), X(w_2), X(w_3)) = 0 \tag{1}$$

A proof should convince the verifier that there exists an exponential (in m) *global assignment* to the values $\{X(w)\}_{w \in \{0,1\}^m}$ that is *consistent* in the sense that it satisfies all 2^{3m} algebraic constraints in Equation (1). The verifier's complexity is only *polynomial* in m. PCP proofs for statements of this type are key in constructing PCPs for general non-deterministic computations.

***Locally* consistent *partial* assignments.** In the non-interactive argument setting, we cannot construct quite as strong a primitive as the one described above (and we do not obtain a general result for nondeterministic computations). Rather, abstracting one of the main steps of the [KRR14] construction, we aim for a somewhat relaxed guarantee for the above setting.

We construct a non-interactive argument that, rather than proving the existence of a global consistent assignment as above, proves that there exists a *(probabilistic) locally-consistent partial assignment generator* Assign. The assignment generator receives as its input Q queries $w_1, \ldots, w_Q \in \{0,1\}^m$, for $Q = \mathrm{polylog}(n)$. It generates a (distribution on) partial assignments $(a_1, \ldots, a_Q) \in \{0,1\}^Q$ to all Q of these queries, and provides two important guarantees:

1. *Local consistency:* For every sequence of Q queries $(w_1, \ldots, w_Q)$, all the algebraic constraints involving triplets of queries in $(w_1, \ldots, w_Q)$ are simultaneously satisfied:

$$\Pr_{(a_1,\ldots,a_Q) \leftarrow \mathsf{Assign}(w_1,\ldots,w_Q)} [\exists q_1, q_2, q_3 \in [Q] : f(w_{q_1}, w_{q_2}, w_{q_3}, a_{q_1}, a_{q_2}, a_{q_3}) \neq 0] = \mathrm{negl}$$

2. *No-Signaling:* The distribution of the values assigned to a specific set of wires is independent of the other wires that are queried (up to some negligible distinguishing factor).

   Formally, if two query vectors $\mathbf{w}$ and $\mathbf{w}'$ agree on a subset T of their queries, then the distributions of Assign's answers on $\mathbf{w}$ and on $\mathbf{w}'$, when restricted to the wires in T, are computationally indistinguishable.

We construct a "basic delegation protocol" that proves the existence of a locally-consistent partial assignment generator as described above. Namely, we can obtain such a generator from any prover that convinces the verifier to accept (with noticeable probability). The main challenge is proving local consistency. Once we construct Assign as above, we combine it with the "augmented circuit transformation" of [KRR14] to complete the proof of Informal Theorem 1.1. We also provide a modular construction simplifying the presentation and the analysis of this transformation. We proceed with a sketch of some of the ideas behind the "basic protocol". Full details are in Section 4.2.

2.1.1 Basic Protocol and Partial Assignment Generator for $Q = 3$

The most important property of the basic protocol is that any prover P*, who convinces the verifier to accept with noticeable probability, can be used to construct a locally-consistent partial assignment generator Assign as above, for $Q = \mathrm{polylog}(n)$. Here we sketch a simplified protocol, from which we derive an assignment generator for $Q = 3$, and some of the ideas behind the proof of security. Assign guarantees the following relaxed property:

$$\forall w_1, w_2, w_3 \in \{0,1\}^m : \Pr_{(a_1,a_2,a_3) \leftarrow \mathsf{Assign}(w_1,w_2,w_3)} [f(w_1, w_2, w_3, a_1, a_2, a_3) \neq 0] = \text{negl} \ . \tag{2}$$

Setup (honest prover). Take $\widetilde{X} : R^m \to R$ to be a low-degree extension of the correct global assignment $X : \{0,1\}^m \to \{0,1\}$. For an input $z = (w_1, w_2, w_3) \in R^{3m}$, define the polynomial:

$$P_0(\mathbf{z}) = f(\mathbf{z}, \widetilde{X}(w_1), \widetilde{X}(w_2), \widetilde{X}(w_3)) \ . \tag{3}$$

The prover's goal is convincing the verifier that this polynomial is 0 over $\{0,1\}^{3m}$. To do so, we define a sequence of *linearized* polynomials $\{P_j : R^{3m} \to R\}_{j \in [3m]}$ as follows.

$$\forall j \in [3m], P_j(\mathbf{z}) = \sum_{y_j \in \{0,1\}} \beta(y_j, z_j) \cdot P_{j-1}(z_1, \ldots, z_{j-1}, y_j, z_{j+1}, \ldots, z_{3m}) \ , \tag{4}$$

where β is a low-degree arithmetic circuit over the ring R that tests for equality over boolean inputs. By construction, P_{3m} is the *multi-linear extension* of P_0. If P_0 is 0 over $\{0,1\}^{3m}$, then P_{3m} should be 0 for all $\mathbf{z} \in R^{3m}$. Indeed, for $j = 3m$ we require that:

$$P_{3m}(\mathbf{z}) = 0 \ . \tag{5}$$

The challenge generator Gen_{BP}**.** The CRS includes (encodings of) three uniformly random low-degree curves $\gamma_1, \gamma_2, \gamma_3 : R \to R^m$. These curves are all encoded in the first level of the GE.

The prover P_{BP}**.** The (honest) proof Π includes (encodings of) polynomials computed as follows:

- Restrictions $\widetilde{X}^1, \widetilde{X}^2, \widetilde{X}^3$ of the polynomial $\widetilde{X}$ to the curves γ_1, γ_2, and γ_3.
- Restrictions $\{P_j^{(1,2,3)}\}_{j \in [0,3m]}$ of the polynomials $\{P_j\}_{j \in [0,3m]}$ to the concatenated three-variate manifold $\gamma(t_1, t_2, t_3) = (\gamma_1(t_1), \gamma_2(t_2), \gamma_3(t_3))$.

Moreover, for the restriction of P_j, the prover leaves the j-th coordinate free, adding a fourth variable that can be used to "set" this coordinate. For the restrictions of $\widetilde{X}$, the prover leaves the first coordinate free, adding a second variable that can be used to "set" this coordinate (see more on this below).

Note that, for completeness, it is important that the polynomials $\widetilde{X}$ and P_j all have low degree. This enables the (honest) prover to compute (encodings of) their restriction to the encoded curves in the CRS.

The verifier V_{BP}**.** The verifier receives the encoded curves $\gamma_1, \gamma_2, \gamma_3$, and the above restrictions of $\widetilde{X}$ and of the polynomials $\{P_j\}_{j \in [0,3m]}$. The verifier defines $P_{3m}(\mathbf{z}) \equiv 0$ (so that Equation (5) holds by definition). It then checks that Equations (3) and (4) hold for these restricted polynomials. Checking Equation (4) requires computing each polynomial P_{j-1} restricted to the manifold γ, but with the j-th coordinate replaced by 0 and by 1. Similarly, checking Equation (3) requires computing the polynomial $\widetilde{X}$ restricted to the curves γ_1, but with the first coordinate replaced by 0 and 1. (this is why the prover leaves these coordinates "free" as described above).

Security proof sketch. To prove soundness, we need to use a convincing prover P^* to construct a locally-consistent partial assignment generator Assign satisfying Equation (2). To start, suppose that P^* convinces V_{BP} with high probability in the above protocol (the probability is over the coins of Gen_{BP} and P^*). On input $(w_1, w_2, w_3) \in \{0,1\}^{3m}$, the assignment generator creates a "simulated" CRS', whose curves pass through (w_1, w_2, w_3). That is, at some uniformly random inputs $t_1^*, t_2^*, t_3^* \in R$, it holds that $\{\gamma_i(t_i^*) = w_i\}_{i \in [3]}$. Assign feeds this simulated CRS' to P^* to get a proof:

$$\Pi' = \left\{ \widetilde{X}^1, \widetilde{X}^2, \widetilde{X}^3, \{P_j^{(1,2,3)}\}_{j \in [0,3m]} \right\} \ .$$

Assign checks that Π' makes V_{BP} accept, and if so, it extracts from Π' the (un-encoded) values:

$$(a_1, a_2, a_3) \leftarrow \left(\widetilde{X}^1(t_1^*), \widetilde{X}^2(t_2^*), \widetilde{X}^3(t_3^*)\right) \ .$$

This uses the fact that the answers should be boolean, and so their encodings can be inverted using the zero-testing procedure (testing for equality to 0 and to 1). Note that for the *honest* prover, indeed we get:

$$\left\{\widetilde{X}^i(t_i^*) = \widetilde{X}(w_i)\right\}_{i \in [3]} \ .$$

It remains to show that Assign satisfies Equation (2). First, note that by semantic security of the encoded curves (see Definition 1.3), the simulated CRS' is indistinguishable from the real CRS. In particular, the verifier's acceptance probability should be roughly similar in both settings (this is one place where we use public verifiability: anyone can test whether the verifier accepts or rejects a proof). Thus, with high probability Π' should make V_{BP} accept (when using CRS'). In particular, Equation (4) should hold for for all j's, and for all inputs on the restricted polynomials P_j. Note, however, that by construction:

$$\forall \mathbf{z} \in \{0,1\}^{3m} : P_j(\mathbf{z}) = P_{j-1}(\mathbf{z}) \ .$$

In particular, since the restricted polynomials satisfy these equations on all inputs, the equations are true on the input (t_1^*, t_2^*, t_3^*), where the curves $\gamma_1, \gamma_2, \gamma_3$ pass through $(w_1, w_2, w_3) \in \{0,1\}^{3m}$. On this input, we have that $P_j^{(1,2,3)} = P_{j-1}^{(1,2,3)}$. In particular:

$$\forall j \in [3m] : \left(P_j^{(1,2,3)}(t_1^*, t_2^*, t_3^*) = 0\right) \Rightarrow \left(P_{j-1}^{(1,2,3)}(t_1^*, t_2^*, t_3^*) = 0\right) \ .$$

(We abuse notation here by ignoring the "free" variable. We can set this free variable to agree with the curve, see Section 4.2). Recall moreover that $P_{3m}^{(1,2,3)} \triangleq 0$, so we conclude that $P_0^{(1,2,3)}(t_1^*, t_2^*, t_3^*) = 0$ on $\{0,1\}^{3m}$. Since the verifier checked that (3) holds everywhere, it will also hold on this input, and we conclude that in this case Assign's output (a_1, a_2, a_3) satisfying $(f(w_1, w_2, w_3, a_1, a_2, a_3) = 0)$.

Note that this happens whenever the P^* succeeds and generates an accepting proof Π'. Even if P^* only succeeds with inverse polynomial probability, we can run it with many independently sampled CRSs until it succeeds. On the first successful iteration, Assign extracts the partial assignment. Similarly to the above, this procedure will succeed with overwhelming probability.

2.2 Non-Interactive Arguments for Bounded Depth

In this section we give an overview of the techniques used to prove Informal Theorem 1.2. In a nutshell, this is a non-interactive version of the [GKR08] interactive proof. The CRS includes graded encodings of all the GKR verifier's queries. While these encodings are not semantically secure (e.g. they allow equality testing), we use the specific algebraic properties of the GKR protocol to show that breaking soundness implies solving a hard problem on graded encodings.

We abstract a key component of our argument system: a non-interactive variant of the famous *Sum-Check Protocol* [LFKN90]. Sum-check protocols play a key role in the GKR interactive proof, and are "inherited" in our non-interactive argument. The full protocol is in Section 5. It is significantly more complex than the simple sum-check component given below. Much of this additional complexity is due to the need to carefully compose (non-interactive) sub-protocols, whose outputs are themselves encoded. These encoded sub-protocol outputs are then fed as (encoded) inputs into further sub-protocols. The encoding levels need to be carefully managed due to the bounded degree supported by the GE.

Interactive sum-check ([LFKN90]). Let R be a ring, and $m \in \mathbb{N}$. Let $f : R^m \to R$ be an easy-to-compute arithmetic circuit of individual degree δ, and $c \in R$ a value. A prover claims that:

$$\sum_{\mathbf{w} \in \{0,1\}^m} f(\mathbf{w}) = c \ . \qquad (6)$$

A proof should convince the verifier that these exponentially many (in m) summands sum to c. The verifier's complexity is only *polynomial* in m (and in δ). Proofs for statements of this type are key in constructing interactive proofs (and PCPs) for general computations.

2.2.1 Non-Interactive Sum-Check Argument

In a nutshell, we transform the canonical protocol of [LFKN90] into a non-interactive argument by sending encodings of all the verifier's challenges in the CRS.

The challenge generator Gen_{SC}**.** The CRS includes encodings of random ring elements $\{[r_i]_{\mathbf{e}_i}\}_{i\in\{1,\dots,m\}}$. The i-th ring element is encoded in the level corresponding to the i-th unit vector.

The prover P_{SC}**.** For $i \in [m]$, the honest prover uses the encoded items $\{[r_i]_{\mathbf{e}_i}\}$ to generate encodings of the polynomials:

$$g_i(\xi) = \sum_{w_{i+1},\dots,w_m\in\{0,1\}} f(r_1,\dots,r_{i-1},\xi,w_{i+1},\dots,w_m) \ .$$

We emphasize that these are univariate polynomials of degree δ. Note that the prover's claim in Equation (6) is that $(g_1(0)+g_1(1)) = c$. Note also that $g_m(\xi) = f(r_1,\dots,r_{m-1},\xi)$. Each encoded polynomial g_i is sent as a list of encoded coefficients $\{[g_{i,j}]_{\mathbf{v}_i}\}_{j\in[\delta+1]}$, where:

$$g_i(\xi) = \sum_{j=0}^{\delta} g_{i,j}\cdot\xi^j \ ,$$

and the level $\mathbf{v}_i$ is only non-zero in its first $i-1$ coordinates.

The verifier V_{SC}**.** The verifier receives encoded polynomials $\{g_i^*\}_{i\in[m]}$, and tests that:

$$g_1^*(0)+g_1^*(1) = c \tag{7}$$
$$\forall i\in[m-1]: \ g_i^*(r_i) = g_{i+1}^*(0)+g_{i+1}^*(1) \tag{8}$$
$$g_m^*(r_m) = f(r_1,\dots,r_m) \tag{9}$$

If all these test pass, the verifier accepts. Note that these tests can be run over the encoded polynomials and encoded inputs $\{[r_i]_{\mathbf{e}_i}\}$ in CRS.

A soundness property. Suppose that a cheating prover P^* convinces the verifier to accept with noticeable probability. Let $\{g_i\}_{i\in[m]}$ be the *correct* polynomials, as they would have been computed by the honest prover algorithm. Let $\{g_i^*\}_{i\in[m]}$ be the (cheating) polynomials sent by P^*. We will show that for some $i\in[m]$ it must be the case that:

$$g_i^* \not\equiv g_i, \ g_i^*(r_i) = g_i(r_i) \tag{10}$$

and the coefficients $\{g_{i,j}^*\}$ of g_i^* are given in a level $\mathbf{v}_i$ such that $\mathbf{v}_i[i]=0$ (see above).

Looking ahead, we show based on Assumption 1.6 that given the encoding of a random element $[r_i]_{\mathbf{e}_i}$, finding a level-$\mathbf{v}_i$ encoding of a low-degree non-zero polynomial that vanishes on r_i is hard contradicting Equation (10).

Proving that Equation (10) holds for some $i\in[m]$ follows the (information theoretic) soundness proof for the sum-check protocol. The prover P^* is cheating, and so:

$$((g_1^*(0)+g_1^*(1)) \neq (g_1(0)+g_1(1))) \quad \Rightarrow \quad g_1^* \not\equiv g_1 \ .$$

Moreover, because V_{SC}'s tests in (8) pass, we know that for every $i\in[m-1]$:

$$(g_i^*(r_i) \neq g_i(r_i)) \quad \Rightarrow \quad g_{i+1}^* \not\equiv g_{i+1} \ .$$

Finally, because V_{SC}'s tests in (7) we have that:

$$g^*_m(r_m) = g_m(r_m) \ .$$

From the above equations, we conclude that there must exists $i \in [m]$ such that Equation (10) holds.

Security reduction sketch. The proof of the argument system's soundness is via a reduction that uses any cheating prover P^* to break Assumption 1.6. Fix $i \in [m]$ whose existence is promised in Equation (10). The reduction gets as input an encoding $[\alpha]_{\mathbf{e}_i}$. It uses P^* to output encodings $[\beta]_{\mathbf{v}}$ and $[\gamma]_{\mathbf{u}}$ such that $(\beta \cdot \alpha^k) = \gamma$ for $k \in [\delta]$, where $\mathbf{v}[i] = 0$ and $\mathbf{u}[i] < k$.

The reduction operates by generating a CRS whose i-th encoding is set to $r_i \leftarrow \alpha$. For $i' \neq i$, it sets $r_{i'}$ to be the encoding of a uniformly random element in level $\mathbf{e}_{i'}$. The CRS is generated according to the right distribution, and so P^* should generate a cheating proof Π^*, containing coefficients $\{g^*_{i,j}\}$ such that Equation (10) holds (the encoding levels are as in P_{SC}).

The security reduction computes the *correct* coefficients $\{g_{i,j}\}$ in time 2^m (in our applications m will be logarithmic). Let k be the largest power such that $g^*_{i,k} \neq g_{i,k}$. We have that:

$$\sum_{j=0}^{k} g^*_{i,j} \cdot \alpha^j = \sum_{j=0}^{k} g_{i,j} \cdot \alpha^j \ .$$

Which implies:

$$\left(g^*_{i,k} - g_{i,k}\right) \cdot \alpha^k = \sum_{j=1}^{k-1} \left(g_{i,k} - g^*_{i,k}\right) \cdot \alpha^j \ ,$$

(with $\left(g^*_{i,k} - g_{i,k}\right) \neq 0$). Now, taking:

$$\beta = (g^*_{i,k} - g_{i,k}),\ \gamma = \sum_{j=1}^{k-1} \left(g_{i,k} - g^*_{i,k}\right) \cdot \alpha^j \ ,$$

we have, as required, that $(\beta \cdot \alpha^k) = \gamma$. Moreover, we can compute the encoding of β in level $\mathbf{v}_i$, where $\mathbf{v}_i[i] = 0$, and the encoding of γ in $\mathbf{u} = (\mathbf{v}_i + (k-1) \cdot \mathbf{e}_i)$. As required, $\mathbf{u}[i] < k$.

3 Tools and Definitions

3.1 Graded Encodings

Graded (multi-linear) encoding schemes were introduced by Garg, Gentry and Halevi [GGH13]. We use two variants of graded encodings: symmetric and asymmetric. Symmetric graded encodings are used in the delegation protocol for P computations in Section 4. The asymmetric variant is used in the delegation protocol for bounded depth computations in Section 5.

3.1.1 Symmetric Graded Encodings

In symmetric graded encodings, the *level* of an encoding is an integer.

Definition 3.1. *[Symmetric Graded Encoded Scheme] A symmetric graded encoding scheme is associated with a tuple of PPT algorithms* ($\mathsf{InstGen}, \mathsf{Samp}, \mathsf{Encode}, \mathsf{reRand}, \mathsf{Add}, \mathsf{Sub}, \mathsf{Mult}, \mathsf{isZero}$)*:*

- **Instance Generation:** $\mathsf{InstGen}$ *takes as input the security parameter* 1^n *and a degree parameter* δ*, and outputs public parameters* pp*, which define an underlying ring* R *and re-randomization parameters* rp*. We restrict our attention to graded encoding schemes where* R *is* $\mathbb{Z}_p$ *and* p *is a prime of size exponential in* n*. The public parameters* pp *determine a collection of sets* $\{E_l^\alpha : l \in [0,\delta], \alpha \in R\}$*.*
- **Sampling:** Samp *takes as input the public parameters* pp*, and outputs a level-zero encoding in* E_0^α *where* α *is a uniformly distributed element in* R*.*
- **Encoding:** Encode *takes as input the public parameters* pp*, the re-randomization parameters* rp*, a level-zero encoding* $u \in E_0^\alpha$ *and an index of a level* $l \in [\delta]$ *and outputs an encoding in* E_l^α*.*
- **Re-Randomization:** reRand *takes as input the public parameters* pp*, the re-randomization parameters* rp*, and an encoding* $u \in E_l^\alpha$ *and outputs another encoding* $u' \in E_l^\alpha$*. Moreover, for any two encodings* $u_1, u_2 \in E_l^\alpha$ *the outputs of* reRand *on* u_1 *and on* u_2 *are statistically close.*
- **Addition:** Add *takes as input the public parameters* pp *and two encodings* $u_1 \in E_l^{\alpha_1}$ *and* $u_2 \in E_l^{\alpha_2}$ *such that* u_1, u_2 *are in the same level* l*, and outputs an encoding in* $E_l^{\alpha_1+\alpha_2}$*.*
- **Subtraction:** Sub *takes as input the public parameters* pp *and two encodings* $u_1 \in E_l^{\alpha_1}$ *and* $u_2 \in E_l^{\alpha_2}$ *such that* u_1, u_2 *are in the same level* l*, and outputs an encoding in* $E_l^{\alpha_1-\alpha_2}$*.*
- **Multiplication:** Mult *takes as input the public parameters* pp *and encodings* $u_1 \in E_{l_1}^{\alpha_1}$ *and* $u_2 \in E_{l_2}^{\alpha_2}$ *such that* $l_1 + l_2 \in [0,\delta]$*, and outputs an encoding in* $E_{l_1+l_2}^{\alpha_1 \cdot \alpha_2}$*.*
- **Zero testing:** isZero *takes as input the public parameters* pp *and an encoding in* E_δ^α *and outputs 1 if and only if* $\alpha = 0$*.*

We also assume that the public parameters pp *include a trivial encoding of* 0 *and an encoding in* E_1^1*. We often denote an element in* E_l^α *by* $[\alpha]_l$*.*

Remark 3.2 (The re-randomization parameters). For security purposes we separate between the public parameters and the re-randomization parameters. Without the re-randomization parameters the scheme becomes more secure, however it no longer supports encoding of elements in levels that are higher than the 0-level, or re-randomization of encodings (see [GGH13] for more details). We note that since the public parameters include an encoding in E_1^1 it is always possible to raise the level of an encoding. However, the resulting encoding is not a random one, and we do not assume that it is secure.

Remark 3.3 (Noisy encodings). In existing graded encoding candidates [GGH13, CLT13], the encodings have noise that grows with the number of operations performed. When the noise becomes too large functionality is lost. In our applications we can set the noise parameters such that honest parties never reach the noise ceiling. To simplify notations we omit the noise management from the interface of the graded encodings.

The Polynomial Resolution assumption. We formulate a new hardness assumption on symmetric graded encoding schemes. Roughly speaking, we assume that if the maximal level is δ, then given a level-1 encoding of a random element $t \in R$, and level-1 encodings of random coefficients $\alpha_0, \dots, \alpha_\delta \in R$ of a degree δ polynomial, it is hard to distinguish a level-1 encoding of the polynomial's evaluation at t from a level-1 encoding of a random ring element.

Assumption 3.4 (Polynomial Resolution (PR))**.** *Let* $\delta = \delta(n)$ *be a polynomial degree bound, where* $n \in \mathbb{N}$ *is the security parameter. For every PPTM adversary* Adv*, the following two distributions are indistinguishable:*

$$\left(\mathsf{pp}, \mathsf{rp}, [\alpha_0]_1, \dots, [\alpha_\delta]_1, [t]_1, \left[y = \sum_{i=0}^{\delta} \alpha_i \cdot t^i\right]_1\right) \overset{c}{\approx} \left(\mathsf{pp}, \mathsf{rp}, [\alpha_0]_1, \dots, [\alpha_\delta]_1, [t]_1, [z]_1\right),$$

where (in both distributions) the ring elements $\alpha_0, \ldots, \alpha_\delta, t, z$ are independent and uniformly random, and where all the parameters and encodings are sampled as follows:

1. *Sample* $\mathsf{pp}, \mathsf{rp} \leftarrow \mathsf{InstGen}(1^n, \delta)$
2. *Using the operation* Samp *sample random level-*0 *encodings:*
$$[\alpha_1]_0, \ldots, [\alpha_\delta]_0, [t]_0, [z]_0 \quad .$$
3. *Using the operations* $\mathsf{Add}, \mathsf{Sub}, \mathsf{Mult}$*, obtain an encoding* $[y]_0$ *where:*
$$y = \sum_{i \in [0,\delta]} \alpha_i \cdot t^i \quad .$$
4. *Using the operation* $\mathsf{Encode}, \mathsf{reRand}$ *obtain random level-*1 *encodings:*
$$[\alpha_1]_1, \ldots, [\alpha_\delta]_1, [t]_1, [y]_1, [z]_1 \quad .$$

Remark 3.5 (Avoiding re-randomization). Note that in Assumption 3.4 the adversary gets access to the re-randomization parameters rp. For this reason, Assumption 3.4 is known to be false in certain candidate graded encoding constructions [GGH13, CHL$^+$14]. Very recently, Boneh *et al.* proposed an immunization techniques that modifies those candidates to allow re-randomization [BWZ14]. No attacks on the immunized schemes are known at this time. Still, in Section 3.2.3 we give an alternative assumption that suffices for our applications. There the adversary is not given the re-randomization parameters, and we conjecture that security holds in all known GE candidates.

3.1.2 Asymmetric Graded Encodings

In *asymmetric* graded encodings, the *level* of an encoding is a vector in $\mathbb{N}^\kappa$. We define a partial order over levels.

Definition 3.6 (Partial order over $\mathbb{N}^\kappa$)**.** *For $\kappa \in \mathbb{N}$ and tow vectors $\mathbf{v}, \mathbf{u} \in \mathbb{N}^\kappa$ we define:*
$$\mathbf{v} \leq \mathbf{u} \quad \Leftrightarrow \quad \forall i \in [\kappa] : \mathbf{v}[i] \leq \mathbf{u}[i] \quad .$$
In what follows we use the following notation: for $\delta \in \mathbb{N}$ and $\mathbf{v}, \mathbf{u} \in \mathbb{N}^\kappa$ let $\mathbf{v} + \mathbf{u}$ be the point-wise addition of $\mathbf{v}$ and $\mathbf{u}$, and let $\delta \cdot \mathbf{v}$ be the vector $\mathbf{v}$ where every coordinate is multiplied by δ. Define the following set:
$$\Lambda(\kappa, \delta) \triangleq \{\mathbf{v} \in \mathbb{N}^\kappa : \forall i \in [\kappa], \mathbf{v}[i] \leq \delta\} \quad .$$
For $i \in [\kappa]$ let $\mathbf{e}_i(\kappa)$ be the vector in $\mathbb{N}^\kappa$ that is 0 everywhere except for the i-th coordinate, where it is 1, and let $\mathbf{s}_i(\kappa)$ be the vector:
$$\mathbf{s}_i(\kappa) = \sum_{j \in [i]} \mathbf{e}_i(\kappa) \quad .$$
When κ is clear from the context we simply write $\mathbf{e}_i$ and $\mathbf{s}_i$.

Definition 3.7 (Asymmetric Graded Encoded Scheme)**.** *An asymmetric graded encoding scheme is associated with a tuple of PPT algorithms* $(\mathsf{InstGen}, \mathsf{Samp}, \mathsf{Encode}, \mathsf{Add}, \mathsf{Sub}, \mathsf{Mult}, \mathsf{isZero})$ *as follows:*

- **Instance Generation:** $\mathsf{InstGen}$ *takes as input the security parameter 1^n and multi-linearity parameters κ, δ, and outputs public parameters* pp*, which define an underlying ring R. We restrict our attention to graded encoding schemes where R is $\mathbb{Z}_p$ and p is a prime of size exponential in n. The public parameters* pp *determine a collection of sets $\{E^\alpha_\mathbf{v} : \mathbf{v} \in \Lambda(\kappa, \delta), \alpha \in R\}$.*

- **Sampling:** Samp *takes as input the public parameters* pp *and an index of a level* $\mathbf{v} \in \left\{\vec{\mathbf{0}}\right\} \cup \{\mathbf{e}_i\}_{i\in[\kappa]}$*, and outputs an encoding in* $E^{\alpha}_{\mathbf{v}}$ *where* α *is a uniformly distributed element in* R*.*
- **Addition:** Add *takes as input the public parameters* pp *and two encodings* $u_1 \in E^{\alpha_1}_{\mathbf{v}}$ *and* $u_2 \in E^{\alpha_2}_{\mathbf{v}}$ *such that* u_1, u_2 *are in the same level* $\mathbf{v}$*, and outputs an encoding in* $E^{\alpha_1+\alpha_2}_{\mathbf{v}}$*.*
- **Subtraction:** Sub *takes as input the public parameters* pp *and two encodings* $u_1 \in E^{\alpha_1}_{\mathbf{v}}$ *and* $u_2 \in E^{\alpha_2}_{\mathbf{v}}$ *such that* u_1, u_2 *are in the same level* $\mathbf{v}$*, and outputs an encoding in* $E^{\alpha_1-\alpha_2}_{\mathbf{v}}$*.*
- **Multiplication:** Mult *takes as input the public parameters* pp *and encodings* $u_1 \in E^{\alpha_1}_{\mathbf{v}_1}$ *and* $u_2 \in E^{\alpha_2}_{\mathbf{v}_2}$ *such that* $\mathbf{v}_1 + \mathbf{v}_2 \in \Lambda(\kappa, \delta)$*, and outputs an encoding in* $E^{\alpha_1\cdot\alpha_2}_{\mathbf{v}_1+\mathbf{v}_2}$*.*
- **Zero testing:** isZero *takes as input the public parameters* pp *and an encoding in* $E^{\alpha}_{\mathbf{v}}$ *for* $\mathbf{v} \in \Lambda(\kappa, \delta)$ *and outputs 1 if and only if* $\alpha = 0$*.*

We also assume that the public parameters pp *include a trivial encoding of* 0 *and an encoding in* $E^{1}_{\mathbf{e}_i}$ *for all* $i \in [\kappa]$*. We often denote an element in* $E^{\alpha}_{\mathbf{v}}$ *by* $[\alpha]_{\mathbf{v}}$*.*

Remark 3.8 (Sampling random encodings). Note that unlike in the symmetric graded encoding formulation (Definition 3.1), here we do not generate re-randomization parameters, and do not support the operations $\mathsf{Encode}, \mathsf{reRand}$. Instead we support an operation Samp that samples a random encoding of random elements not only in level zero, but also in any level $\mathbf{e}_i$ for $i \in [\kappa]$. Roughly, the operation Samp can be implemented by publishing in the public parameters many random encodings of random elements under the different levels. To sample a random encoding in some level, the operation Samp sums a random subset of the encodings in this level. Publishing this extra information in the public parameters in not known to reduce the security of existing candidates [GGH13, CLT13]. This sampling procedure was suggested by Brakerski [Bra14].

Remark 3.9 (Noisy encodings). In existing garaged encoding candidates [GGH13, CLT13] encodings have noise that grows with the number of operations performed. When the noise becomes too large functionality is lost. In our applications we can set the noise parameters such that honest parties never reach the noise ceiling. To simplify notations we omit the noise management from the interface of the graded encodings.

The hardness assumption. We formulate a new hardness assumption on asymmetric graded encoding schemes. Roughly speaking, we assume that given an encoding $[\alpha]_{\mathbf{e}_i}$ of a random element, where $\mathbf{e}_i$ is the i-th unit vector, it is hard to find encodings $[\beta]_{\mathbf{v}}, [\gamma]_{\mathbf{u}}$ s.t. $(\beta \cdot \alpha^j = \gamma)$, $\mathbf{v}[i] = 0$, $\mathbf{u}[i] < j$.

We note that a milder assumption would simply say that it is hard to find an encoding $[\gamma]_{\mathbf{u}}$ such that both $(\alpha^j = \gamma)$ and $(\mathbf{u}[i] < j)$. We make the adversary's job easier by allowing it to use an encoding of β, but we insist that the i-th dimension of this encoding's level be 0.

Assumption 3.10. *Let* $\kappa = \kappa(n), \delta = \delta(n)$ *be polynomials, where* $n \in \mathbb{N}$ *is a security parameter. For every poly-size adversary* Adv*, for every* $i \in [\kappa], j \in [\delta]$ *and for every* $\mathbf{v}, \mathbf{u} \in \Lambda$ *such that* $\mathbf{v}[i] = 0$ *and* $\mathbf{u}[i] < j$*:*

$$\Pr\left[\begin{array}{l} \mathsf{pp} \leftarrow \mathsf{InstGen}(1^n, \kappa(n), \delta(n)); \\ [\alpha]_{\vec{\mathbf{0}}} \leftarrow \mathsf{Samp}(\mathsf{pp}); \\ [\alpha]_{\mathbf{e}_i} \leftarrow \mathsf{Encode}(\mathsf{pp}, [\alpha]_{\vec{\mathbf{0}}}, \mathbf{e}_i); \\ [\beta]_{\mathbf{v}}, [\gamma]_{\mathbf{u}} \leftarrow \mathsf{Adv}(\mathsf{pp}, [\alpha]_{\mathbf{e}_i}); \\ \left(\beta \cdot \alpha^j = \gamma\right) \quad \wedge \quad (\beta \neq 0) \end{array}\right] \leq \mu(n) \enspace ,$$

where arithmetic is over the underlying ring R *defined by* pp*.*

Remark 3.11. Assumption 3.10 is *falsifiable* [Nao03]. We can efficiently check whether the encodings returned by Adv satisfy the equation $\beta \cdot \alpha^j = \gamma$ (and $\beta \neq 0$). This condition can be verified using the public parameters pp and the operations $\mathsf{Add}, \mathsf{Sub}, \mathsf{Mult}, \mathsf{isZero}$, applied to the adversary's output encodings. This is because, by the conditions on $\mathbf{v}, \mathbf{u}$ we have $(\mathbf{v} + j \cdot \mathbf{e}_i) \in \Lambda$ and $\mathbf{u} \in \Lambda$. Note that in particular, this does not require finding the encoded ring elements β, γ or knowing the description of the ring R. For simplicity of notations we do not explicitly describe this efficient verification procedure as part of Assumption 3.10.

Remark 3.12. We note that since we do not publish any non-trivial encodings of zero as part of our public parameters (there are no re-randomization parameters), the known "weak discrete logarithm" attacks [GGH13, CHL$^+$14] do not apply. In particular, under our formulation of asymmetric graded encodings, there are no known attack on Assumption 3.10 for existing graded encoding candidates.

Valid encoded inputs to arithmetic circuits. Let f be an arithmetic circuit taking n inputs and let:

$$\left\{[x_i]_{\mathbf{v}_i}\right\}_{i \in [n]} \quad ,$$

be an input for f encoded under public parameters pp where levels are taken from the set $\Lambda(\kappa, \delta)$. We say that the encoded input is *valid for* f if it is possible to use the operations $\mathsf{Add}, \mathsf{Sub}, \mathsf{Mult}$ to evaluate f on the encoded input such that the level of the output and of every intermediate result is in $\Lambda(\kappa, \delta)$. The *output level* of f, given an encoded input, is the level of the output encoding obtained by evaluating f on the encoded input. Given a partial encoded input, which contains encoding of some of the input variables but not all of them, we say that a partial encoded input is *valid for* f, if given level-0 encodings of the missing input variables, the resulting (full) encoded input is valid for f. The output level of f with given a partial encoded input is defined similarly.

3.2 Curves, Manifolds and their Encodings

For $m, \delta \in \mathbb{N}$, Let $\Gamma(m, \delta)$ be the set:

$$\Gamma(m, \delta) = \left\{ \mathbf{d} \in [0, \delta]^m : \sum_{i \in m} \mathbf{d}_i \leq \delta \right\} \quad .$$

We view each $\mathbf{d} \in \Gamma(m, \delta)$ as describing an m-variate of total degree δ. An m-variate manifold γ of dimension m' and degree δ is given by a set of elements:

$$\gamma = \left\{ \alpha_{i,\mathbf{d}} \in R : i \in [m'], \mathbf{d} \in \Gamma(m, \delta) \right\} .$$

such that for every $i \in [m']$:

$$\gamma_i(t_1, \ldots t_m) = \sum_{\mathbf{d} \in \Gamma(m,\delta)} \alpha_{i,\mathbf{d}} \cdot t_1^{\mathbf{d}_1} \ldots t_m^{\mathbf{d}_m} \quad .$$

Operations on manifolds. Given m-variate manifolds γ^1, γ^2, we denote by $\left(\gamma^1, \gamma^2\right)$ the concatenated $2m$–variate manifold:

$$\left(\gamma^1, \gamma^2\right)(t_1, \ldots, t_{2m}) = \left(\gamma^1(t_1, \ldots, t_m), \gamma^2(t_{m+1}, \ldots, t_{2m})\right) \quad .$$

Given an m-variate manifold γ^1 of dimension m' and an m'-variate manifold γ^2 of dimension m'', we denote by $\gamma^2 \circ \gamma^1$ the *composed* m-variate manifold of dimension m'':

$$\gamma^2 \circ \gamma^1(t_1, \ldots t_m) = \gamma^2(\gamma^1_1(t_1, \ldots t_m), \ldots, \gamma^1_{m'}(t_1, \ldots t_m)) \quad .$$

Given an m-variate manifold γ of dimension m' and $i \in [m']$, we denote by $\gamma^{i \to s}$ the $(m+1)$-variate manifold where the i-th output is overwritten by the $(m+1)$-th input s:

$$\gamma^{i \to s}(t_1, \dots, t_m, s) = (\gamma_1(t_1, \dots, t_m), \dots, \gamma_{i-1}(t_1, \dots, t_m), s, \gamma_{i+1}(t_1, \dots, t_m), \dots, \gamma_{m'}(t_1, \dots, t_m)) \ .$$

In what follows, we often think of an m-variate polynomial as an m-variate manifold of dimension 1. A *curve* of dimension m is simply a univariate manifold of that same dimension.

Let $\gamma = \{\alpha_{i,\mathbf{d}}\}$ be a manifold. Given public parameters pp for a symmetric grade encoding scheme and level ℓ we consider an encoded manifold:

$$[\gamma]_\ell = \left\{[\alpha_{i,\mathbf{d}}]_\ell\right\} \ .$$

3.2.1 Curve Encryption

In this section we define a procedure Curve, which generates semantically secure curve encryptions over symmetric GEs (see the overview in Section 1.3.1). This procedure uses re-randomization, and in Section 3.2.2 we show that its security can be based on the Polynomial Resolution Assumption (Assumption 3.4).

The procedure Curve. The procedure Curve samples a random level-1 encoded curve of dimension m and degree δ that passes though a specific point. Given public parameters pp, re-randomization parameters rp, degree parameter δ, and a point $\mathbf{w} = (w_1, \dots, w_m) \in \{0,1\}^m$:

1. Curve uses the operation Samp to sample level-zero encodings of random elements:
$$\left\{[\alpha_{i,d}]_0\right\}_{i \in [m], d \in [\delta]} \quad , \quad [t]_0 \ .$$
2. Curve uses the operation $\mathsf{Add}, \mathsf{Sub}, \mathsf{Mult}$, to obtain the encodings:
$$\left\{[\alpha_{i,0}]_0\right\}_{i \in [m]} \ ,$$
such that for every $i \in [m]$:
$$\alpha_{i,0} = w_i - \sum_{j \in [\delta]} \alpha_{i,j} \cdot t^j \ .$$
Note that when $w_i = 1$, we use a level-1 encoding of 1 (available in pp) to compute the (encoded) value of $\alpha_{i,0}$.
3. Use the operations $\mathsf{Encode}, \mathsf{reRand}$, Curve obtains random level-1 encodings:
$$[\gamma]_1 = \left\{[\alpha_{i,j}]_1\right\}_{i \in [m], j \in [0,\delta]} \ ,$$

Finally, Curve outputs the encoded curve $[\gamma]_1$ and the encoded input $[t]_0$.

3.2.2 Semantic Security of Curve

In this section we prove that, under Assumption 3.4, the Curve procedure is semantically secure, and completely hides the point $\mathbf{w}$.

Lemma 3.13. *Let m, δ be polynomial. Assuming a symmetric graded encoding scheme satisfying Assumption 3.4, there exists a negligible function μ such that for every poly-size adversary Adv, for every $n \in \mathbb{N}$, and for every $\mathbf{w}_0, \mathbf{w}_1 \in \{0,1\}^{m(n)}$, where $m(n) = \mathrm{poly}(n)$:*

$$\Pr\left[\begin{array}{l} b \leftarrow \{0,1\}; \\ \mathsf{pp}, \mathsf{rp} \leftarrow \mathsf{InstGen}(1^n, \delta(n)); \\ [\gamma]_1, [t]_0 \leftarrow \mathsf{Curve}(\mathsf{pp}, \mathsf{rp}, \delta(n), \mathbf{w}_b); \\ b' \leftarrow \mathsf{Adv}(\mathsf{pp}, \mathsf{rp}, [\gamma]_1); \\ b = b' \end{array}\right] \le \frac{1}{2} + \mu(n) \ ,$$

Proof. Assume towards contradiction that there exists a polynomial p and a poly-size adversary Adv such that for infinitely many values of $n \in \mathbb{N}$, there exist $w_0, w_1 \in \{0,1\}^{m(n)}$ such that:

$$\Pr\left[\begin{array}{l} b \leftarrow \{0,1\}; \\ \mathsf{pp}, \mathsf{rp} \leftarrow \mathsf{InstGen}(1^n, \delta(n)); \\ [\gamma]_1, [t]_0 \leftarrow \mathsf{Curve}(\mathsf{pp}, \mathsf{rp}, \delta(n), \mathbf{w}_b); \\ b' \leftarrow \mathsf{Adv}(\mathsf{pp}, [\gamma]_1); \\ b = b' \end{array}\right] \geq \frac{1}{p(n)} \ . \tag{11}$$

Next, we construct an adversary Adv' that breaks Assumption 3.4. Fix $n \in \mathbb{N}$, and $\mathbf{w}_0, \mathbf{w}_1 \in \{0,1\}^m$ such that Equation (11) holds. Adv' is given public parameters pp with an underlying field $\mathbb{Z}_p$, re-randomization parameters rp, and level-1 encodings:

$$[\alpha_0]_1, \dots, [\alpha_\delta]_1, [t]_1 \ ,$$

sampled from one of the following distributions:

Case 1. $\alpha_0, \dots, \alpha_\delta, t$ are independently random elements in $\mathbb{Z}_p$.

Case 2. $\alpha_1, \dots, \alpha_\delta, t$ are independently random elements in $\mathbb{Z}_p$ and:

$$\alpha_0 = -\sum_{i \in [\delta]} \alpha_i \cdot t^i \ .$$

Equivalently, $\alpha_0, \dots, \alpha_\delta, t$ are independently random elements in $\mathbb{Z}_p$ such that:

$$\sum_{i \in [0,\delta]} \alpha_i \cdot t^i = 0 \ .$$

Adv' samples a bit $b \leftarrow \{0,1\}$, and constructs an encoded curve $[\gamma']_1$ that is distributed as in the output of $\mathsf{Curve}(\mathsf{pp}, \mathsf{rp}, \delta(n), \mathbf{w}_b)$ in Case 1 and independently of b in Case 2. Adv' will then execute the adversary Adv on pp and the encoded curve $[\gamma']_1$ and output 1 iff Adv output b. It follows from Equation (11) that Adv breaks Assumption 3.4. Next we construct the encoded curve $[\gamma']_1$ and prove that it is distributed as required.

Let $\mathbf{t}$ be the vector:

$$\mathbf{t} = \left(1, t, t^2, \dots, t^{\delta(n)}\right)^T \ .$$

(Where the arithmetics here in in what follows is over $\mathbb{Z}_p$.) Let $\mathbf{B}$ be the matrix:

$$\mathbf{B} = \begin{pmatrix} \alpha_0 & -t & 0 & \cdots & 0 \\ & 1 & -t & & \vdots \\ \vdots & 0 & 1 & \ddots & 0 \\ & \vdots & & \ddots & -t \\ \alpha_\delta & 0 & \cdots & 0 & 1 \end{pmatrix} \ . \tag{12}$$

We will rely on the following property of the matrix $\mathbf{B}$.

Claim 3.14. *Let $\mathbf{R} \in \mathbb{Z}_p^{m \times (\delta+1)}$ be a random matrix.*

1. If:

$$\sum_{d \in [0,\delta]} \alpha_d \cdot t^d \neq 0 \ ,$$

then $\mathbf{B}$ is regular with all but negligible probability and in particular, $\mathbf{R} \cdot \mathbf{B}$ is uniform.

2. *If:*
$$\sum_{d\in[0,\delta]} \alpha_d \cdot t^d = 0 \ ,$$
$\mathbf{R}\cdot\mathbf{B}$ *is a uniform matrix such that* $\mathbf{R}\cdot\mathbf{B}\cdot\mathbf{t} = \mathbf{0}$.

Proof. Part 1 follows from the fact that:
$$\det(\mathbf{B}) = \sum_{d\in[0,\delta]} \alpha_d \cdot t^d \ .$$
Part 2 holds since:
$$\sum_{d\in[0,\delta]} \alpha_d \cdot t^d = 0 \ ,$$
implies that $\mathbf{B}$ is of rank δ and that $\mathbf{B}\cdot\mathbf{t} = 0$. □

Using its input encodings Adv' computes $[\mathbf{B}]_1$, a matrix that contains level-1 encodings of the elements of $\mathbf{B}$. Using the operation Samp, Adv' samples a matrix $\mathbf{R}\in\mathbb{Z}_p^{m\times(\delta+1)}$ of random level-0 encodings:
$$[\mathbf{R}]_0 = \left\{[r_{i,d}]_0\right\}_{i\in[m],d\in[0,\delta]} \ .$$
Let:
$$\mathbf{W} = \left\{w'_{i,d}\right\}_{i\in[m],d\in[0,\delta]} \in \mathbb{Z}_p^{m\times(\delta+1)} \ ,$$
be a matrix who's first column encodes $\mathbf{w}_b$ and is zero everywhere else. That is, for every $i\in[m]$, $w'_{i,0} = w_b[i]$ and for every $d\in[1,\delta]$, $w'_{i,d} = 0$. Adv' obtains the encoded matrix $[\mathbf{W}]_1$. Using the operations $\mathsf{Add}, \mathsf{Sub}, \mathsf{Mult}$, Adv' computes the encoded matrix $[\mathbf{R}\cdot\mathbf{B}+\mathbf{W}]_1$. Adv' uses the operation reRand to re-randomize every encoding in $[\mathbf{R}\cdot\mathbf{B}+\mathbf{W}]_1$. We think of this matrix as defining the m-dimensional degree δ encoded curve $[\gamma']_1$.

Next we prove that $[\gamma']_1$ is distributed correctly in both cases. It follows from Claim 3.14, Part 1 that in Case 1, the matrix $\mathbf{R}\cdot\mathbf{B}$ and therefore also the matrix $\mathbf{R}\cdot\mathbf{B}+\mathbf{W}$ are uniform. Since $[\gamma']_1$ consists of random encodings of the elements of $\mathbf{R}\cdot\mathbf{B}+\mathbf{W}$ (that is, encoding that have been re-randomized using the operation reRand) the distribution $[\gamma']_1$ is statistically close to a random curve that is independent of b.

It follows from Claim 3.14, Part 1 that in Case 2, $\mathbf{R}\cdot\mathbf{B}$ is a uniform matrix such that $\mathbf{R}\cdot\mathbf{B}\cdot\mathbf{t} = \mathbf{0}$. Therefore, $\mathbf{R}\cdot\mathbf{B}+\mathbf{W}$ is a uniform matrix such that $\mathbf{R}\cdot\mathbf{B}\cdot\mathbf{t} = w^T$. Since $[\gamma']_1$ consists of random encodings of the elements of $\mathbf{R}\cdot\mathbf{B}+\mathbf{W}$ it follows that the distribution $[\gamma']_1$ is statistically close the the output of $\mathsf{Curve}(\mathsf{pp}, \mathsf{rp}, \delta(n), \mathbf{w}_b)$. □

3.2.3 Avoiding Encoding Re-Randomization

As discussed in Remark 3.5, we would like to prove a version of Lemma 3.13 based on a weaker variant of the Polynomial Resolution Assumption (Assumption 3.4), where the adversary is not given the re-randomization parameters rp. The difficulty is that when using Lemma 3.13 to prove the security of the delegation protocol in Section 4, the reduction needs to execute the procedure Curve. Therefore, the reduction needs access to the re-randomization parameters. In fact, even the reduction from the the semantic security of encoded curves to Assumption 3.4 described in the proof of Lemma 3.13 needs access to the re-randomization parameter.

In this section we formulate an alternative assumption stating that there exist a procedure Curve' that samples a curve passing though a given point even without the re-randomization parameters. To support this assumption, we give a candidate implementation for the procedure Curve', which we conjecture to be secure for all existing graded encoding constructions.

In order to sample encoded curves without the use of the re-randomization parameters (and without the operations $\mathsf{Encode}, \mathsf{reRand}$) we assume that the symmetric graded encoding scheme supports a strong version of the sampling operation that can sample encodings also in level 1. Specifically, we consider the operation Samp that takes as input the public parameters pp and an index of a level $\ell \in \{0, 1\}$, and outputs an encoding in E_ℓ^α where α is a uniformly distributed element in the ring underlying pp. We assume that this element contains sufficient noise to "mask" encodings with smaller noise that may be added to it. Roughly, the operation Samp can be implemented by publishing in pp many random level-1 encodings of uniformly random elements. To sample a random encoding in level 1, the operation Samp sums a random subset of these encodings. Publishing this extra information in the public parameters is not known to reduce the security of existing candidates [GGH13, CLT13, GGH14] (see Remark 3.5). We start by formulating the assumption that a semantically secure curve encryption exists.

Assumption 3.15. *There exists a efficient procedure* Curve' *satisfying the following properties:*

- *Functionality. Given public parameters* pp*, a degree parameter* δ *and a point* $\mathbf{w} \in \{0, 1\}^m$*,* Curve' *outputs an encoded curve* $[\gamma]_1$ *of dimension* m *and degree* δ *as well as an encoding* $[t]_0$ *such that* γ *is random curve condition on* $\gamma(t) = \mathbf{w}$*.*
- *Semantic Security. Let* m, δ *be polynomial. There exists a negligible function* μ *such that for every poly-size adversary* Adv*, for every* $n \in \mathbb{N}$*, and for every* $\mathbf{w}_0, \mathbf{w}_1 \in \{0, 1\}^{m(n)}$*, where* $m(n) = \mathrm{poly}(n)$*:*

$$\Pr\left[\begin{array}{l} b \leftarrow \{0,1\}; \\ \mathsf{pp}, \mathsf{rp} \leftarrow \mathsf{InstGen}(1^n, \delta(n)); \\ [\gamma]_1, [t]_0 \leftarrow \mathsf{Curve}'(\mathsf{pp}, \delta(n), \mathbf{w}_b); \\ b' \leftarrow \mathsf{Adv}(\mathsf{pp}, [\gamma]_1); \\ b = b' \end{array}\right] \leq \frac{1}{2} + \mu(n) \ ,$$

Candidate instantiation. We consider the following candidate instantiation for the procedure Curve'. Given public parameters pp, a degree parameter δ and a point $\mathbf{w} = (w_1, \ldots, w_m) \in \{0, 1\}^m$, Curve' is defined as follows:

1. Curve' uses the operation Samp to sample a level-0 encoding $[t]_1$.
2. Curve' uses the operation Samp to sample level-1 encodings of random elements:
$$\left\{[\beta_{i,d}]_1\right\}_{i \in [m], d \in [\delta]} \ .$$
3. Curve' uses the operation $\mathsf{Add}, \mathsf{Sub}, \mathsf{Mult}$, to obtain the encodings:
$$[\gamma]_1 = \left\{[\alpha_{i,d}]_1\right\}_{i \in [m], d \in [0,\delta]} \ .$$
such that for every $i \in [m]$:
$$\alpha_{i,0} = w_i - t \cdot \beta_{i,1} \quad , \quad \alpha_{i,\delta} = \beta_{i,\delta} \ ,$$
and for every $d \in [\delta - 1]$:
$$\alpha_{i,d} = \beta_{i,d} - t \cdot \beta_{i,d+1} \ .$$
Note that when $w_i = 1$ we use a level-1 encoding of 1 (available in pp) to compute the (encoded) value $\alpha_{i,0}$.

Finally, Curve' outputs the encoded curve $[\gamma]_1$ and the encoded input $[t]_0$.

3.3 (Publicly-Verifiable) Non-Interactive Arguments

We define (publicly verifiable) non-interactive arguments with adaptive and non-adaptive soundness. By default, when we speak of arguments we refer to non-adaptive soundness unless we explicitly mention otherwise.

Definition 3.16 (Non-Interactive Argument: Adaptive and Non-Adaptive)**.** *A tuple of PPT algorithms* $(\mathsf{Gen}, \mathsf{P}, \mathsf{V})$ *are a (publicly verifiable) non-interactive argument for a language* $\mathcal{L}$ *with non-adaptive (or adaptive) soundness if they are as follows. Let* $k = k(n)$ *be a polynomial. For security parameter* $n \in \mathbb{N}$*, and for instance* $x \in \{0,1\}^k$*,* $\mathsf{Gen}(1^n)$ *outputs a string* CRS*,* $\mathsf{P}(\mathsf{CRS}, x)$ *outputs a proof* Π *and* $\mathsf{V}(\mathsf{CRS}, x, \Pi)$ *outputs one bit indicating acceptance or rejection. We require completeness and non-adaptive (or, respectively, adaptive) soundness:*

- *Completeness. There exists a negligible function* μ *such that for every* $n \in \mathbb{N}$*, and for every* $x \in \mathcal{L} \cap \{0,1\}^k$*:*
$$\Pr\left[\begin{array}{l} \mathsf{CRS} \leftarrow \mathsf{Gen}(1^n); \\ \Pi \leftarrow \mathsf{P}(\mathsf{CRS}, x); \\ \mathsf{V}(\mathsf{CRS}, x, \Pi) = 1 \end{array}\right] \geq 1 - \mu(n) \ .$$
- *Non-Adaptive Soundness. for every pair of poly-size circuits* $\mathsf{P}_1^*, \mathsf{P}_2^*$ *there exists a negligible function* μ *such that for every* $n \in \mathbb{N}$*:*
$$\Pr\left[\begin{array}{l} \mathsf{CRS} \leftarrow \mathsf{Gen}(1^n); \\ x^* \leftarrow \mathsf{P}_1^*(1^n); \\ \Pi^* \leftarrow \mathsf{P}_2^*(\mathsf{CRS}, x^*); \\ \mathsf{V}(\mathsf{CRS}, x^*, \Pi^*) = 1 \quad \wedge \quad x^* \notin \mathcal{L} \end{array}\right] \leq \mu(n) \ .$$
- *Adaptive Soundness. for every pair of poly-size circuits* $\mathsf{P}_1^*, \mathsf{P}_2^*$ *there exists a negligible function* μ *such that for every* $n \in \mathbb{N}$*:*
$$\Pr\left[\begin{array}{l} \mathsf{CRS} \leftarrow \mathsf{Gen}(1^n); \\ x^* \leftarrow \mathsf{P}_1^*(\mathsf{CRS}); \\ \Pi^* \leftarrow \mathsf{P}_2^*(\mathsf{CRS}, x^*); \\ \mathsf{V}(\mathsf{CRS}, x^*, \Pi^*) = 1 \quad \wedge \quad x^* \notin \mathcal{L} \end{array}\right] \leq \mu(n) \ .$$
- *Poly-Time Argument. The running time of* P *is at most* $\mathrm{poly}(n, k)$.

3.4 Ring-Independent Arithmetic Circuits.

In this section we define ring-independent arithmetic circuits. These are arithmetic circuits that evaluate to the same values on the boolean hypercube when evaluated over any ring. We choose to work with ring-independent arithmetic circuits since they can be evaluated over encoded inputs, where the evaluator may not know the underlying ring, and the evaluation is independent of the encoding parameters' underlying ring.

Definition 3.17 ((Strongly) Ring-Independent Arithmetic Circuit)**.** *An arithmetic circuit* C *with addition, subtraction and multiplication gates and with constants in* $\{0,1\}$ *is ring-independent if there exists a boolean function* $f : \{0,1\}^n \to \{0,1\}$ *such that for every ring* R *and for every input* $x \in \{0,1\}^n$ *we have that* $C(x) = f(x)$ *when* C *is evaluated over* R*. In this case we say that the circuit* C *computes the boolean function* f*.*

We say that C *is* strongly *ring-independent the sub-circuit computing any internal wire in* C *is ring-independent.*

Fact 3.18. *Let f, g be a pair of ring-independent arithmetic circuits on n inputs.*

1. *The circuit computing $f \cdot g$ is ring independent.*
2. *If for every $\mathbf{x} \in \{0,1\}^n$, at least one of the two values $f(\mathbf{x}), g(\mathbf{x})$ is zero, then $f + g$ is ring independent.*

3.5 Multi-linear Extension.

For a boolean function $f : \{0,1\}^n \to \{0,1\}$ and a ring R, we define the multi-linear extension $\tilde{f}_R$, a multi-linear polynomial that agrees with f on all inputs in $\{0,1\}^n$. We show how to compute $\tilde{f}_R$ over any ring R using a single ring-independent circuit (this circuit always agrees with f over the hypercube, but its values outside the hypercube will vary from ring to ring).

Let β_n be a ring-independent, multi-linear arithmetic circuit with $2n$ inputs described by the following arithmetic expression:

$$\beta_n(x_1, \ldots, x_n, y_1, \ldots, y_n) = \prod_{i \in [n]} x_i y_i + (1 - x_i)(1 - y_i) \ .$$

observe that β_n computes the identity function over input pairs from the hypercube. For every $\mathbf{x}, \mathbf{y} \in \{0,1\}^n$, $\beta_n(\mathbf{x}, \mathbf{y}) = 1$ iff $\mathbf{x} = \mathbf{y}$. (When n is clear from the context we simply write β.)

The the multi-linear extension $\tilde{f}_R$ is defined by the following ring-independent, multi-linear arithmetic circuit:

$$\tilde{f}(\mathbf{x}) = \sum_{\mathbf{y} \in \{0,1\}^n} \beta_n(\mathbf{x}, \mathbf{y}) \cdot f(\mathbf{y}) \ . \tag{13}$$

Since for every $\mathbf{x} \in \{0,1\}^n$ there exist only one value of $\mathbf{y} \in \{0,1\}^n$ such that $\beta_n(\mathbf{x}, \mathbf{y}) \neq 0$, it follows by Fact 3.18 that $\tilde{f}$ is a ring-independent arithmetic circuit computing f.

3.6 Uniform and Constructible Circuits

In this section we specify the notions of uniform and of constructible circuit ensembles, which will be used in our protocols. We then review a proof that polynomial-time computation can be decided by a constructible circuit ensemble.

Definition 3.19 (t-Uniform Circuit Ensemble). *Let $\{C_k\}_{k \in \mathbb{N}}$ be an ensemble of boolean circuits that consist only of fan-in 2 AND gates and of NOT gates. We say that the ensemble is* t-uniform *for a function $t : \mathbb{N} \to \mathbb{N}$ if there exists a Turing Machine that on input 1^k runs in time $t(k)$ and outputs C_k.*

Definition 3.20 (Constructible Circuit Ensemble). *Let $\{C_k\}_{k \in \mathbb{N}}$ be an ensemble of boolean circuits that consist only of fan-in 2 AND gates and of NOT gates. We say that the ensemble is* constructible *if it is polynomial-time uniform (and in particular has polynomial sized circuits), and there exists an ensemble of ring-independent arithmetic circuits $\left\{\widetilde{\mathsf{and}}_k, \widetilde{\mathsf{not}}_k\right\}_{k \in \mathbb{N}}$ as follows:*

- *The circuit $\widetilde{\mathsf{and}}_k$ takes as input three wire labels $u, v, w \in \{0,1\}^{\log(|C_k|)}$ in C_n. It outputs 1 if u is the output wire of an AND gate with input wires v, w.*
- *The circuit $\widetilde{\mathsf{not}}_k$ takes as input two wire labels $u, v \in \{0,1\}^{\log(|C_k|)}$ in C_k. It outputs 1 if u is the output wire of a NOT gate with input wire v.*

Moreover, we require that $\widetilde{\mathsf{and}}_k, \widetilde{\mathsf{not}}_k$ are $\mathrm{polylog}(k)$*-uniform circuit ensembles with* $\mathrm{polylog}(k)$ *degree.*

Claim 3.21 (From Turing Machines to Constructible Circuits)**.** *Every language that can be computed by a Turing Machine running in time $t(k)$ and space $s(k)$ can be computed by a* constructible *circuit ensemble $\{C_k\}_{k\in\mathbb{N}}$ of depth $O(t(k))$ and width $O(s(k))$.*

We will show constructibility by NC^1 circuits. By the following remark this suffices for the notion of constructibility given in Definition 3.20.

Remark 3.22. To show that an ensemble $\{C_k\}$ of boolean circuits is *constructible*, as per Definition 3.20, it suffices to show that the functions $\widetilde{\mathsf{and}}_k$ and $\widetilde{\mathsf{not}}_k$, restricted to boolean inputs, can be computed by $\mathrm{polylog}(k)$-uniform NC^1 boolean circuits (of depth $O(\log\log(k))$).

This is because, for any ring, replacing AND gates by multiplication and NOT gates by subtraction from 1, gives a ring-independent arithmetic circuit computing the functions $\widetilde{\mathsf{and}}_k, \widetilde{\mathsf{not}}_k$. For NC^1 boolean circuits, the degree of the resulting ring-independent circuit is bounded by $\mathrm{polylog}(k)$.

Proof of Claim 3.21. We recall the well-known construction of circuits from Turing Machines. Let T be a Turing Machine running in time $t(k)$ and space $s(k)$. We show that the language computed by T can be computed by a *constructible* circuit ensemble $\{C_k\}$ of depth $O(t(k))$ and width $O(k + s(k))$.

We assume that T is an *oblivious* Turing Machine [PF79] with an input tape and a work/output tape. We assume further that the position of the machine's reading heads in the i-th step (on the reading tape and on the work tape) can be computed by uniform NC^1 circuits of size $\mathrm{polylog}(k)$. Note that an arbitrary Turing machine can be made oblivious, e.g. by using the trivial transformation that makes a complete pass over the work and input tapes for every step of the original machine. Let q be the (constant) size of the Turing machine's state space, and σ the (constant) size of its alphabet.

For input length k, fix the circuit $C = C_k$, and set $t = t(k), s = s(k)$. The circuit C has a "sub-circuit" for each step of the computation. The i-th sub-circuit has $(k + q + (\sigma \cdot s))$ input wires carrying the computation's input x, the TM state Q_{i-1}, and the contents of the work tape after the $(i-1)$-th step. The i-th sub-circuit also has $(k + q + (\sigma \cdot s))$ output wires, carrying x, the TM state Q_i, and the contents of the work tape after the i-th step.

Let $\alpha(i) \in [k]$ be the location of the input tape reading head before step i, and $\beta(i) \in [s]$ the location of the work tape reading head before step i. The "active inputs" for the i-th step are the wires carrying the state Q_{i-1}, the $\alpha(i)$-th bit of the input, and the $\beta(i)$-th symbol in the work tape (the active inputs' bit values are carried on $O(1)$wires). In computing the i-th step's output, the input bits are unchanged. The i-th state Q_i is computed by a constant-size sub-circuit that operates on the active inputs. The work tape is unchanged, except for the $\beta(i)$-th symbol, which is computed by a constant-size sub-circuit thatoperates on the active inputs. These two constant-size circuits can be computed in constant time form the Turing Machine's description.

Constructibility. Each wire is labeled with its time-step i. For the i-th sub-circuit, each of its output wires is a function of $O(1)$ input wires, and the input wires relevant to each output wire can be computed using $\alpha(i)$ and $\beta(i)$ (by an NC^1 circuit of size $\mathrm{polylog}(k)$). The constant-size sub-computation for each output wire is then computed by a circuit constructible in constant time. We obtain the subcircuit performing the i-th computation step by composing the "outer" circuit described using gates that perform the constant-size operations for computing each output symbol, with an "inner" boolean circuit implementing these constant-size operations. To do this, we concatenate the labels for the outer and inner circuits, and use a wiring predicate that computes the AND of the outer and the inner circuits' wiring predicates. The resulting wiring predicate can be computed by a uniform NC^1 circuit of $\mathrm{polylog}(k)$ size.

Complexity. The resulting circuit C has depth $O(t(k))$ and width $O(k + s(k))$. □

3.7 Encoded Polynomials

In this section we consider multivariate polynomials, where some of the input variables have been fixed. We show how to compute the coefficients of the resulting restricted polynomial (as a function of the fixed variables). We use this procedure to compute the (encodings of) coefficients of polynomials when the fixed inputs are encoded under a GE scheme. (Naturally, when we consider encoded inputs, we can only obtain encoded coefficients).

Notation. For a multi-variate polynomial $g(\mathbf{t}) : R^k \to R$ of total degree δ, we use:

$$\Gamma(\delta) = \left\{ \mathbf{d} \in [0, \delta]^k : \sum_{i \in k} \mathbf{d}_i \leq \delta \right\} \enspace .$$

to denote the set of monomials with total degree δ. For $\mathbf{t} \in R^k$, $\mathbf{d} \in \Gamma(\delta)$ we denote by $\mathbf{t}^{\mathbf{d}}$ the value of the monomial $\mathbf{d}$ on input $\mathbf{t}$:

$$\mathbf{t}^{\mathbf{d}} = \prod_{i \in [\delta]} \mathbf{t}_i^{\mathbf{d}_i} \enspace .$$

We use $\alpha_{\mathbf{d}}(g)$ to denote the coefficient of the monomial $\mathbf{d}$ in the multi-variate polynomial g. Thus, we can write:

$$g(\mathbf{t}) = \sum_{\mathbf{d} \in \Gamma(\delta)} \alpha_{\mathbf{d}}(g) \cdot \mathbf{t}^{\mathbf{d}} \enspace .$$

Also, for two monomials $\mathbf{d}_1, \mathbf{d}_2$, we use $(\mathbf{d}_1 + \mathbf{d}_2)$ to denote the point-wise sum.

Computing coefficients. Claim 3.23 below shows how to compute the coefficients of a restricted polynomial, when some of the input variables have been fixed. Later in this work, when given encodings of the fixed variables, we follow the proof of this claim to obtain encodings of these coefficients.

Claim 3.23. *Let $f(\mathbf{t}, \mathbf{s}) : R^{k+m} \to R$ be an arithmetic circuit computing a multivariate polynomial on $(k + m)$ variables, separated into two input vectors $\mathbf{t} \in R^k$ and $\mathbf{s} \in R^m$. Let δ_t and δ_s be the total degrees of f in the variables in $\mathbf{t}, \mathbf{s}$ (respectively). Fixing any input $\mathbf{s}$, viewing f as a function of $\mathbf{t}$, we can decompose it as follows:*

$$f(\mathbf{t}, \mathbf{s}) = \sum_{\mathbf{d} \in \Gamma(\delta_t)} (\alpha_{\mathbf{d}}(f)(\mathbf{s})) \cdot \mathbf{t}^{\mathbf{d}} \enspace .$$

For every monomial $\mathbf{d} \in \Gamma(\delta_t)$, the coefficient $\alpha_{\mathbf{d}}(f)(\mathbf{s})$, viewed as a function of $\mathbf{s}$, can be computed by an arithmetic circuit of total degree δ_s and size $O(\delta_t^{2k} \cdot |f|)$. Given a description of f and the monomial $\mathbf{d}$, this circuit can be constructed in time poly$(\delta_t^k, |f|)$.

Proof. To construct a circuit computing $\alpha_{\mathbf{d}}(f)(\mathbf{s})$, we proceed by induction over the circuit f. In the base case, for constant or input variables the claim is trivial. Let ϕ be the top gate of a sub-circuit with total degree δ, and ϕ_1, ϕ_2 be its children, of total degrees δ_1, δ_2. When the top gate is an addition gate, and $\phi = \phi_1 + \phi_2$, we simply have

$$\alpha_{\mathbf{d}}(\phi) = \alpha_{\mathbf{d}}(\phi) + \alpha_{\mathbf{d}}(\phi) \enspace .$$

When the top gate is a multiplication gate, and $\phi = \phi_1 \cdot \phi_2$, we have:

$$\alpha_{\mathbf{d}}(\phi) = \sum_{\mathbf{d}_1 + \mathbf{d}_2 = \mathbf{d}} \alpha_{\mathbf{d}_1}(\phi) \cdot \alpha_{\mathbf{d}_2}(\phi) \enspace .$$

In both cases, by induction the total degree (in $\mathbf{s}$) remains bounded by $(\delta_1 + \delta_2) \leq \delta$. The number of monomials is at most δ_t^k, and so, by induction, the size is at most:

$$O\left((\delta_t^k \cdot \delta_t^k \cdot \text{poly}(|\phi_1|)) + (\delta_t^k \cdot \delta_t^k \cdot \text{poly}(|\phi_2|)) + \delta_t^k \right) = O(\delta_t^{2k} \cdot |f|) \enspace .$$

□

4 Non-Interactive Arguments for P

We construct a publicly-verifiable non-interactive argument system for any polynomial-time computation. Let $\mathcal{L}$ ba a language in P, decidable by a constructible circuit ensemble $\{C_k\}_{k \in \mathbb{N}}$ (see Section 3.6). The non-interactive argument proves statements of the form: $C_k(x) = 1$.

Similarly to [KRR14], we separate our description and analysis of the protocol into two steps:

Step 1. First, we build a "basic protocol" with a weak soundness property. If a prover P^* can make the verifier accept that $C(x) = 1$ (with noticeable probability), then we use this P^* to generate assignments for any small set S of C's wires, and these assignments will be *locally consistent* (see below).

In slightly more detail, we obtain from P^* a *Q-local consistent (partial) assignment generator* Assign. This is a probabilistic algorithm that takes as input a sequence of Q wire labels $S \in [|C|]^Q$, and outputs (a distribution on) boolean assignments to those wires (in $\{0,1\}^Q$). We obtain such an assignment generator for $Q = \mathrm{polylog}(|x|)$. With high probability the assignments generated by Assign are *locally consistent* in the following sense:

1. Assignments made to any and all input wires are consistent with the inputs x.
2. Assignments made to intermediate wires are locally consistent with the wires' gates. In particular, if S includes input wires u, v for and AND gate, and output wire w for the same gate, then the assignment given to w is the conjunction of the assignments given to u and to v. Similarly, if S includes the input wire u and output wire w of a NOT gate, then the assignment given to w is the negation of the assignment given to u.
3. The assignment given to the output wire is always 1.

We emphasize that *locally consistent assignments need not be consistent with $C(x)$'s actual computation.* Indeed, it is not immediately obvious why the existence of a locally-consistest assignment generator implies that there exists a globally consistent accepting assignment.[3]

Finally, the assignment generator is also required to be *computationally no-signaling* (see [KRR13]): the distribution of the values assigned to a specific set of wires is independent of the other wires that are queried (up to some negligible distinguishing factor). In particular, if two sets S and S' both contain a subset T of wire-queries, then the distributions of Assign's answers on S and on S', when restricted to the wires in T, are computationally indistinguishable.

A similar statement holds for the protocol of [KRR13, KRR14]. We note that, while our protocol is inspired by theirs, it is quite different. The proof of the existence of Assign with the above properties is also quite different from theirs.

Step 2. To prove that when the verifier accepts indeed $C(x) = 1$, we use the *augmented circuit construction* from [KRR14]. We describe this as a transformation from any constructible circuit ensemble $\{C_k\}$ to another constructible circuit ensemble $\{C'_k\}$ computing the same function. If there exists a local assignment generator as above for C'_k, then it must be that $C'(x) = 1$ (and thus $C(x) = 1$). The final protocol simply executes the "basic protocol" above on the augmented circuit C'. If the verifier accepts with noticeable probability, then we are guaranteed that there exists a local assignment generator for C'_k and therefore $C(x) = 1$.

The transformation we describe follows the construction and the proof of [KRR14]. In terms of exposition, we give a modular presentation of the construction and its soundness proof following the above two steps. We also add an explicit proof of constructibility for the augmented circuit, which simplifies the final protocol (for more details see Remark 4.7).

[3][KRR13] do prove such an implication for assignment generators that have a no-signaling property (see below) and when the circuit width is smaller than Q. They use this to argue soudnmness for space-bounded computations.

4.1 Completeness and Soundness

We proceed to formalize the notion of a local assignment generator and the claims made above. The basic protocol is presented in Section 4.2, and its soundness is proved in Section 4.3. The construction and soundness of the augmented circuit are in Section 4.4.

4.1.1 Basic Protocol Completeness and Soundness

The basic protocol is given by a trio of algorithms $(\mathsf{Gen}_{BP}, \mathsf{P}_{BP}, \mathsf{V}_{BP})$. It's completeness is formalized as follows:

Claim 4.1 (Basic Protocol Completeness)**.** *For every constructible ensemble of circuits $\{C_k\}_{k\in\mathbb{N}}$, the protocol $(\mathsf{Gen}_{BP}, \mathsf{P}_{BP}, \mathsf{V}_{BP})$ given in Section 4.2, satisfies the following completeness property.*

For every input $x \in \{0,1\}^k$ such that $C_k(x) = 1$:

$$\Pr_{\mathsf{CRS}\leftarrow\mathsf{Gen}_{BP}(1^n,1^k)} [1 \leftarrow \mathsf{V}_{BP}(\mathsf{CRS}, x, \mathsf{P}_{BP}(\mathsf{CRS}, x))] = 1 \ \ .$$

To formalize the basic protocol's weak soundness property, we first formalize the definition of a local assignment generator:

Definition 4.2 (Local Assignment Generator)**.** *Let $n \in \mathbb{N}$ be a security parameter, $\{C_k\}_{k\in\mathbb{N}}$ a constructible circuit ensemble, $\{x_n\}_{n\in\mathbb{N}}$ an input sequence such that $k(n) = |x_n| = \mathrm{poly}(n)$ and $S(n) = |C_{k(n)}|$.*

A Q-local assignment generator Assign *for $\{x_n\}$ is a probabilistic polynomial-time Turing Machine that takes as input a security parameter 1^n and a vector $\mathbf{w} \in [S(n)]^{Q(n)}$ of $C_{k(n)}$'s wires, and outputs (a distribution over) assignments $\mathbf{a} \in \{0,1\}^{Q(n)}$.*

Taking $Q = Q(n)$, we require that:

- *Everywhere Q-Local Consistency. For every wire-vector $\mathbf{w} = (w_1, \ldots, w_Q) \in [S(n)]^Q$, with probability $1 - \mathrm{negl}(n)$ over a draw:*

$$(\mathbf{a} = (a_1, \ldots, a_Q)) \leftarrow \mathsf{Assign}(1^n, \mathbf{w}) \ \ ,$$

  the assignment $\mathbf{a}$ is locally consistent *with the computation of C. That is, for every $i, j, k \in [Q]$:*

  1. *If $w_i = w_j$, then $a_i = a_j$.*
  2. *If w_i, w_j are the input wires of an AND gate and w_k is its output wire, then $a_k = a_i \cdot a_j$.*
  3. *If w_i is the input wire of a NOT gate and w_j is its output wire, then $a_j = 1 - a_i$.*
  4. *If w_i is an input wire in $C_{k(n)}$ then the value a_i is consistent with x_n.*
  5. *If w_i is the output wire of $C_{k(n)}$, then $a_i = 1$.*

- *No-signaling. For every poly-size distinguisher D and every subset $T \subseteq [Q]$ of the Q queries , for every $\mathbf{w}^0, \mathbf{w}^1 \in [S(n)]^Q$ such that $\mathbf{w}^0|_T = \mathbf{w}^1|_T$:*

$$\left| \Pr_{\mathbf{a}^0\leftarrow\mathsf{Assign}(1^n,\mathbf{w}^0)} \left[D(\mathbf{a}^0|_T) = 1\right] - \Pr_{\mathbf{a}^1\leftarrow\mathsf{Assign}(1^n,\mathbf{w}^1)} \left[D(\mathbf{a}^1|_T) = 1\right] \right| \leq \mathrm{negl}(n) \ \ .$$

The soundness of the basic delegation protocol says that if the verifier accepts the proof with noticeable probability, then there exists a local assignment generator for the computation.

Claim 4.3 (Basic Protocol Soundness). *For every constructible ensemble of circuits $\{C_k\}_{k\in\mathbb{N}}$ the protocol $(\mathsf{Gen}_{BP}, \mathsf{P}_{BP}, \mathsf{V}_{BP})$ given in Section 4.2, instantiated with a Graded Encoding Scheme satisfying Assumption 3.4, satisfies the following property.*

For every poly-size circuit P^*, for every sequence of inputs $\{x_n\}_{n\in\mathbb{N}}$ where $|x_n| = \mathrm{poly}(n)$, and for every polynomial p, if for every large enough $n \in \mathbb{N}$:

$$\Pr_{\mathsf{CRS}\leftarrow\mathsf{Gen}_{BP}(1^n,1^k)}\left[1 \leftarrow \mathsf{V}_{BP}(\mathsf{CRS}, x_n, \mathsf{P}^*(\mathsf{CRS}, x_n))\right] \geq \frac{1}{p(n)} \enspace ,$$

then there exists Q-local-assignment generator for $\{x_n\}$ as per Definition 4.2, where $Q = O(\log^6 n)$.

The proof of the claim is given in Section 4.3.

4.1.2 Full Protocol Completeness and Soundness

We define the augmented circuit transformation used in the full protocol and its properties.

Claim 4.4 (Augmented Circuits). *Let $\{C_k\}_{k\in\mathbb{N}}$ be a constructible ensemble of boolean circuits. There exists an* augmented ensemble $\{C'_k\}_{k\in\mathbb{N}}$ *with following properties:*

1. *The augmented ensemble $\{C'_k\}$ is also constructible.*
2. *For all input lengths $k \in \mathbb{N}$ and inputs $x \in \{0,1\}^k$: $C_k(x) = C'_k(x)$.*
3. *(Augmented Soundness.) For every sequence of inputs $\{x_n\}_{n\in\mathbb{N}}$ such that $k(n) = |x_n| = \mathrm{poly}(n)$, if there is an Q-local assignment generator for $\{x_n\}$ where $Q = O(\log^6 n)$, then for all but finitely many n's: $C_{k(n)}(x_n) = 1$.*

The proof of the claim is given in Section 4.4.

The augmented protocol. Let $\{C_k\}_{k\in\mathbb{N}}$ be a constructible ensemble of circuits and let $\{C'_k\}_{k\in\mathbb{N}}$ be the augmented constructible ensemble given by Claim 4.4. Let $(\mathsf{Gen}_{AP}, \mathsf{P}_{AP}, \mathsf{V}_{AP})$ be the protocol $(\mathsf{Gen}_{AP}, \mathsf{P}_{AP}, \mathsf{V}_{AP})$ given in Section 4.2 arguing about the augmented ensemble $\{C'_k\}$. By combining Claim 4.3 and 4.4 we derive the soundness of the augmented protocol:

Theorem 4.5. *The Protocol $(\mathsf{Gen}_{AP}, \mathsf{P}_{AP}, \mathsf{V}_{AP})$ instantiated with a Graded Encoding Scheme satisfying Assumption 3.4, is a non-interactive argument with non-adaptive soundness as per Definition 3.16.*

4.2 The Basic Protocol

In this section we construct the basic delegation protocol $(\mathsf{Gen}_{BP}, \mathsf{P}_{BP}, \mathsf{V}_{BP})$. Let $\{C_k\}_{k\in\mathbb{N}}$ be a constructible ensemble of circuits. Let $S(k) = 2^{m(k)} = \mathrm{poly}(k)$ be the size of C_k (we assume without loss of generality that $S(k)$ is a power of 2). Before describing the procedures $\mathsf{Gen}_{BP}, \mathsf{P}_{BP}, \mathsf{V}_{BP}$ we define an ensemble of ring-independent circuits $\{\phi_k\}$ that describes the structure of C_k.

4.2.1 The Circuit ϕ_x^{b}

For every input $x \in \{0,1\}^k$, let φ_x be a 3-CNF boolean formula such that $\varphi_x(y_1, \ldots, y_{S(k)}) = 1$ iff the wire-assignments $y_1, \ldots, y_{S(k)}$ describe the correct computation of the circuit $C(x)$ and also $C(x) = 1$. More specifically, φ_x contains the following clauses:

- Input clauses: For every input wire, there is a clause verifying that the value of this wire is consistent with x.

- Gate clauses: For every set of wires (of size 2 or 3) that are connected to a gate in C, there are clauses verifying that the values of these wires are consistent with the gate.
- Output clause: There is a clause verifying that value of the output wire is 1.
- Boolean clauses: For every wire, except for the output wire, there is a (seemingly redundant) clause verifying that value of the wire is either 0 or 1.

  We remark that the boolean clauses are added so that φ_x can also be used to verify the consistency of *partial* assignments that may contain general ring elements (we elaborate in the security proof).

For every triplet of bits $\mathbf{b} = (b_1, b_2, b_3) \in \{0,1\}^3$, let $\phi_x^{\mathbf{b}} : \{0,1\}^{3m} \to \{0,1\}$, be a boolean function such that for every triplet of wires $w_1, w_2, w_3 \in \{0,1\}^m$, $\phi_x^{\mathbf{b}}(w_1, w_2, w_3) = 1$ iff φ_x contains the clause:

$$(y_{w_1} = b_1) \vee (y_{w_2} = b_2) \vee (y_{w_3} = b_3) \enspace ,$$

and otherwise $\phi_x^{\mathbf{b}}(w_1, w_2, w_3) = 0$.

Claim 4.6 (Efficient Computation of $\phi_x^{\mathbf{b}}$)**.** *For every* $\mathbf{b} \in \{0,1\}^3$*, the function* $\phi_x^{\mathbf{b}}$ *can be computed by a uniform ensemble of ring-independent circuits with degree* $\delta(k) = \mathrm{polylog}(k)$ *and size* $\tilde{O}(k)$*.*

Proof. The circuit for $\phi_x^{\mathbf{b}}$ computes the sum of 4 ring-independent sub-circuits, each checking if the input represents a specific type of clause contained in φ. Since every input represents at most one type of clause, $\phi_x^{\mathbf{b}}$ is ring independent (Claim 3.18). Checking for boolean clauses and for the output clause can be implemented by a uniform ring-independent circuit of $\mathrm{polylog}(k)$ degree and $\mathrm{polylog}(k)$ size. Checking for input clauses can be implemented by a uniform ring-independent circuit of degree $\mathrm{polylog}(k)$ degree and $\tilde{O}(k)$ size. Since C_k is a constructible ensemble, checking for gate clauses can be implemented by a uniform ring-independent circuit of $\mathrm{polylog}(k)$ degree and $\mathrm{polylog}(k)$ size, using the wiring predicates of C_k (see Definition 3.20). □

Remark 4.7. By considering a constructible ensemble $\{C_k\}$, we are able to prove

Claim 4.8. *(claim.phi-easy). This will be used to argue that the verifier* V_{BP} *can efficiently evaluate* $\phi_x^{\mathbf{b}}$ *(on encoded inputs). This defers from the protocol of [KRR14] where they only argue that the ensemble* $\{C_k\}$ *is log-space uniform and therefore, the time to evaluate* $\phi_x^{\mathbf{b}}$ *may be as long as performing the entire computation. The solution of [KRR14] is to delegate the computation of* $\phi_x^{\mathbf{b}}$ *to the prover using a sperate delegation protocol for log-space computations.*

4.2.2 The Challenge Generator Gen_{BP}

Gen_{BP} is given as input the security parameter 1^n and the input size 1^k. Let $C = C_k$, $S = S(k)$, and $m = m(k)$. Let $Q = O(\log^6 n)$ be a parameter defined as in Claim 4.3, and let $\delta = \delta(k)$ (recall that $\delta(k)$ is the degree of the circuit $\phi_x^{\mathbf{b}}$). Let $\delta' = 3m \cdot (\delta + 1)$. Gen_{BP} samples public parameters and re-randomization parameters:

$$\mathsf{pp}, \mathsf{rp} \leftarrow \mathsf{InstGen}(1^n, \delta') \enspace .$$

Let $\mathbb{Z}_p$ be the underlying field of the public parameters pp. Gen_{BP} samples random curves passing though 0^m encoded in level 1 using the procedure Curve described in Section 3.2.1:

$$\forall q \in [Q], \quad [\gamma_q]_1, [t]_1 \leftarrow \mathsf{Curve}(\mathsf{pp}, \mathsf{rp}, \delta', 0^m).$$

(The output $[t]_1$ of Curve is ignored.) Gen_{BP} outputs the CRS that contains the public parameters pp and the encoded curves $[\gamma_1]_1, \ldots, [\gamma_Q]_1$.

The running time of Gen_{BP} and the length of CRS are $\mathrm{poly}(n, m(k))$.

4.2.3 The Prover P_{BP}

We start by introducing some notation. For an instance $x \in \mathcal{L}$, let $X : \{0,1\}^m \to \{0,1\}$ be a boolean function such that for every wire $w \in \{0,1\}^m$, $X(w)$ is the value of the wire w in the computation $C(x)$. Let $\widetilde{X}$ be the ring-independent multi-linear extension of X (see Section 3.5).

For every $\mathbf{b} \in \{0,1\}^3$ let $P_0^{\mathbf{b}}$ be the arithmetic circuit given by the expression:

$$P_0^{\mathbf{b}}(w_1, w_2, w_3) = \phi_x^{\mathbf{b}}(w_1, w_2, w_3) \cdot \prod_{i \in [3]} \left(1 - \beta(b_i, \widetilde{X}(w_i))\right) \ .$$

(Where β is the ring-independent circuit for computing the identity function on bits. See Section 3.5.)

For every $j \in [3m]$, let $P_j^{\mathbf{b}}$ be the arithmetic circuit given by the expression:

$$P_j^{\mathbf{b}}(z_1, \ldots, z_{3m}) = \sum_{y_1, \ldots y_j \in \{0,1\}} \beta(y_1, \ldots y_j, z_1, \ldots z_j) \cdot P_0^{\mathbf{b}}(y_1, \ldots y_j, z_{j+1}, \ldots, z_{3m}) \ .$$

It follows from Fact 3.18 that for every $j \in [0, 3m]$, $P_j^{\mathbf{b}}$ is ring-independent. Note that, for every $j \in [0, 3m]$, $P_j^{\mathbf{b}}$ is multi-linear in its first j input variables, and of individual degree at most $\delta + 1$ in its last $3m - j$ input variables (since $\widetilde{X}$ is multi-linear). Note also that since $x \in \mathcal{L}$, $P_0^{\mathbf{b}}(\mathbf{z}) = 0$ for every $\mathbf{z} \in \{0,1\}^{3m}$ and $\mathbf{b} \in \{0,1\}^3$. Therefore, for every $j \in [3m]$, for every ring R, and for every $\mathbf{z} \in R^j \times \{0,1\}^{3m-j}$, $P_j^{\mathbf{b}}(\mathbf{z}) = 0$ over R.

We continue our description of P_{BP}. Given the input:

$$\mathsf{CRS} = \left(\mathsf{pp}, \left([\gamma_1]_1, \ldots, [\gamma_Q]_1\right)\right) \ ,$$

and an instance $x \in \mathcal{L}$, P_{BP} computes the polynomials:

$$\widetilde{X} \quad , \quad \left\{P_j^{\mathbf{b}}\right\}_{\mathbf{b} \in \{0,1\}^3, j \in [3m]} \ ,$$

as above. For every $q \in [Q]$ let $\widetilde{X}^q(t, s)$ be the bivariate polynomial:

$$\widetilde{X}^q(t, s) = \widetilde{X} \circ \left(\gamma_q^{1 \to s}\right) \ ,$$

of degree at most $(\delta' \cdot (m-1))$ in t and linear in s.

For every $\mathbf{q} = (q_1, q_2, q_3) \in [Q]^3$, $\mathbf{b} \in \{0,1\}^3$ and $j \in [3m]$, let $P_{j,\mathbf{q}}^{\mathbf{b}}$ be the 4-variate polynomial:

$$P_{j-1}^{\mathbf{b},\mathbf{q}}(t_1, t_2, t_3, s) = P_{j-1}^{\mathbf{b}} \circ \left((\gamma_{q_1}, \gamma_{q_2}, \gamma_{q_3})^{j \to s}\right) \ .$$

The degree of $P_{j-1}^{\mathbf{b},\mathbf{q}}$ in the variables t_1, t_2, t_3 is at most $(\delta' \cdot m \cdot (\delta + 1))$ (depending on the value of j), and the degree in the variable s is at most $\delta + 1$.

P_{BP} uses the operations $\mathsf{Add}, \mathsf{Sub}, \mathsf{Mult}$ to obtain the encoded polynomials:

$$\Pi = \left(\left\{\left[\widetilde{X}^q\right]_{m-1}\right\}_{q \in [Q]} \quad , \quad \left\{\left[P_{j-1}^{\mathbf{b},\mathbf{q}}\right]_{(3m-j)(\delta+1)+j-1}\right\}_{\mathbf{q} \in [Q]^3, \mathbf{b} \in \{0,1\}^3, j \in [3m]}\right) \ ,$$

where $\widetilde{X}^q$ is encoded under the level $m-1$ since $\widetilde{X}$ is multi-linear, and the coefficients of the manifold $\gamma_q^{1 \to s}$ are encoded in level 1 in all output coordinates, except for the first coordinate whose coefficients are known in the clear. Similarly, $P_{j-1}^{\mathbf{b},\mathbf{q}}$ is encoded under the level $(3m - j)(\delta + 1) + j - 1$ since $P_{j-1}^{\mathbf{b}}$ is multi-linear in its first $j - 1$ coordinates and of individual degree at most $\delta + 1$ in its last $3m - j$ coordinates, and since the coefficients of the manifold $(\gamma_{q_1}, \gamma_{q_2}, \gamma_{q_3})^{j \to s}$ are encoded in level 1 in all output coordinates, except for the j coordinate whose coefficients are known in the clear. Finally, P_{BP} outputs the proof Π.

The running time of P_{BP} is $\mathrm{poly}(n, S(k))$ the length of the proof is $\mathrm{poly}(n, m(k))$.

4.2.4 The Verifier V_{BP}

V_{BP} is given as input the CRS:

$$\mathsf{CRS} = \left(\mathsf{pp}, \left([\gamma_1]_1, \dots, [\gamma_Q]_1\right)\right) \ ,$$

the instance x and the proof:

$$\Pi = \left(\left\{ \left[\widetilde{X}^q\right]_{m-1} \right\}_{q\in[Q]} \quad , \quad \left\{ \left[P^{\mathbf{b},\mathbf{q}}_{j-1}\right]_{(3m-j)(\delta+1)+j-1} \right\}_{\mathbf{q}\in[Q]^3, \mathbf{b}\in\{0,1\}^3, j\in[3m]} \right) \ ,$$

The verifier computes the following polynomials:

- For every $q \in [Q]$, define $\widetilde{X}'^q$ to be the univariate polynomial:

$$\widetilde{X}'^q \triangleq \widetilde{X} \circ \gamma_q \ . \tag{14}$$

  Note that $\widetilde{X}'^q(t)$ is just the polynomial $\widetilde{X}^q(t,s)$ (as defined by the honest prover P_{BP}), with the variable s is restricted so as to agree with $\gamma_q(t)$. That is:

$$\widetilde{X}'^q(t) \equiv \widetilde{X}^q(t, \gamma_q(t)[1]) \ ,$$

  where $\gamma_q(t)[1]$ denotes the first coordinate of the curve γ_q evaluated on t.

- For every $\mathbf{q} = (q_1, q_2, q_3) \in [Q]^3$ and $\mathbf{b} \in \{0,1\}^3$ and $j \in [3m]$, define $P'^{\mathbf{b},\mathbf{q}}_{j-1}$ to be the 3-variate polynomial:

$$P'^{\mathbf{b},\mathbf{q}}_{j-1} \triangleq P^{\mathbf{b}}_{j-1} \circ (\gamma_{q_1}, \gamma_{q_2}, \gamma_{q_3}) \ . \tag{15}$$

  We have that:

$$P'^{\mathbf{b},\mathbf{q}}_{j-1}(t_1,t_2,t_3) \equiv P^{\mathbf{b},\mathbf{q}}_{j-1}(t_1,t_2,t_3, (\gamma_{q_1}, \gamma_{q_2}, \gamma_{q_3})(t_1,t_2,t_3)[j]) \ , \tag{16}$$

  where $(\gamma_{q_1}, \gamma_{q_2}, \gamma_{q_3})(t_1,t_2,t_3)[j]$ denotes the j-th coordinate of the manifold $(\gamma_{q_1}, \gamma_{q_2}, \gamma_{q_3})$ evaluated on t_1, t_2, t_3.

- Define $P'^{\mathbf{b},\mathbf{q}}_{3m}$ be the constant zero 3-variate polynomial:

$$P'^{\mathbf{b},\mathbf{q}}_{3m} \equiv 0 \tag{17}$$

Following the above definitions, using the operations $\mathsf{Add}, \mathsf{Sub}, \mathsf{Mult}$ and the encodings in CRS and Π, V_{BP} obtains the encoded polynomials:

$$\left\{ \left[\widetilde{X}'^q\right]_m \right\}_{q\in[Q]} \quad , \quad \left\{ \left[P'^{\mathbf{b},\mathbf{q}}_j\right]_{(3m-j)(\delta+1)+j} \right\}_{\mathbf{q}\in[Q]^3, \mathbf{b}\in\{0,1\}^3, j\in[0,3m]} \ .$$

Next, V_{BP} verifies that for every $\mathbf{q} \in [Q]^3$, $\mathbf{b} \in \{0,1\}^3$ the following identities hold:

$$P'^{\mathbf{b},\mathbf{q}}_0(t_1,t_2,t_3) \equiv \phi^{\mathbf{b}}_x(\gamma_{q_1}(t_1), \gamma_{q_2}(t_2), \gamma_{q_3}(t_3)) \cdot \prod_{i\in[3]} 1 - \beta(b_i, \widetilde{X}'_{q_i}(t_i)) \ , \tag{18}$$

$$\forall j \in [3m]: \quad P'^{\mathbf{b},\mathbf{q}}_j(t_1,t_2,t_3) \equiv \sum_{y\in\{0,1\}} \beta(y, (\gamma_{q_1}, \gamma_{q_2}, \gamma_{q_3})(t_1,t_2,t_3)[j]) \cdot P^{\mathbf{b},\mathbf{q}}_{j-1}(t_1,t_2,t_3,y) \ . \tag{19}$$

Note that V_{BP} can verify these identities: it uses the operations $\mathsf{Add}, \mathsf{Sub}, \mathsf{Mult}$ and the circuit $\phi^{\mathbf{b}}_x$ to compute the encoded polynomials as a list of encoded coefficients. It then uses the operations $\mathsf{Sub}, \mathsf{isZero}$ to test Equalities (18) and (19) for every $\mathbf{q} \in [Q]^3$, $\mathbf{b} \in \{0,1\}^3$ and $j \in [0,3m]$. If all these tests pass, then V_{BP} accepts. Otherwise, it rejects.

By Claim 4.6 the running time of V_{BP} is $\mathrm{poly}(n, m(k)) \cdot \tilde{O}(k)$.

4.3 The Basic Protocol's Soundness and Assignment Generator

In this section we prove Claim 4.3. Let $\{C_k\}_{k \in \mathbb{N}}$ be a constructible ensemble of circuits. Let P^* be a poly-size prover, let p be a polynomial and let $\{x_n\}_{n \in \mathbb{N}}$ be sequence of inputs such that $k(n) = |x_n| = \mathrm{poly}(n)$ and for all large enough n:

$$\Pr_{\mathsf{CRS} \leftarrow \mathsf{Gen}_{BP}(1^n, 1^{k(n)})} [1 \leftarrow \mathsf{V}_{BP}(\mathsf{CRS}, x_n, \mathsf{P}^*(\mathsf{CRS}, x_n))] \geq \frac{1}{p(n)} \ . \tag{20}$$

We construct a Q-local assignment generator Assign for $\{x_n\}$ where $Q = O(\log^6 n)$ is as defined in the claim statement.

Local assignment generator Assign**.** Fix $n \in \mathbb{N}$ and take $k = k(n), S = S(k(n)), m = m(k(n)), Q = Q(n)$. Recall that on input 1^n and a vector of wires $\mathbf{w} = (w_1, \ldots, w_Q) \in \{0,1\}^{m \cdot Q}$, the output of Assign is a locally consistent partial assignment $\mathbf{a} \in \{0,1\}^Q$ for the wires in $\mathbf{w}$.

We start by giving a high-level description of the assignment generator. On input $(1^n, \mathbf{w})$, the generator Assign emulates Gen_{BP}, except that it samples the encoded curves $\gamma_1, \ldots, \gamma_Q$ so that the q-th curve passes through w_q instead of though 0^m:

$$\forall q \in [Q], \gamma_q(t_q) = w_q,$$

where t_q is a random and secret ring element. The curves are sampled using the sampling procedure described in Section 3.2.1. By the semantic security of the curve sampling procedure (Lemma 3.13), this change is not detectable by the prover P^*.

The generator Assign runs P^* on a CRS generated as above, and checks that the proof generated by P^* is an accepting proof. If not, it simply generates a fresh CRS using the same procedure (with curves passing through w) and tries again. After $(2n \cdot p(n))$ attempts, if no accepting proof is found, Assign aborts and outputs the all-0 assignment (this will only happen with negligible probability). Once an accepting proof is found, Assign obtains the encoded polynomials $\{\widetilde{X}'^q\}_{q \in [Q]}$ as computed by V_{BP}. Then Assign recovers the q-th bit of the partial assignment as:

$$a_q \leftarrow \widetilde{X}'^q(t_q)$$

For each bit of the partial assignment, Assign tests if it is 0 or 1 (if not, Assign fails - this will only happen with negligible probability)

The assignment generator. $\mathsf{Assign}(1^n, \mathbf{w})$ repeats the following for up to $2n \cdot p(n)$ iterations, until some iteration produces an assignment. If no iterations succeed in producing an assignment, Assign outputs the default assignment 0^Q.

1. Sample public parameters and re-randomization parameters:

$$\mathsf{pp}, \mathsf{rp} \leftarrow \mathsf{InstGen}(1^n, \delta') \ ,$$

as sampled by Gen_{BP}.

2. For every $q \in [Q]$ sample an encoded curve passing though w_q using the procedure Curve (see Section 3.2.1):

$$[\gamma_q]_1, [t_q]_0 \leftarrow \mathsf{Curve}(\mathsf{pp}, \mathsf{rp}, \delta') \ .$$

3. Set:

$$\mathsf{CRS} = \left(\mathsf{pp}, \left([\gamma_1]_1, \ldots, [\gamma_Q]_1\right)\right) \ ,$$

and run the prover P^* to obtain:

$$\Pi \leftarrow \mathsf{P}_{BP}(\mathsf{CRS}, x) \ .$$

Verify that:

$$\mathsf{V}_{BP}(\mathsf{CRS}, x, \Pi) \ = 1.$$

Otherwise, proceed to the next iteration.

4. Using the proof Π, obtain the encoded polynomial:

$$\left\{ \left[\widetilde{X}'^q \right]_m \right\}_{q \in [Q]} \ ,$$

as computed by V_{BP} (see Equation (14)).

5. Use the operation $\mathsf{Add}, \mathsf{Mult}$ to obtain the encodings:

$$\left\{ \left[\widetilde{X}'^q(t_q) \right]_m \right\}_{q \in [Q]} \ ,$$

6. For every $q \in [Q]$, use the operations $\mathsf{Sub}, \mathsf{isZero}$ to test if $\widetilde{X}'^q(t_q) = 0$ or if $\widetilde{X}'^q(t_q) = 1$. If both tests fail, outputs the default assignment 0^Q.

7. Output the assignment:

$$\left(\widetilde{X}_1'(t_1), \ldots, \widetilde{X}_Q'(t_Q) \right) \ .$$

Local assignment generator properties. We proceed to show that if indeed P^* succeeds in making V_{BP} accept (with polynomial probability), that is, if Equation (20) holds, then Assign is a Q-local assignment generator for $\{x_n\}$ satisfying the everywhere Q-local consistency and no-signaling properties in Definition 4.2

Everywhere Q-local consistency. In each iteration, Assign generates curves that pass through the wires specified by $\mathbf{w}$. The probability that P^* generates a proof that V_{BP} accepts remains almost unchanged compared to random curves passing through 0^m, and is at least $1/2p(n)$. This follows from the semantic security of the curve encodings (Lemma 3.13). Otherwise, we could use P^* and V_{BP} to distinguish random curves passing through 0^m from ones passing through $\mathbf{w}$. Thus, the probability that after $(2n \cdot p(n))$ independent iterations P^* doesn't generate a proof that makes V_{BP} accept is $\exp(-n)$.

To complete the local consistency proof, we show that in any iteration where P^* generates a proof Π that makes V_{BP} accept, it is *always* the case that the assignments derived from Π are locally consistent.

Towards this end, recall that from the proof Π, the generator Assign obtains the encoding of the polynomials $\{\widetilde{X}'^q\}$. For any triplet of queries $\mathbf{q} = (q_1, q_2, q_3) \in [Q]^3$, let t_1, t_2, t_3 to be the points where the curves $\gamma_{q_1}, \gamma_{q_2}, \gamma_{q_3}$ get values $w_{q_1}, w_{q_2}, w_{q_3}$. We want to show that the triplet of assignments:

$$a_1 \leftarrow \widetilde{X}'^q(t_1), a_2 \leftarrow \widetilde{X}'^q(t_2), a_3 \leftarrow \widetilde{X}'^q(t_3)$$

are locally consistent, as per Definition 4.2.

In Claim 4.9 below, we show that for every $\mathbf{b} \in \{0,1\}^3, j \in [3m]$ it is always the case that:

$$P_0'^{\mathbf{b},\mathbf{q}}(t_1, t_2, t_3) = 0.$$

This implies local consistency as follows. From V_{BP}'s test in Equation (18) have that that:

$$\begin{aligned}
0 &= P_0'^{\mathbf{b},\mathbf{q}}(t_1, t_2, t_3) \\
&= \phi_x^{\mathbf{b}}(\gamma_{q_1}(t_1), \gamma_{q_2}(t_2), \gamma_{q_3}(t_3)) \cdot \prod_{i \in [3]} 1 - \beta(b_i, \widetilde{X}'_{q_i}(t_i)) \\
&= \phi_x^{\mathbf{b}}(w_{q_1}, w_{q_2}, w_{q_3}) \cdot (1 - \beta(b_1, a_1)) \cdot (1 - \beta(b_2, a_2)) \cdot (1 - \beta(b_3, a_3)) \ .
\end{aligned}$$

This implies that for every clause in the 3-CNF φ_x that involves the assignments on wires $w_{q_1}, w_{q_2}, w_{q_3}$, the bits values a_1, a_2, a_3 assigned to those wires satisfy the clause. By the construction of φ_x, we conclude that the values assigned to a_1, a_2, a_3 are locally consistent (as per Definition 4.2).

Claim 4.9 below completes the proof of local consistency:

Claim 4.9 (V_{BP} Accept $\Rightarrow$ Consistency)**.** *For every input $x \in \{0,1\}^k$ and every wire-vector $\mathbf{w} = (w_1, \ldots, w_Q) \in [S(n)]^Q$, for every challenge CRS generated by $\mathsf{Assign}(1^n, \mathbf{w})$, if V_{BP} accepts the proof $\Pi \leftarrow \mathsf{P}^*(x, \mathsf{CRS})$, then the following holds.*

For every $\mathbf{q} = (q_1, q_2, q_3) \in [Q]^3, \mathbf{b} \in \{0,1\}^3$, let $P_0'^{\mathbf{b},\mathbf{q}}$ be the encoded polynomial computed by V_{BP} from Π via Equation (18), and let t_1, t_2, t_3 be the points where the curves $\gamma_{q_1}, \gamma_{q_2}, \gamma_{q_3}$ get values $w_{q_1}, w_{q_2}, w_{q_3}$. We have that:

$$P_0'^{\mathbf{b},\mathbf{q}}(t_1, t_2, t_3) = 0$$

Proof. We begin with notations. We take $\mathbf{t} = (t_1, t_2, t_3)$, and $\mathbf{z} = (\gamma_{q_1}, \gamma_{q_2}, \gamma_{q_3})(t_1, t_2, t_3)$. Note $\mathbf{z} = (w_{q_1}, w_{q_2}, w_{q_3}) \in \{0,1\}^{3m}$ is a *boolean vector.*

Following V_{BP}'s computations of the encoded polynomials:

$$\left\{ \left[P_j'^{\mathbf{b},\mathbf{q}} \right]_{(3m-j)(\delta+1)+j} \right\}_{\mathbf{q} \in [Q]^3, \mathbf{b} \in \{0,1\}^3, j \in [0,3m]} ,$$

by the fact that V_{BP} accepts, we know that for every $\mathbf{b} \in \{0,1\}^3, j \in [3m]$:

$$\begin{aligned} P_j'^{\mathbf{b},\mathbf{q}}(t_1, t_2, t_3) &= \sum_{y \in \{0,1\}} \beta(y, (\gamma_{q_1}, \gamma_{q_2}, \gamma_{q_3})(t_1, t_2, t_3)[j]) \cdot P_{j-1}^{\mathbf{b},\mathbf{q}}(t_1, t_2, t_3, y) && (21) \\ &= \sum_{y \in \{0,1\}} \beta(y, \mathbf{z}[j]) \cdot P_{j-1}^{\mathbf{b},\mathbf{q}}(t_1, t_2, t_3, y) && (22) \\ &= P_{j-1}^{\mathbf{b},\mathbf{q}}(t_1, t_2, t_3, \mathbf{z}[j]) && (23) \\ &= P_{j-1}'^{\mathbf{b},\mathbf{q}}(t_1, t_2, t_3) && (24) \end{aligned}$$

Where Equality (21) follows from V_{BP}'s test in Equation (19). Equality (22) is by definition of $\mathbf{z}$. Equality (23) follows because z is a *boolean* vector. Finally, Equality (24) follows by the definition of $P_{j-1}'^{\mathbf{b},\mathbf{q}}$ (see Equation (16)).

We conclude that under the conditions in the Claim's statement:

$$\forall \mathbf{b} \in \{0,1\}^3, j \in [3m]: \quad P_j'^{\mathbf{b},\mathbf{q}}(t_1, t_2, t_3) = \quad P_{j-1}'^{\mathbf{b},\mathbf{q}}(t_1, t_2, t_3) \tag{25}$$

Also, by V_{BP}'s test in Equation (17), we have:

$$\forall \mathbf{b} \in \{0,1\}^3 : P_{3m}'^{\mathbf{b},\mathbf{q}}(t_1, t_2, t_3) = 0 \tag{26}$$

From Equations (26) and (25), we conclude that

$$\forall \mathbf{b} \in \{0,1\}^3 : P_0'^{\mathbf{b},\mathbf{q}}(t_1, t_2, t_3) = 0$$

□

No-signaling. The no-signaling property of Assign follows rather directly from the semantic security of the encoded curves (Lemma 3.13) by a standard hybrid argument.

Assume toward contradiction that there exists a polynomial p_1 and a poly-size distinguisher D such that for infinitely many values of $n \in \mathbb{N}$, there exists a set $T \subseteq [Q]$, and wire vectors $\mathbf{w}^0, \mathbf{w}^1 \in \{0,1\}^{m \cdot Q}$ such that that $\mathbf{w}^0|_T = \mathbf{w}^1|_T$ and:

$$\left| \Pr_{\mathbf{a} \leftarrow \mathsf{Assign}(1^n, \mathbf{w})} [D(\mathbf{a}|_T) = 1] - \Pr_{\mathbf{a} \leftarrow \mathsf{Assign}(1^n, \mathbf{w})} [D(\mathbf{a}|_T) = 1] \right| \geq \frac{1}{p_1(n)} \ . \tag{27}$$

Let $\delta' = \delta'(n)$ be a degree parameter defined as in the procedure Gen_{BP} and let:

$$\mathsf{pp}, \mathsf{rp} \leftarrow \mathsf{Gen}_{BP}(1^n, \delta') \ .$$

For every $w \in \{0,1\}^m$ let $[\gamma_w]_1$ be the encoded curve generated by $\mathsf{Curve}(\mathsf{pp}, \mathsf{rp}, \delta', \mathbf{w})$.

We use the D to construct another distinguisher D' such that:

$$\left| \begin{array}{c} \Pr\left[D'\left(\mathsf{pp}, \mathsf{rp}, \left\{\left[\gamma_{\mathbf{w}^0_q}\right]_1\right\}_{q \in [Q] \setminus T}\right) = 1\right] - \\ \Pr\left[D'\left(\mathsf{pp}, \mathsf{rp}, \left\{\left[\gamma_{\mathbf{w}^1{}_q}\right]_1\right\}_{q \in [Q] \setminus T}\right) = 1\right] \end{array} \right| \geq \frac{1}{n \cdot p(n) \cdot p_1(n)} \ . \tag{28}$$

(Recall that the polynomial p defined the success probability of the adversary P^* as per Equation (20).) We get a contradiction to Lemma 3.13 by a standard hybrid argument.

To prove Equation (28) we consider a sequence of hispid distributions. Recall that the strategy of Assign proceeds in at most $n \cdot p(n)$ iterations until an assignment is produced. For every $i \in [0, n \cdot p(n)]$ let $\mathbf{a}_i$ be an assignment produced by the assignment generator that follows the strategy of $\mathsf{Assign}(1^n, \mathbf{w}^0)$ for the first i iterations, and the strategy of $\mathsf{Assign}(1^n, \mathbf{w}^1)$ for the rest of the iterations. By Equation (27) we have that for some $i \in [n \cdot p(n)]$:

$$\left|\Pr\left[D(\mathbf{a}^i|_T) = 1\right] - \Pr\left[D(\mathbf{a}^{i-1}|_T) = 1\right]\right| \geq \frac{1}{n \cdot p(n) \cdot p_1(n)} \ . \tag{29}$$

Fix such i. The distinguisher D' is defined as follows. Given as input parameters pp, rp and encoded curves:

$$\left\{\left[\gamma_{\mathbf{w}^b_q}\right]_1\right\}_{q \in [Q] \setminus T} \ ,$$

for some $b \in \{0,1\}$, D' emulates the assignment generator Assign as follows. In the first $i-1$ iterations D' follows the strategy of $\mathsf{Assign}(1^n, \mathbf{w}^0)$ except it it uses its input parameters pp, rp instead of sampling parameters on its own in Step 1. From the $i+1$-th iteration onwards, D' follows the strategy of $\mathsf{Assign}(1^n, \mathbf{w}^1)$ using the parameters pp, rp. If an assignment $\mathbf{a}$ is obtained in one of these iterations, D' obtains $\mathbf{a}|_T$.

In the i-th iteration, D' emulates Assign using the parameters pp, rp except that in Step 3 it samples CRS as follows. For every $q \in T$, D' samples:

$$\gamma_{\mathbf{w}^0{}_q} 1, [t_q]_0 \leftarrow \mathsf{Curve}(\mathsf{pp}, \mathsf{rp}, \delta', \mathbf{w}^0_q) \ .$$

The challenge CRS consists of the sampled curves as well as the input curves:

$$\left\{\left[\gamma_{\mathbf{w}^0{}_q}\right]_1\right\}_{q \in T} \quad , \quad \left\{\left[\gamma_{\mathbf{w}^b_q}\right]_1\right\}_{q \in [Q] \setminus T} \ .$$

Since $\mathbf{w}^0|_T = \mathbf{w}^1|_T$ we have that CRS is distributed as in the execution of $\mathsf{Assign}(1^n, \mathbf{w}^b)$. Using the challenge CRS, D' continues to emulates the i-th iteration of Assign. If an accepting proof is produced in Step 4, since D' only has t_q for $q \in T$, it continues to emulates Steps 5,6 and 7 only for $q \in T$ and obtains an the assignment $\mathbf{a}|_T$.

Finally D' outputs the same as $D(\mathbf{a}|_T)$. We argue that the assignment $\mathbf{a}|_T$ is indistinguishable from $\mathbf{a}^i|_T$ if $b = 0$ and from $\mathbf{a}^{i-1}|_T$ if $b = 1$. This, together with (29) proves (28) and concludes the proof of the no-signaling property.

Assume without loss of generality that $b = 0$. The only deference between the distributions $\mathbf{a}|_T$ and $\mathbf{a}^i|_T$ is that when sampling $\mathbf{a}^i|_T$, if in the i-th iteration of Assign an accepting proof is produced in Step 4 but for some $q \in [Q] \setminus T$ both tests in Step 6 fail, then $\mathbf{a}^i|_T$ is set to the default assignment, while $\mathbf{a}^i|_T$ will not (since D' only emulates Step 6 for $q \in T$). However, following the proof of Claim 4.9 this event only happens with negligible probability.

4.4 The Augmented Circuit

In this section we review the augmented circuit transformation from [KRR14]. We show that the augmented circuit remains *constructible*, and prove the soundness property stated in Claim 4.4.

4.4.1 Construction

We follow the augmented circuit construction from [KRR14]. For each layer of the circuit, we augment it with a computation of its low-degree extension (LDE), and with a (seemingly redundant) computation of low degree tests (LDTs) on this extension. We also verify that every low-degree test in every level is successful.

In reviewing this construction, we take care to explicitly label the circuit wires so as to ensure *constructibility* of the augmented circuit (see Definition 3.20). While [KRR14] argued log-space uniformity for the entire augmented circuit, we consider "gate-by-gate uniformity", and show how to verify the wiring of each gate in poly-logarithmic time and degree. (In fact, we show *constructibility* using an NC^1 circuit of poly-logarithmic size, see Remark 3.22).

Let $\{C_k\}_{k\in\mathbb{N}}$ be a constructible circuit ensemble. Fix an input length k and the circuit $C = C_k$ of depth $T = T(k)$ and width $S = S(k)$. Let $\mathbb{G}$ be a finite field of characteristic 2 and size $O(\log^2 S)$. Let $\mathbb{H} \subset \mathbb{G}$ be a subfield of $\mathbb{G}$ of size $\log S$, and fix the dimension $m = (\log S/\log\log S)$, so that $|\mathbb{H}|^m = S$ and $m \cdot |\mathbb{H}| < (|\mathbb{G}| - 1)/2$. (We assume without loss of generality that $(\log S)$, $(\log S/\log\log S)$ are integers.)

We assume that C is a layered circuit. Note that any constructible circuit ensemble can be transformed into a constructible ensemble of layered circuits by adding "dummy gates" to each layer. This transformation increases the width of the circuit, introducing a quadratic size overhead.

Each layer of the original circuit C is augmented using a constructible sub-circuit $\mathsf{AUG} : \{0,1\}^S \to \{0,1\}^{\mathrm{poly}(S)}$. We start by describing the sub-circuit AUG as an arithmetic circuit over $\mathbb{G}$ and show that is it constructible. We define constructibility for an arithmetic circuit similarly to the case of boolean circuits (Definition 3.20). The only difference is that the wiring predicates are for arithmetic addition, multiplication and inversion gates. After describing AUG as an arithmetic circuit, we explain how to turn it into a boolean circuit in a way that preserves constructibility.

The sub-circuit AUG is composed of two steps: computing an LDE of that layer's computation, and then running LDTs. Each step is performed by a constructible arithmetic sub-circuit over $\mathbb{G}$. We proceed with a description of these sub-circuits:

The LDE sub-circuits. To compute the LDE, we treat AUG's S-bit input as a function $V : \mathbb{H}^m \to \{0,1\}$, and compute the LDE $\widetilde{V} \in \mathbb{G}^m$. For $\mathbf{h} \in \mathbb{H}^m$, the input wire carrying $V(\mathbf{h})$ is labeled $(\mathsf{INP}, \mathbf{h})$. For each $\mathbf{a} \in \mathbb{G}^m$, the $(\mathsf{LDE}, \mathbf{a})$-subcircuit of AUG computes the value $\widetilde{V}(\mathbf{a})$, and its output wire is labeled $(\mathsf{LDE}, \mathbf{a})$.

Recall the definition of the LDE:

$$\widetilde{V}(\mathbf{a}) = \sum_{\mathbf{h}\in\mathbb{H}^m} I(\mathbf{a},\mathbf{h}) \cdot V(\mathbf{h}), \tag{30}$$

Where I is the identify function over $\mathbb{H}^m$. For each $\mathbf{h} \in \mathbb{H}^m$, the sub-circuit computes the value:

$$I(\mathbf{a}, \mathbf{h}) = \prod_{i \in [m]} \prod_{h' \in \mathbb{H} \setminus \{\mathbf{h}_i\}} \frac{(\mathbf{a}_i - h')}{(\mathbf{h}_i - h')}, \tag{31}$$

using a straightforward and constructible $(\mathsf{LDE}, \mathbf{a}, \mathsf{IDN}, \mathbf{h})$-subcircuit (whose output wire is labeled $(\mathsf{LDE}, \mathbf{a}, \mathsf{IDN}, \mathbf{h})$). Given the intermediate values $\{I(\mathbf{a}, \mathbf{h})\}_{\mathbf{h} \in \mathbb{H}^m}$, the value of $\widetilde{V}(\mathbf{a})$ is computed as in Equation (30), using a constructible subcircuit whose inputs are on the wires:

$$\{(\mathsf{LDE}, \mathbf{a}, \mathsf{IDN}, \mathbf{h}), (\mathsf{INP}, \mathbf{h})\}_{\mathbf{h} \in \mathbb{H}^m}.$$

The LDT sub-circuits. In its second (and seemingly redundant) step, the circuit AUG runs a low-degree test to verify that the restriction of $\widetilde{V}$ to every line $L : \mathbb{G} \to \mathbb{G}^m$ is a univariate polynomial of degree $m \cdot |\mathbb{H}|$ (at most). For each line L specified by $\mathbf{a}, \mathbf{b} \in \mathbb{G}^m$, the $(\mathsf{LDT}, \mathbf{a}, \mathbf{b})$ sub-circuit checks that the restriction of $\widetilde{V}$ to the line:

$$L(t) = (\mathbf{a} \cdot t) + \mathbf{b}$$

has degree at most $m \cdot |\mathbb{H}|$. We denote the polynomial obtained using this restriction by:

$$\gamma(t) = \widetilde{V}(L(t)).$$

The input to this sub-circuit are the $|\mathbb{G}|$ values $\{\widetilde{V}(L(t))\}_{t \in \mathbb{G}}$, carried on wires:

$$\{(\mathsf{LDE}, \mathbf{y})\}_{\mathbf{y} = (\mathbf{a} \cdot t + \mathbf{b}), t \in \mathbb{G}} \ .$$

The output wire is labeled $(\mathsf{LDT}, \mathbf{a}, \mathbf{b})$. Note that these input wire labels can be recognized (and computed) by a circuit of $O(|\mathbb{G}|^3)$ size and $O(\log |\mathbb{G}|)$ depth (e.g. using a hardwired truth table for all possible $(\mathbf{a}, \mathbf{b}, \mathbf{y})$). This will be used by the circuit's wiring predicate.

The low-degree test is performed by considering an arbitrary subset of field elements $\mathbb{H}' \subset \mathbb{G}$, of size $|\mathbb{H}'| = (m \cdot |\mathbb{H}| + 1)$. We use interpolation to compute a polynomial $\gamma'(t)$ of degree $|\mathbb{H}'| - 1 = m \cdot |\mathbb{H}|$, which agrees with the values of $\gamma(t)$ on the subset $\mathbb{H}'$. We then verify that $\gamma'(t) = \gamma(t)$ also for all $t \in \mathbb{G} \setminus \mathbb{H}'$. This test passes if and only if the original polynomial $\gamma(t)$ has degree at most $m \cdot |\mathbb{H}|$.

The interpolation of γ' is done using the equation:

$$\gamma'(t) = \sum_{s \in \mathbb{H}'} I(t, s) \cdot \widetilde{V}(L(s)), \tag{32}$$

where I is the identity function, defined as in Equation (31):

$$I(t, s) = \prod_{s' \in \mathbb{H}' \setminus \{s\}} \frac{(t - s')}{(s - s')},$$

The $(\mathsf{LDT}, \mathbf{a}, \mathbf{b})$-subcircuit begins by computing the values $\{\gamma(t) = \widetilde{V}(\mathbf{a} \cdot t + \mathbf{b})\}_{t \in \mathbb{G}}$ using a constructible circuit with output wire $(\mathsf{LDT}, \mathbf{a}, \mathbf{b}, \gamma, t)$. This computation of γ's values uses the fact that the labels $\{(\mathsf{LDE}, \mathbf{y})\}_{\mathbf{y} = (\mathbf{a} \cdot t + \mathbf{b}), t \in \mathbb{G}}$ can be recognized by an NC^1 circuit of size $\mathrm{poly}(|\mathbb{G}|)$ (see above).

We also compute the values of $\{\gamma'(t)\}_{t \in \mathbb{G}}$ (as in Equation (32), using a sub-circuit with output wire $(\mathsf{LDT}, \mathbf{a}, \mathbf{b}, \gamma', t)$. To do this, we begin by computing $\{I(t, s)\}_{t \in \mathbb{G}, s \in \mathbb{H}'}$, taking the label $(\mathsf{LDT}, \mathbf{a}, \mathbf{b}, \mathsf{IDN}, t, s)$ to be the output wire of the (straightforward) constructible sub-circuit computing this value (for t, s). We also compute $\{\widetilde{V}(L(s))\}_{s \in \mathbb{H}'}$ using a constructible circuit with output wire $(\mathsf{LDT}, \mathbf{a}, \mathbf{b}, \widetilde{V}, s)$ (again using the fact that the wiring predicate can recognize labels on the line L). Finally, we can compute the $\{\gamma'(t)\}$ values using a constructible sub-circuit that takes as its input the wires

$$\{(\mathsf{LDT}, \mathbf{a}, \mathbf{b}, \mathsf{IDN}, t, s), (\mathsf{LDT}, \mathbf{a}, \mathbf{b}, \widetilde{V}, s)\}_{t \in \mathbb{G}, s \in \mathbb{H}'}$$

and performs the computation of Equation (32).

The final output on wire $(\mathsf{LDT}, \mathbf{a}, \mathbf{b})$ is computed by a constructible circuit whose inputs are on the wires $\{(\mathsf{LDT}, \mathbf{a}, \mathbf{b}, \gamma, t), (\mathsf{LDT}, \mathbf{a}, \mathbf{b}, \gamma', t)\}_{t \in \mathbb{G}}$ testing that for every $t \in \mathbb{G}$, $\gamma(t) = \gamma(t')$.

Constructibility of the AUG **sub-circuit** The sub-circuit AUG takes as input wires $\{(\mathsf{INP}, \mathbf{h})\}_{\mathbf{h} \in \mathbb{H}^m}$ encoding the computation of a lyer in original circuit C. It output the wires $\{(\mathsf{LDE}, \mathbf{h})\}_{\mathbf{h} \in \mathbb{H}^m}$ computed by the LDE sub-circuits and the wires $\{\mathsf{LDT}, \mathbf{a}, \mathbf{b}\}_{\mathbf{a}, \mathbf{b} \in \mathbb{G}^m}$ computed by the LDT sub-circuits. The LDE and LDT sub-circuits are themselves constructed from (constant number of) smaller constractable sub-sircuits as described above. The wiring predicate for AUG simply computs the disjunction of wiring predicates for all these sub-circuits, which remains an NC^1 boolean circuit of $\mathrm{poly}(|\mathbb{G}|)$ size.

Turning AUG **into a boolean circuit.** Next we describe how to turn AUG into a constructible *boolean* circuit $\mathsf{AUG_b}$. This is done by replacing arithmetic gates by boolean implementations of the finite field operations. We view this translation as a "composition" of an "external" circuit using arithmetic gates, with an "internal" boolean circuits implementing the arithmetic gates. The conversion from arithmetic to boolean maintains constructibility because the boolean implementations are themselves constructible.

Specifically, for every type of arithmetic gate A let $\{C_k^A\}$ be a constructible "internal" ensemble of boolean circuits implementing the gate A over $\mathbb{G}$ (note that the field $\mathbb{G}$ depends on k) the circuit $\mathsf{AUG_b}$ is constructed by concatenating wire labels for $\mathsf{AUG_b}$ with those for C^A. The wiring predicate for a boolean gate B of $\mathsf{AUG_b}$ is computed by checking if that: (i) the external wires are connected to a gate A in the external circuit (this check is performed using the wiring predicates of all gates in $\mathsf{AUG_b}$), and (ii) the internal wires are connected to a gate of type B in C^A (this check is performed using the wiring predicates of the gate B in the ensemble $\{C_k^A\}$).

Another technical point is that we use elements from the field $\mathbb{G}$, and even vectors in $\mathbb{G}^m$, to label wires in the arithmetic circuit $\mathsf{AUG_b}$. These labels can be converted to and from boolean strings using an NC^1 circuit of $\mathrm{poly}(|\mathbb{G}|)$ size (e.g. using a fixed translation table to translate one field element at a time). We treat this translation as being (implicitly) performed by the wiring predicates whenever it is needed.

Complexity of $\mathsf{AUG_b}$**.** The boolean implementations of the arithmetic gates have size $\mathrm{poly}(|\mathbb{G}|)$ and depth $\log |\mathbb{G}|$. Moreover, since $\mathbb{G}$ has characteristic 2, addition can actually be implemented by a boolean circuit with constant depth and fan-in 2.

The arithmetic circuit computing $(\mathsf{LDE}, \mathbf{a})$ uses depth of $O(\log m \cdot \mathrm{polylog}|\mathbb{G}|)$ to compute the Identity function, and depth $O(m \cdot \log |\mathbb{H}|)$ for computing the sum in (30). The size is $\mathrm{poly}(|\mathbb{G}|^m)$. Translating the arithmetic circuit into a boolean circuit we get total depth $O(\log S)$ and total size $\mathrm{poly}(S)$. (This uses the fact that addition over fields of characteristic 2 can be implemented using constant depth fan-in 2 circuits). The arithmetic circuit computing $(\mathsf{LDT}, \mathbf{a}, \mathbf{b})$ has depth $O(\log |\mathbb{G}|)$ and size $O(|\mathbb{G}|^2)$. Translating the arithmetic circuit into a boolean circuit we get total depth $O(\mathrm{poly} \log \log(S))$, and total size $O(\log^6 S)$. Overall the circuit $\mathsf{AUG_b}$ is of total depth $O(\log S)$, and total size $\mathrm{poly}(S)$.

The augmented circuit C'**.** The complete construction of the augmented C' simply augments each layer using $\mathsf{AUG_b}$. Numbering layers from bottom (input) to top (output), the augmented circuit runs the original circuit's ℓ-th layer computation on the LDE computed by $\mathsf{AUG_b}$ on the prior layer (restricted to the values in $\mathbb{H}^m$). Finally, we also add an AND gate that checks that, for all layers, all the low degree tests were successful. This is an AND of $T \cdot \mathrm{poly}(S)$ bits accumulated throughout the computation.

The wires are labelled as follows. We append ℓ to the all wires dealing with the ℓ-th layer. The $\mathbb{H}^m$ input wires to layer ℓ are labeled $\{(\ell - 1, \mathsf{LDE}, \mathbf{h})\}_{\mathbf{h} \in \mathbb{H}^m}$. The computation of the original circuit's ℓ-th layer is performed on this input, carried forward on the wires labeled $\{(\ell, \mathsf{INP}, \mathbf{h})\}_{\mathbf{h} \in \mathbb{H}^m}$, and fed as input to the ℓ-th iteration of $\mathsf{AUG_b}$, whose outputs are on wires $\{(\ell, \mathsf{LDE}, \mathbf{h})\}_{\mathbf{h} \in \mathbb{H}^m}$.

The wiring predicates for C' are computed accordingly from the wiring predicates of the original circuit C and of $\mathsf{AUG_b}$. Finally, the AND testing that all low-degree tests is also constructible in a straightforward manner. The constructibility of C' follows.

Complexity of C'. The augmented construction increases the size of each layer of C by a $\mathrm{poly}(S)$ factor, and its depth by a $O(\log(S))$ factor. The final AND computation has $O(\log T)$ depth and $T \cdot \mathrm{poly}(S)$ size. We conclude that $|C'| = O(|C| \cdot T \cdot \mathrm{poly}(S))$.

4.4.2 Proof of Claim 4.4 (Augmented Soundness)

In this section we prove that the augmented circuit transformation described in Section 4.4.1 satisfies the properties in Claim 4.4. Properties 1 and 2 follow by construction. We focus on proving Property 3 (augmented soundness).

Let $\{C_k\}$ be a circuit ensemble, take $\{C'_k\}$ to be the augmented circuit ensemble, as described in Section 4.4.1. Let Assign be a Q-local assignment generator, as in Definition 4.2, for the augmented circuit ensemble $\{C'_k\}$ on a sequence of inputs $\{x_n\}_{n\in\mathbb{N}}$, where $Q(n) = O(\log^6 n)$. Fix a large enough security parameter $n \in \mathbb{N}$, the instance $x = x_n$ of length $k = k(n)$, the circuit $C = C_k$ of depth T and width S, and the augmented circuit $C' = C'_k$.

We prove that $C'(x) = 1$. This also implies $C(x) = 1$ (because C, C' are functionally equivalent).

Setup and notations. Take $Q = Q(n)$, and let $\mathbb{G}, \mathbb{H}$ and m be as defined in the construction of C'. Let $\mathbf{b}_x$ be the assignment to the wires of C' defined by the computation of $C'(x)$. For a set W of wires in C', s.t. $|W| \leq Q$, let $\mathsf{R}(W)$ be a random sample from the distribution $\mathsf{R}_{n,\mathbf{w}}$ (where $\mathbf{w} \in [|C'|]^Q$ is a lexicographically ordered vector containing the wires in W). For a set of wires W and for a sample $\mathbf{b} \leftarrow \mathsf{R}(W)$ we say that $\mathbf{b}$ is *correct* on a subset $W' \subseteq W$ of wires, if the assigned values $\mathbf{b}|'_W$ are consistent with the computation $C'(x)$. We denote this event by $\mathsf{CR}(\mathbf{b}, W')$.

Recall that we viewed the augmented circuit as an arithmetic circuit over $\mathbb{G}$ (see Section 4.4.1). We use this view in the soundness proof below. For a wire labeled L in the arithmetic circuit (carrying an element $g \in \mathbb{G}$), we use $W(\mathsf{L})$ to denote the set of $\log|\mathbb{G}|$ *"boolean"* wires in C' (carrying bits that encode the field element g). For a set of ("arithmetic") wires B, let $W(B)$ denote the corresponding set $\bigcup_{\mathsf{L}\in B} W(\mathsf{L})$ of ("boolean") wires in C'. For a sample $\mathbf{b} \leftarrow \mathsf{R}(W(B))$, and for $\mathsf{L} \in B$, denote by $\mathbf{b}(\mathsf{L})$ the element in $\mathbb{G}$ that is encoded by the assignment to the wires in $W(\mathsf{L})$.

The random variables Z_ℓ and "good" wires. For each layer $\ell \in [T]$ of the original circuit C, we keep track of c random variables. These random variables are sampled by picking $c = \log^2(n)$ independent and uniformly random points in the LDE of layer ℓ. We denote these random variables by $Z_\ell = (\mathbf{z}_{\ell,1}, \ldots, \mathbf{z}_{\ell,c})$, where each $\mathbf{z}_{\ell,i}$ is independently random in $\mathbb{G}^m$. These random variables will be used in the analysis of the augmented circuit construction. In the soundness proof, we consider (and analyze) various events defined over these random variables. We emphasize that the randomness is always taken over the choice of these points in Z_ℓ (unless we explicitly note otherwise).

Let B_ℓ be the set of labels $\{(\ell, \mathsf{LDE}, \mathbf{z}_{\ell,i})\}_{i\in[c]}$ carrying the field elements in the chosen coordinates of layer ℓ's LDE. As defined above, $W(B_\ell)$ is the set of (boolean) wires in C' carrying the bits encoding these field elements. We use $|W(B_\ell)| = (\log|\mathbb{G}| \cdot c)$ to denote the size of this set, and we abuse notation by using this even before the set B_ℓ is sampled (because its size is fixed in advance). Let $\mathbf{b}$ be an assignment to the chosen points in the LDE, sampled from $\mathsf{R}(W(B_\ell))$. Let CR_ℓ be the event $\mathsf{CR}(\mathbf{b}, W(B_\ell))$, indicating that the assignment in $\mathbf{b}$ is correct on $W(B_\ell)$.

Let W be a set of wires such that $(|W| + |W(B_\ell)|) \leq Q$. For a layer $\ell \in [T]$ we say that W is p-good for layer ℓ if:

$$\Pr_{\mathbf{b}\leftarrow\mathsf{R}(W\cup W(B_\ell))}[\neg\mathsf{CR}(\mathbf{b}, W) \mid \mathsf{CR}(\mathbf{b}, W(B_\ell))] \leq p \ .$$

That is, conditioned on a choice of B_ℓ, on the sampling of the assignment $\mathbf{b}$ from $\mathsf{R}(W \cup W(B_\ell))$, and on $\mathbf{b}$ being correct on $W(B_\ell)$, the assignment $\mathbf{b}$ is also correct on W with all but probability (at most) p. When the layer ℓ is clear from the context we simply say that W is p-good. If W is $\mu(n)$-good for some negligible function μ, we simply say that W is good.

Proof overview. We follow the outline of the proof in [KRR14]. The main step is showing that, for every layer $\ell \in [T]$, the event CR_ℓ occurs with overwhelming probability. The proof is by induction on the layer. The base case of the input layer is proved in Claim 4.12. In the inductive step (Claim 4.13), we prove that that for every layer $\ell \in [T-1]$, the probability of $\mathsf{CR}_{\ell+1}$ is negligibly close to the probability of CR_ℓ (this is sufficient because the number of layers is polynomial). We conclude the proof by showing that if in the output layer CR_ℓ occurs with overwhelming probability, then $C'(x) = 1$ (Claim 4.14).

In the proof of the inductive step, we show that, for any layer ℓ and *any* point $\mathbf{a} \in \mathbb{G}^m$ in the LDE of layer ℓ, the (set of wires representing) point $\mathbf{a}$ is "good for layer ℓ". That is, when we read the assignment to $\mathbf{a}$ together with the assignments to Z_ℓ, if the assignments to Z_ℓ were correct, then with overwhelming probability the values read in $\mathbf{a}$ are also correct.

The intuition for this is through a mental experiment. Suppose that we read more points, indeed suppose that we chose c random lines passing through $\mathbf{a}$, and read all the $(c \cdot |\mathbb{G}|)$ points on all these lines. In that scenario, with overwhelming probability, the restriction of the values read to the lines would be low-degree polynomials, because of the LDTs that the augmented circuit runs on all such lines: by local consistency, if we read the wires performing this test, the test should pass, and by no-signalling, the test should still pass even when we don't read the wires performing the LDT. Moreover, if the value read for $\mathbf{a}$ is not correct, then all these low-degree polynomials are not equivalent to the correct (low-degree) polynomials, and so the read polynomials must be far from the correct polynomials (by the Schwartz-Zippel Lemma).Picking the points Z_ℓ to be uniformly random points, one point on each (random) line, the probability that they are all correct is $\exp(-c)$, which is negligible. Thus, in this mental experiment (where we read many random lines), the probability that the points in Z_ℓ are all correct but $\mathbf{a}$ is not correct, is negligible. By the no-signalling property, this probability is maintained even when we only read the points in Z_ℓ, and don't read the remainder of the lines (note that indeed in both cases the distribution of points in Z_ℓ is uniformly random). Note that this required reading the values of $(c \cdot |\mathbb{G}|)$ points, or

$$(c \cdot |\mathbb{G}|) \cdot \log |\mathbb{G}| = O(\log^2(n) \cdot \log^2(S) \cdot \log\log(S)) = O(\log^4(n) \cdot \log\log(n)) \leq Q$$

wires of C' (carrying boolean values).

The above argument shows that, conditioned on the assignment being correct on Z^ℓ, it is correct on any point $\mathbf{a}$ in the ℓ-th layer with probability close to 1. We emphasize that this negligible failure probability (conditioned on CR_ℓ) does not grow with the layer (or with the probability the CR_ℓ doesn't occur). To complete the inductive step, we show that conditioned on CR_ℓ, we also have that any point in $Z^{\ell+1}$ is correct with all but negligible probability. This uses local consistency, which holds for every gate in the computation of layer $(\ell+1)$'s LDE from the LDE of layer ℓ. Local consistency guarantees that, for each gate in this computation, the probability of an error in the output wire (conditioned on CR_ℓ), is at most the sum of error probabilities in the input wires (conditioned on CR_ℓ). Since the computation of the LDE has only logarithmic depth, we get that the accumulated error probability for each point in $Z^{\ell+1}$ (conditioned on CR_ℓ) remains negligible. Taking a union bound over all points in $Z^{\ell+1}$, we get that they are all correct with all but negligible probability (conditioned on CR_ℓ). The inductive step follows.

Full proof. We start by proving a useful claim, showing that a good set of wires remains good when read together with other wires.

Claim 4.10. *Let $\ell \in [T]$ be a layer such that $\Pr[\mathsf{CR}_\ell] \geq 0.9$. Let W be a set such that $|W \cup W(B_\ell)| \leq Q$, and let $W' \subseteq W$. Let:*

$$\mathbf{b} \leftarrow \mathsf{R}(W \cup W(B_\ell)) \quad , \quad \mathbf{b}' \leftarrow \mathsf{R}(W' \cup W(B_\ell)) \ .$$

Then:

$$\left|\Pr[\neg\mathsf{CR}(\mathbf{b}, W') \mid \mathsf{CR}(\mathbf{b}, W(B_\ell))] - \Pr[\neg\mathsf{CR}(\mathbf{b}', W') \mid \mathsf{CR}(\mathbf{b}', W(B_\ell))]\right| \leq \mathrm{negl}(n) \ .$$

Proof. Let a, b, a', b' denote the probabilities:

$$\begin{aligned} a &= \Pr[\neg\mathsf{CR}(b, W') \wedge \mathsf{CR}(b, W(B_\ell))] \ , \\ b &= \Pr[\mathsf{CR}(b, W(B_\ell))] \ , \\ a' &= \Pr[\neg\mathsf{CR}(b', W') \wedge \mathsf{CR}(b', W(B_\ell))] \ , \\ b' &= \Pr[\mathsf{CR}(b', W(B_\ell))] \ . \end{aligned}$$

Since the event CR_ℓ occurs with probability at least 0.9 it follows from the no-signaling property of Assign that $b \geq 0.9 - \mathrm{negl}(n)$ and $b' \geq 0.9 - \mathrm{negl}(n)$. By the no-signaling property of of Assign, we have that $|a - a'| \leq \mathrm{negl}(n)$ and $|b - b'| \leq \mathrm{negl}(n)$. Overall we have that:

$$\left|\frac{a}{b} - \frac{a'}{b'}\right| = \frac{|ab' - a'b|}{bb'} = \frac{|ab' - a'b' + a'b' - a'b|}{bb'} \leq \frac{b'\,|a - a'| + a'\,|b' - b|}{bb'} \leq \mathrm{negl}(n) \ .$$

The claim follows, because:

$$\begin{aligned} \frac{a}{b} &= \Pr[\neg\mathsf{CR}(\mathbf{b}, W') \mid \mathsf{CR}(\mathbf{b}, W(B_\ell))] \ , \\ \frac{a'}{b'} &= \Pr[\neg\mathsf{CR}(\mathbf{b}', W') \mid \mathsf{CR}(\mathbf{b}', W(B_\ell))] \ , \end{aligned}$$

□

Corollary 4.11.

1. *For a layer $\ell \in [T]$ such that $\Pr[\mathsf{CR}_\ell] \geq 0.9$, if a set W is good, then every set $W' \subseteq W$ is also good.*

2. *For any sets W, W' such that $|W \cup W'| \leq Q$, if W and W' are both p-good then $W \cup W'$ is $(2p + \mathrm{negl}(n))$-good.*

Main claims. The statement of Claim 4.4 follows from the following claims:

Claim 4.12 (Induction base). *For the input layer we have that $\Pr[\mathsf{CR}_1] \geq 1 - \mathrm{negl}(n)$.*

Claim 4.13 (Induction step). *For a level $\ell \in [T-1]$, such that $\Pr[\mathsf{CR}_\ell] \geq 0.9$, we have that:*

$$\Pr[\mathsf{CR}_{\ell+1}] \geq \Pr[\mathsf{CR}_\ell] - \mathrm{negl}(n) \ .$$

Claim 4.14. *If in the output layer $\Pr[\mathsf{CR}_T] \geq 1 - \mathrm{negl}(n)$ then $C'(x) = 1$.*

We defer the proofs of Claims 4.12 and 4.14 (see below), and begin with Claim 4.13:

Proof of Claim 4.13 (induction step). Fix a layer $\ell \in [T-1]$ such that $\Pr[\mathsf{CR}_\ell] \geq 0.9$. The proof will relies on the following claims:

Claim 4.15. *For every $\mathbf{a} \in \mathbb{G}^m$, the set $W((\ell, \mathsf{LDE}, \mathbf{a}))$ is good.*

Claim 4.16. *For every $\mathbf{a} \in \mathbb{G}^m$, the set $W((\ell+1, \mathsf{LDE}, \mathbf{a}))$ is good.*

Before proving the claims we use them to complete the proof of Claim 4.13. By Claim 4.16, for every $i \in [c]$, the wire set $W((\ell+1, \mathsf{LDE}, \mathbf{z}_{\ell+1,i}))$ is good. Since $c = \log^2 n$, it follows from Corollary 4.11 that the set $W(B_{\ell+1})$ is good. That is, conditioned on CR_ℓ, we have that $\mathsf{CR}_{\ell+1}$ occurs with all but negligible probability. This implies that the probability of $\mathsf{CR}_{\ell+1}$ is at most negligibly smaller than that of CR_ℓ. To see this, take $\mathbf{b} \leftarrow \mathsf{R}(W(B_\ell \cup B_{\ell+1}))$. Let α, β denote the probabilities:

$$\alpha = \Pr[\mathsf{CR}(\mathbf{b}, W(B_\ell))] \quad , \quad \beta = \Pr[\mathsf{CR}(\mathbf{b}, W(B_{\ell+1}))] \ .$$

By the no-signaling property of Assign, we have that:

$$|\Pr[\mathsf{CR}_\ell] - \alpha| \leq \mathrm{negl}(n) \quad , \quad |\Pr[\mathsf{CR}_{\ell+1}] - \beta| \leq \mathrm{negl}(n) \ .$$

Therefore, to prove the claim, it is sufficient to prove that $\beta \geq \alpha - \mathrm{negl}(n)$. Since the set $W(B_{\ell+1})$ is good we have that:

$$\Pr[\mathsf{CR}(\mathbf{b}, W(B_{\ell+1})) | \mathsf{CR}(\mathbf{b}, W(B_\ell))] \geq 1 - \mathrm{negl}(n) \ .$$

Therefore:

$$\begin{aligned}
\beta &\geq \Pr[\mathsf{CR}(\mathbf{b}, W(B_\ell))] \cdot \Pr[\mathsf{CR}(\mathbf{b}, W(B_{\ell+1})) | \mathsf{CR}(\mathbf{b}, W(B_\ell))] \\
&\geq \alpha \cdot \Pr[\mathsf{CR}(\mathbf{b}, W(B_{\ell+1})) | \mathsf{CR}(\mathbf{b}, W(B_\ell))] \\
&\geq \alpha \cdot (1 - \mathrm{negl}(n)) \\
&\geq \alpha - \mathrm{negl}(n),
\end{aligned}$$

as required. To complete the proof of Claim 4.13, it remains to prove Claims 4.15 and 4.16.

Proof of Claim 4.15. Let:

$$\mathbf{b} \leftarrow \mathsf{R}(W(B_\ell) \cup W((\ell, \mathsf{LDE}, \mathbf{a}))) \ .$$

Since the event CR_ℓ occurs with probability at least 0.9, it follows from the no-signaling property that $\mathsf{CR}(\mathbf{b}, W(B_\ell)$ occurs with probability at least $0.9 - \mathrm{negl}$. Therefore, it suffices to prove that:

$$\Pr\left[\neg\mathsf{CR}(\mathbf{b}, W((\ell, \mathsf{LDE}, \mathbf{a}))) \wedge \mathsf{CR}(\mathbf{b}, W(B_\ell))\right] \leq \mathrm{negl}(n) \ . \tag{33}$$

For every line $L : \mathbb{G} \to \mathbb{G}^m$, let $B(L)$ be the set $\{(\ell, \mathsf{LDE}, L(t))\}_{t \in \mathbb{G}}$ of wire labels computing the LDE of level ℓ on the line L. We start by proving that for every line, with overwhelming probability the valued assigned to the line correspond to a low degree polynomial:

Claim 4.17. *For every line $L : \mathbb{G} \to \mathbb{G}^m$, let $\mathbf{b} \leftarrow \mathsf{R}(W(B(L)))$. Except with probability $\mathrm{negl}(n)$ the function $\gamma : \mathbb{G} \to \mathbb{G}$ given by:*

$$\gamma(\zeta) = \mathbf{b}((\ell, \mathsf{LDE}, L(\zeta))) \ ,$$

is a polynomial of degree at most $m \cdot |\mathbb{H}|$.

Proof. Let L be defined by $\mathbf{a}, \mathbf{b} \in \mathbb{G}^m$ and let B_{LDT} be the set of wire labels in the sub-circuit computing $(\ell, \mathsf{LDT}, \mathbf{a}, \mathbf{b})$. Recall that the augmented circuit computes the AND of all the LDT sub-circuits' outputs, using a tree of ANDs. On this tree, consider the path from the root to the sub-circuit $(\ell, \mathsf{LDT}, \mathbf{a}, \mathbf{b})$. The length of this path is $O(\log(T) + \log(S)) = O(\log n)$. Let W_{AND} be the set of all $O(\log n)$ wires connected to one of the AND gates on this path. Let W_L be the set:

$$W_L = W(B(L)) \cup W(B_{\mathsf{LDT}}) \cup W_{\mathsf{AND}} \ .$$

Note that $|W_L| = O(\log^6(n)) \leq Q$. Let $\mathbf{b}' \leftarrow \mathsf{R}(W_L)$ and let $\gamma' : \mathbb{G} \to \mathbb{G}$ be the function:

$$\gamma'(\zeta) = \mathbf{b}'((\ell, \mathsf{LDE}, L(\zeta))) \ .$$

By the everywhere local-consistency property of Assign, and by the construction of C', it follows that except with probability $\mathrm{negl}(n)$, the low degree test for the line L passes. Therefore, γ' is a polynomial of degree $m \cdot |\mathbb{H}|$. We obtained γ' by querying all the wires in the LDT and on the path from the LDT to the AND of LDTs (in addition to the points on the line L), as opposed to γ, which is obtained when we *only queried the points on the line L.* Still, by the no-signaling property of Assign, the probability that γ is a polynomial of degree $m \cdot |\mathbb{H}|$ is also $\mathrm{negl}(n)$. □

For every $i \in [c]$ let $\mathbf{z}_i = \mathbf{z}_{\ell,i}$ and let L_i be random line that passes though $\mathbf{a}$ and $\mathbf{z}_i$. Let $t_i \in \mathbb{G}$ be such that $L_i(t_i) = \mathbf{z}_i$. Note that since $\mathbf{z}_i$ is random, t_i is random even conditioned on L_i. Let $B_{\mathbf{a}}$ be the set of labels of all the lines L_i:

$$B' = \bigcup_{i \in [c]} B(L_i) \ .$$

(Note that $B_\ell \subset B'$). Observe that $|W(B')| = c \cdot |\mathbb{G}| = O(\log^4 n) \leq Q$. Let $\mathbf{b}' \leftarrow \mathsf{R}(W(B'))$, and recall that $\mathbf{b}_x$ denotes the correct assignment defined by the computation $C'(x)$. For every $i \in [c]$ let $\gamma_i, \gamma_i' : \mathbb{G} \to \mathbb{G}$ be the functions:

$$\gamma_i(\zeta) = \mathbf{b}_x((\ell, \mathsf{LDE}, L_i(\zeta))) \quad , \quad \gamma_i'(\zeta) = \mathbf{b}'((\ell, \mathsf{LDE}, L_i(\zeta))) \ .$$

Conditioned on the event $\neg\mathsf{CR}(\mathbf{b}', W((\ell, \mathsf{LDE}, \mathbf{a})))$ we have that for all $i \in [c]$, $\gamma_i \not\equiv \gamma_i'$. By the no-signaling property of Assign, by Claim 4.17 and by the Union bound, we have that with overwhelming probability, γ_i' is of degree $m \cdot |\mathbb{H}|$ for every $i \in [c]$. Also, by construction γ_i is *always* of low degree. We conclude that $\gamma_i \not\equiv \gamma_i'$, and γ_i, γ_i' are of degree at most $m \cdot |\mathbb{H}|$. By the Schwartz-Zippel Lemma, for each $i \in [c]$, with probability at least $1 - \frac{m \cdot |\mathbb{H}|}{|\mathbb{G}|} > 0.5$ over the choice of t_i, we have that $\gamma_i(t_i) \neq \gamma_i'(t_i)$, and $\mathbf{b}'$ is *not correct on* $\mathbf{z}_i$. It follows that:

$$\Pr\left[\neg\mathsf{CR}(\mathbf{b}', W((\ell, \mathsf{LDE}, \mathbf{a}))) \wedge \mathsf{CR}(\mathbf{b}', W(B_\ell))\right] \leq 2^{-c} + \mathrm{negl}(n) = \mathrm{negl}(n) \ .$$

Equation (33) follows from the the no-signaling property of Assign. □

Proof of Claim 4.16. We prove that for every $w \in W((\ell + 1, \mathsf{LDE}, \mathbf{a}))$, $\{w\}$ is good. Since $|W(\ell + 1, \mathsf{LDE}, \mathbf{a})| = \log |\mathbb{G}| = O(\log \log n)$, the claim follows using Corollary 4.11.

For every $w \in W((\ell + 1, \mathsf{LDE}, \mathbf{a}))$, let D be the sub-circuit of C', computing the low-degree extension of layer $(\ell + 1)$, whose output wire is w. Every input wire w_{in} of D is in $W((\ell, \mathsf{LDE}, \mathbf{a}'))$ for some $\mathbf{a}' \in \mathbb{H}^m \subset \mathbb{G}^m$. By Claim 4.15, every input wire w_{in} of D is good.

Consider an AND gate in D (the case of NOT gates is handled similarly). Suppose the gate has input wires w_1, w_2 and output wire w_3, such that $\{w_1\}, \{w_2\}$ are both p-good for some p. Let $\mathbf{b} \leftarrow \mathsf{R}(\{w_1, w_2, w_3\} \cup W(B_\ell))$. By Claim 4.10, since $\{w_1\}, \{w_2\}$ are both p-good we have that:

$$\forall i \in \{1, 2\} : \ \Pr[\neg\mathsf{CR}(\mathbf{b}, \{w_i\}) \mid \mathsf{CR}(\mathbf{b}, W(B_\ell))] \leq p + \mathrm{negl}(n) \ .$$

It follows that:

$$\Pr[\neg\mathsf{CR}(\mathbf{b}, \{w_1\}) \vee \neg\mathsf{CR}(\mathbf{b}, \{w_2\}) \mid \mathsf{CR}(\mathbf{b}, W(B_\ell))] \leq 2p + \mathrm{negl}(n) \ .$$

By the everywhere local-consistency property of Assign, we conclude that except with probability $\mathrm{negl}(n)$, when $\mathbf{b}$ is correct on $\{w_1, w_2\}$ it is also correct on $\{w_3\}$. Therefore:

$$\Pr[\neg\mathsf{CR}(\mathbf{b}, \{w_3\}) \mid \mathsf{CR}(\mathbf{b}, W(B_\ell))] \leq 2p\mathrm{negl}(n) \ ,$$

Where we emphasize that the $\mathrm{negl}(n)$ factor above is fixed, depending on the security parameter, and does not depend on the initial error probability p (which may itself also be negligible). It follows from Claim 4.10 that $\{w_3\}$ is $(2p + \mathrm{negl}(n))$-good. By induction on the depth of the sub-circuit D we have that $\{w\}$ is μ-good, where:

$$\mu = 2^{O(\log(n))} \cdot \mathrm{negl}(n) = \mathrm{negl}(n).$$

□

□

Proof of Claim 4.12 (induction base). We prove that for every $w \in W(B_1)$:

$$\Pr_{\mathbf{b} \leftarrow \mathsf{R}(\{w\})}[\neg \mathsf{CR}(\mathbf{b}, \{w\})] \leq \text{negl}(n) \ .$$

Since $|W(B_1)| = c \cdot \log |\mathbb{G}| = O(\log^2(n) \cdot \log \log n)$, the claim follows from the no-signaling property of Assign together with a Union bound.

For every $w \in W(B_1)$, let D be a sub-circuit of C' computing the LDE of layer 1 (the input), whose output wire is w, and such that every input wire w_{in} of D is an input wire of C'. Note that the depth of D is $O(\log(S)) = O(\log(n))$. It follows from the everywhere local-consistency property of Assign that for every input wire w_{in} of D:

$$\Pr_{\mathbf{b} \leftarrow \mathsf{R}(\{w_{in}\})}[\neg \mathsf{CR}(\mathbf{b}, \{w_{in}\})] \leq \text{negl}(n) \ .$$

Consider an AND gate in D (the case of NOT gates is handled similarly) with input wires w_1, w_2 and output wire w_3, such that:

$$\forall i \in \{1, 2\} : \Pr_{\mathbf{b} \leftarrow \mathsf{R}(\{w_i\})}[\neg \mathsf{CR}(\mathbf{b}, \{w_i\})] \leq p \ .$$

Let $\mathbf{b} \leftarrow \mathsf{R}(\{w_1, w_2, w_3\} \cup W(B_\ell))$. By the no signaling property of Assign we have:

$$\forall i \in \{1, 2\} : \Pr[\neg \mathsf{CR}(\mathbf{b}, \{w_i\})] \leq p + \text{negl}(n) \ .$$

Therefore:

$$\Pr[\neg \mathsf{CR}(\mathbf{b}, \{w_1\}) \vee \neg \mathsf{CR}(\mathbf{b}, \{w_2\})] \leq 2p + \text{negl}(n) \ .$$

By the everywhere local-consistency property of Assign, with all but negligible probability, when $\mathbf{b}$ is correct on $\{w_1, w_2\}$ it is also correct on $\{w_3\}$. Therefore:

$$\Pr[\neg \mathsf{CR}(\mathbf{b}, \{w_3\})] \leq 2p + \text{negl}(n) \ .$$

By the no-signaling property of Assign:

$$\Pr_{\mathbf{b} \leftarrow \mathsf{R}(\{w_3\})}[\neg \mathsf{CR}(\mathbf{b}, \{w_3\})] \leq 2p + \text{negl}(n) \ .$$

By induction on the depth of the sub-circuit D we conclude that:

$$\Pr_{\mathbf{b} \leftarrow \mathsf{R}(\{w\})}[\neg \mathsf{CR}(\mathbf{b}, \{w\})] \leq 2^{O(\log(n))} \cdot \text{negl}(n) = \text{negl}(n) \ .$$

□

Proof of Claim 4.14. Let w be the output wire of C' and let $\mathbf{a} \in \mathbb{H}^m$ such that $w \in W((T, \mathsf{LDE}, \mathbf{a}))$ is a wire in the LDE of the circuit's output layer. Since $\Pr[\mathsf{CR}_T] \geq 0.9$, it follows from Claim 4.15 and from Corollary 4.11 that $\{w\}$ is good for layer T. That is, for $\mathbf{b} \leftarrow \mathsf{R}(\{w\} \cup W(B_T))$:

$$\Pr\left[\neg \mathsf{CR}(\mathbf{b}, W) \mid \mathsf{CR}(\mathbf{b}, W(B_T))\right] \leq \text{negl}(n) \ .$$

Since $\Pr[\mathsf{CR}_T] \geq 1 - \text{negl}(n)$ it follows from the no-signaling property of Assign that:

$$\Pr\left[\mathsf{CR}(\mathbf{b}, W(B_T))\right] \geq 1 - \text{negl}(n) \ .$$

Therefore:

$$\Pr\left[\mathsf{CR}(\mathbf{b}, W)\right] \geq 1 - \text{negl}(n) \ .$$

By the everywhere local-consistency property of Assign we have that except with negligible probability, $\mathbf{b}$ assigns the value 1 to w. We conclude that the correct value of w is 1, that is, $C'(x) = 1$. □

5 Non-Interactive Arguments for Bounded-Depth Computations

In this section we present our delegation protocol for bounded depth computations. We start by constructing the non-interactive Sum-Check protocol as described in Section 2.2 (see Section 5.1). We also construct a 2-to-1 sub-protocol (see Section 5.2). Using these sub-protocols as building blocks, we constructs a "Bare-Bones" delegation protocol (Section 5.3). In Section 5.4 we instantiate the Bare-Bones delegation protocol in different ways to get delegation protocols for bounded depth computations.

The Sum-Check, 2-to-1, and Bare-Bones sub-protocols are all used as building blocks in the final delegation protocol for bounded-depth computations, as was the case in [GKR08]. We use graded encodings to make each of the sub-protocols (and the resulting delegation protocol) non-interactive, and to compose them in a non-interactive manner. This complicates matters, because we need to run the sub-protocols on encoded inputs, obtained as the outputs of other sub-protocols. Thus, we construct the sub-protocols to allow (part of) their input to be encoded under a GE. In composing sub-protocols to form the full delegation protocol, we take care to ensure that these encodings' levels are "low enough" to allow the (honest) prover to perform the computations required by the protocol.

We also note that sometimes the encoded inputs used by the (honest) prover might be quite large, larger even than the desired running time of the verifier. In these cases, the verifier won't ever need to access this large encoded portion of the input, it will only be used by the (honest) prover.

5.1 Non-Interactive Sum-Check Sub-Protocol

This section describes a non-interactive sum-check sub-protocol $(\mathsf{Gen}_{SC}, \mathsf{P}_{SC}, \mathsf{V}_{SC})$. Let $n \in \mathbb{N}$ be a security parameter. Let $\delta = \delta(n), \kappa = \kappa(n)$ be polynomials and let:

$$\mathsf{pp} \leftarrow \mathsf{InstGen}(1^n, \kappa, \delta) \ \ ,$$

be parameters for an asymmetric graded encoding scheme. Let $\mathbb{Z}_p$ be the underlying field of the public parameters pp. The sum-check sub-protocol depends on the public parameters pp, and the soundness of protocol holds only for honestly generated parameters. Let:

$$f : \mathbb{Z}_p^{\ell_1} \times \mathbb{Z}_p^{\ell_2} \to \mathbb{Z}_p \ \ ,$$

be a multi-variate low degree polynomial (represented as an arithmetic circuit).

Informally, the sum-check sub-protocol takes as input encodings of $\mathbf{z} \in \mathbb{Z}_p^{\ell_1}, c \in \mathbb{Z}_p$ and outputs encodings of $\mathbf{w}^{\mathsf{out}} \in \mathbb{Z}_p^{\ell_2}, c^{\mathsf{out}} \in \mathbb{Z}_p$. When the verifier accepts, she is guaranteed that if the input specified a false underlying claim, then the output also specifies a false underlying claim:

$$c \neq \sum_{\mathbf{w} \in \{0,1\}^{\ell_2}} f(\mathbf{z}, \mathbf{w}) \ \ \Rightarrow \ \ c^{\mathsf{out}} \neq f(\mathbf{z}, \mathbf{w}^{\mathsf{out}})$$

The verifier in the sum-check sub-protocol will only require as input the encoding of c, but not encodings of $\mathbf{z}$. Therefore, the complexity of the verifier is independent of ℓ_1 ($\mathbf{z}$ does, however, appear in the protocol's "output claim". This output claim is implicit to the verifier).

The inputs and outputs of the sum-check sub-protocol are encoded under the public parameters pp. The parameter m_1 describes the number of dimensions used in the levels of input encodings. The parameter m_2 describes the number of additional dimensions used internally within the protocol. In particular, the level of each input encoding is zero in all coordinates higher than m_1, and the levels of each encoding computed internally and of each output encoding is zero in all coordinates higher than $m_1 + m_2$. Note that it will always be the case that $m_2 = \ell_2$.

5.1.1 Interface

In this section we define the interfaces of the protocol and formulate its soundness property.

The challenge generator Gen_{SC}.

Input:

1. A security parameter 1^n and parameters m_1, m_2.
2. Public parameters for an asymmetric graded encoding:

$$\mathsf{pp} \leftarrow \mathsf{InstGen}(1^n, \kappa, \delta) \ .$$

We require that $\kappa \geq m_1 + m_2$.

Output: A challenge CRS.

Complexity: The running time of Gen_{SC} is $\mathrm{poly}(n)$. The challenge CRS is of length $m_2 \cdot \mathrm{poly}(n)$.

The prover P_{SC}.

Input:

1. The public parameters pp.
2. The challenge CRS.
3. An arithmetic circuit f taking $\ell_1 + \ell_2$ inputs. f should contain only addition subtraction and multiplication gates and use only the constants $\{0, 1\}$. f should be of individual degree at most δ in every variable. We also require that $\ell_2 = m_2$ (for the Gen_{SC} procedure's input m_2).
4. An encoded input:

$$x = \left(\left\{ [z_i]_{\mathbf{v}_i} \right\}_{i \in [\ell_1]}, [c]_{\mathbf{v}} \right) \ ,$$

where:

$$\left\{ [z_i]_{\mathbf{v}_i} \right\}_{i \in [\ell_1]} \ ,$$

is a valid partial input for f and $\mathbf{v}$ is the output level of f for this partial input (recall the definitions of a valid encoded input and output level in Section 3.1). Additionally we require that for every $i \in [\ell_1]$, $\mathbf{v}_i$ is zero in all coordinates higher than m_1.

Output:

1. An encoded output:

$$y = \left(\mathbf{w}^{\mathsf{out}} = \left\{ \left[w_i^{\mathsf{out}} \right]_{\mathbf{e}_{m_1+i}} \right\}_{i \in [\ell_2]}, \left[c^{\mathsf{out}} \right]_{\mathbf{v}+\mathbf{u}} \right) \ ,$$

where:

$$\mathbf{u} = \sum_{j \in [\ell_2]} \delta \cdot \mathbf{e}_{m_1+j} \ .$$

2. A proof Π for the fact that:

$$c \neq \sum_{\mathbf{w} \in \{0,1\}^{\ell_2}} f(\mathbf{z}, \mathbf{w}) \quad \Rightarrow \quad c^{\mathsf{out}} \neq f(\mathbf{z}, \mathbf{w}^{\mathsf{out}}) \ .$$

Complexity: The running time of P_{SC} is $\mathrm{poly}(n, |f|, 2^{\ell_2})$. The proof Π is of length $\delta \cdot \ell_2 \cdot \mathrm{poly}(n)$.

The verifier V_{SC}.

Input:

1. The public parameters pp.
2. The challenge CRS.
3. Part of the encoded input $x_{\mathsf{V}} = [c]_{\mathbf{v}}$.
4. The encoded output y.
5. The proof Π.

Output: 1 if the proof is accepted and 0 otherwise.

Complexity: The running time of V_{SC} is $\delta \cdot \ell_2 \cdot \mathrm{poly}(n)$.

5.1.2 Completeness and Soundness

The protocol's completeness and soundness properties are stated in Claims 5.1 and 5.2 below. Completeness guarantees that if the input statement (defined by the encoded input and the function f) is correct, then the output statement (defined by the encoded output) is also correct. Soundness guarantees that if the input statement is incorrect, then w.h.p. either the verifier rejects, or the output statement is also incorrect (we prove adaptive soundness for adversarially chosen input statements).

Claim 5.1 (Completeness)**.** *The sum-check protocol described in Section 5.1.3 satisfies the following property. Let f be an arithmetic circuit, and let $\kappa, \delta, \ell_1, \ell_2, m_1, m_2$ be parameters as defined in the protocol interface (Section 5.1.1). Let* $\mathsf{InpSamp}$ *be an algorithm that gets as input public parameters* pp *and outputs an encoded input:*

$$x = \left(\left\{ [z_i]_{\mathbf{v}_i} \right\}_{i \in [\ell_1]}, [c]_{\mathbf{v}} \right) ,$$

as defined in the protocol interface and such that:

$$c = \sum_{\mathbf{w} \in \{0,1\}^{\ell_2}} f(z_1, \dots, z_{\ell_1}, \mathbf{w}) ,$$

where the arithmetic, as well as the evaluation of f are over the field $\mathbb{Z}_p$ underlying pp. *Then:*

$$\Pr \left[\begin{array}{l} \mathsf{pp} \leftarrow \mathsf{InstGen}(1^n, \kappa, \delta); \\ \mathsf{CRS} \leftarrow \mathsf{Gen}_{SC}(1^n, \mathsf{pp}, (m_2, m_1)); \\ x \leftarrow \mathsf{InpSamp}(\mathsf{pp}); \\ y, \Pi \leftarrow \mathsf{P}_{SC}(\mathsf{pp}, \mathsf{CRS}, f, x); \\ 1 \leftarrow \mathsf{V}_{SC}(\mathsf{pp}, \mathsf{CRS}, x_{\mathsf{V}}, y, \Pi); \\ \mathsf{OUT} \end{array} \right] = 1 ,$$

where:

$$x_{\mathsf{V}} = [c]_{\mathbf{v}} \quad , \quad y = \left(\left\{ \left[w_i^{\mathsf{out}} \right]_{\mathbf{e}_{m_1+i}} \right\}_{i \in [\ell_2]}, \left[c^{\mathsf{out}} \right]_{\mathbf{v}+\mathbf{u}} \right) ,$$

and OUT *is the event that:*

$$c^{\mathsf{out}} = f(z_1, \dots, z_{\ell_1}, w_1^{\mathsf{out}}, \dots, w_{\ell_2}^{\mathsf{out}}) ,$$

where the evaluation of f is over the field $\mathbb{Z}_p$ underlying pp.

Claim 5.2 (Adaptive Soundness)**.** *Assuming an asymmetric graded encoding scheme satisfying Assumption 3.10, the sum-check protocol described in Section 5.1.3 satisfies the following property. For every pair of poly-size circuits* $\mathsf{P}_1^*, \mathsf{P}_2^*$ *there exists a negligible function* μ *such that for every* $n \in \mathbb{N}$*:*

$$\Pr\left[\begin{array}{l} \mathsf{pp} \leftarrow \mathsf{InstGen}(1^n, \kappa, \delta); \\ \mathsf{CRS} \leftarrow \mathsf{Gen}_{SC}(1^n, \mathsf{pp}, (m_2, m_1)); \\ f, x \leftarrow \mathsf{P}_1^*(\mathsf{pp}, \mathsf{CRS}); \\ y^*, \Pi^* \leftarrow \mathsf{P}_2^*(\mathsf{pp}, \mathsf{CRS}, f, x); \\ 1 \leftarrow \mathsf{V}_{SC}(\mathsf{pp}, \mathsf{CRS}, x_{\mathsf{V}}, y^*, \Pi^*); \\ \mathsf{CHEAT} \end{array}\right] \leq \mu(n) \enspace ,$$

where:

$$x = \left(\left\{[z_i]_{\mathbf{v}_i}\right\}_{i \in [\ell_1]}, [c]_{\mathbf{v}}\right) \quad , \quad x_{\mathsf{V}} = [c]_{\mathbf{v}} \quad , \quad y^* = \left(\left\{[w_i^*]_{\mathbf{e}_{m_1+i}}\right\}_{i \in [\ell_2]}, [c^*]_{\mathbf{v}+\mathbf{u}}\right) \enspace ,$$

f *is as required in the protocol interface (Section 5.1.1), and* CHEAT *is the event that:*

$$\left(c \neq \sum_{\mathbf{w} \in \{0,1\}^{\ell_2}} f(z_1, \ldots, z_{\ell_1}, \mathbf{w})\right) \quad \wedge \quad c^* = f(z_1, \ldots, z_{\ell_1}, w_1^*, \ldots, w_{\ell_2}^*) \enspace ,$$

where the arithmetic, as well as the evaluation of f *are over the field* $\mathbb{Z}_p$ *underlying* pp.

Remark 5.3. Similarly to Remark 3.11, observe that the events OUT and CHEAT can be tested in polynomial time using the public parameters pp, and using the operations $\mathsf{Add}, \mathsf{Sub}, \mathsf{Mult}, \mathsf{isZero}$ on the input and output encodings x, y. For simplicity, we define the events as conditions on the values:

$$\mathbf{z}, \mathbf{w}^*, c, c^* \enspace ,$$

(which cannot be efficiently computed from the input and output encoding), rather than explicitly describing the efficient testing procedure.

5.1.3 Construction

In this section we specify the strategies Gen_{SC}, P_{SC} and V_{SC}.

The challenge generator Gen_{SC}**.** Given the public parameters pp for the graded encoding scheme and the parameters m_1, m_2, Gen_{SC} uses the operation Samp to obtain the set of encodings:

$$\mathsf{CRS} = \left\{[r_i]_{\mathbf{e}_{m_1+i}}\right\}_{i \in [\ell_2]} \enspace ,$$

where $r_1, \ldots, r_{\ell_2}$ are independent and uniform in $\mathbb{Z}_p$ (pp's underlying field). Gen_{SC} outputs CRS.

The prover P_{SC}**.** Recall that the prover's input includes the public parameters pp, the challenge CRS, the arithmetic circuit f, and the encoded input x where:

$$\mathsf{CRS} = \left\{[r_i]_{\mathbf{e}_{m_1+i}}\right\}_{i \in [\ell_2]} \enspace .$$
$$x = \left(\left\{[z_i]_{\mathbf{v}_i}\right\}_{i \in [\ell_1]}, [c]_{\mathbf{v}}\right) \enspace ,$$

In what follows the arithmetic is over $\mathbb{Z}_p$.

For every $i \in [\ell_2]$, for the elements $z_1, \dots, z_{\ell_1}$ defined by the input encoding, and for the elements $r_1, \dots, r_{i-1}$ defined by the encodings in CRS, let g_i be the univariate polynomial:

$$g_i(\xi) = \sum_{w_{i+1},\dots,w_{\ell_2} \in \{0,1\}} f(z_1, \dots, z_{\ell_1}, r_1, \dots, r_{i-1}, \xi, w_{i+1}, \dots, w_{\ell_2}) \ ,$$

and let $\{g_{i,j}\}_{j\in[0,\delta]}$ be the coefficients of g_i:

$$g_i(\xi) = \sum_{j=0}^{\delta} g_{i,j} \cdot \xi^j. \tag{34}$$

The proof consists of the encodings of these coefficients:

$$\Pi = \left\{ [g_{i,j}]_{\mathbf{v}+\mathbf{u}_{i-1}} \right\}_{i\in[\ell_2], j\in[0,\delta]} \ ,$$

where $\mathbf{u}_0$ is the all-zero level and for $i \in [\ell_2]$:

$$\mathbf{u}_i = \sum_{i'\in[i]} \delta \cdot \mathbf{e}_{m_1+i'} \ .$$

See below on how the prover computes these encoding. The output encodings are:

$$y = \left(\left\{ \left[w_i^{\mathsf{out}}\right]_{\mathbf{e}_{m_1+i}} \right\}_{i\in[\ell_2]}, \left[c^{\mathsf{out}}\right]_{\mathbf{v}+\mathbf{u}} \right) \ ,$$

where:

$$w_i^{\mathsf{out}} = r_i \quad , \quad c^{\mathsf{out}} = f(\mathbf{z}, w_1^{\mathsf{out}}, \dots, w_{\ell_2}^{\mathsf{out}}) \ .$$

P_{SC} outputs the encodings y and the proof Π.

To compute the encodings of $\{g_{i,j}\}$ in the proof Π, we use arithmetic circuits $\{f_{i,j}\}_{i\in[\ell_2],j\in[0,\delta]}$ taking $\ell_1 + \ell_2 - 1$ inputs, such that:

$$f(\mathbf{z}, \mathbf{w}) = \sum_{j\in[0,\delta]} f_{i,j}(\mathbf{z}, w_1, \dots w_{i-1}, w_{i+1}, \dots, w_{\ell_2}) \cdot w_i^j \ .$$

Note that a circuit for $f_{i,j}$ can be efficiently computed from f and that the encoded input for f is valid also for $f_{i,j}$, see Section 3.7. Now observe that:

$$g_{i,j} = \sum_{w_{i+1},\dots,w_{\ell_2} \in \{0,1\}} f_{i,j}(z_1, \dots, z_{\ell_1}, r_1, \dots, r_{i-1}, w_{i+1}, \dots, w_{\ell_2}) \ . \tag{35}$$

And indeed Equation (34) holds. Following Equation (35), and using the circuit for $f_{i,j}$ and the operations $\mathsf{Add}, \mathsf{Sub}, \mathsf{Mult}$ on the input encodings and the encoding in CRS

The verifier V_{SC}. Recall that the verifier's input includes:

1. The public parameters pp.
2. The challenge:
$$\mathsf{CRS} = \left\{ [r_i]_{\mathbf{e}_{m_1+i}} \right\}_{i\in[\ell_2]} \ .$$
3. Part of the encoded input $x_{\mathsf{V}} = [c]_{\mathbf{v}}$.

4. The encoded output:
$$y = \left(\left\{ \left[w_i^{\mathsf{out}} \right]_{\mathbf{e}_{m_1+i}} \right\}_{i \in [\ell_2]}, \left[c^{\mathsf{out}} \right]_{\mathbf{v}+\mathbf{u}} \right) \quad ,$$
5. The proof:
$$\Pi = \left\{ [g_{i,j}]_{\mathbf{v}+\mathbf{u}_{i-1}} \right\}_{i \in [\ell_2], j \in [0,\delta]} \quad .$$

Following Equation (34), and using the operations $\mathsf{Add}, \mathsf{Sub}, \mathsf{Mult}$ on the input encodings, V_{SC} obtains:
$$\left\{ [g_i(r_i)]_{\mathbf{v}+\mathbf{u}_i}, [g_i(0) + g_i(1)]_{\mathbf{v}+\mathbf{u}_{i-1}} \right\}_{i \in [\ell_2]} \quad ,$$

Using the operations $\mathsf{Sub}, \mathsf{isZero}$ on the above encodings, the encodings in CRS, and the input and output encodings, V_{SC} tests that:
$$c = g_1(0) + g_1(1) \tag{36}$$
$$\forall i \in [\ell_2 - 1] : \quad g_i(r_i) = g_{i+1}(0) + g_{i+1}(1) \tag{37}$$
$$\forall i \in [\ell_2] : \quad r_i = w_i^{\mathsf{out}} \tag{38}$$
$$g_{\ell_2}(r_{\ell_2}) = c^{\mathsf{out}} \tag{39}$$

V_{SC} accepts iff all tests pass.

5.1.4 Soundness: Proof of Claim 5.2

Proof. Assume towards contradiction that there exists a pair of poly-size circuits $(\mathsf{P}_1^*, \mathsf{P}_2^*)$ and a polynomial p such that for infinitely many values of $n \in \mathbb{N}$, the event CHEAT occurs with probability at least $\frac{1}{p(n)}$. Fix such n. We show that there exist an adversary Adv_i that wins the security game in Assumption 3.10 with probability $\frac{1}{p(n) \cdot \ell_2}$. For every $i \in [\ell_2]$, consider the adversary Adv_i that gets public parameters pp and an encoding $[r]_{\mathbf{e}_{m_1+i}}$ of a random element r. Adv_i starts by emulating the procedure $\mathsf{Gen}_{SC}(1^n, \mathsf{pp}, (m_2, m_1))$ except that instead of sampling the encoding $[r_i]_{\mathbf{e}_{m_1+i}}$ on its own, it uses $[r]_{\mathbf{e}_{m_1+i}}$. That is, Adv_i obtains the encodings:
$$\mathsf{CRS} = \left\{ [r_1]_{\mathbf{e}_{m_1+1}}, \dots, [r_{i-1}]_{\mathbf{e}_{m_1+i-1}}, [r]_{\mathbf{e}_{m_1+i}}, [r_{i+1}]_{\mathbf{e}_{m_1+i+1}}, \dots, [r_{m_2}]_{\mathbf{e}_{m_1+\ell_2}} \right\} \quad .$$
Note that the CRS is distributed exactly as the output of $\mathsf{Gen}_{SC}(1^n, \mathsf{pp}, (m_2, m_1))$. Next, Adv_i executes $\mathsf{P}_1^*(\mathsf{pp}, \mathsf{CRS})$ and obtains:
$$f \quad , \quad x = \left(\{ [z_i]_{\mathbf{v}_i} \}_{i \in [\ell_1]}, [c]_{\mathbf{v}} \right) \quad .$$
Adv_i then executes $\mathsf{P}_2^*(\mathsf{pp}, \mathsf{CRS}, f, x)$ and obtains output encodings and a proof:
$$y^* = \left(\left\{ [w_i^*]_{\mathbf{e}_{m_1+i}} \right\}_{i \in [\ell_2]}, [c^*]_{\mathbf{v}+\mathbf{u}} \right) \quad , \quad \Pi^* = \left\{ \left[g_{i,j}^* \right]_{\mathbf{v}+\mathbf{u}_{i-1}} \right\}_{i \in [\ell_2], j \in [0,\delta]} \quad .$$
Adv_i also executes the honest prover $\mathsf{P}_{SC}(\mathsf{pp}, \mathsf{CRS}, f, x)$ and obtains output encodings and a proof:
$$y = \left(\left\{ \left[w_i^{\mathsf{out}} \right]_{\mathbf{e}_{m_1+i}} \right\}_{i \in [\ell_2]}, \left[c^{\mathsf{out}} \right]_{\mathbf{v}+\mathbf{u}} \right) \quad , \quad \Pi = \left\{ [g_{i,j}]_{\mathbf{v}+\mathbf{u}_{i-1}} \right\}_{i \in [\ell_2], j \in [0,\delta]} \quad .$$
Next, Adv_i uses the operation $\mathsf{Sub}, \mathsf{isZero}$ to find the maximal pair of indexes $(i', j) \in [\ell_2] \times [0, \delta]$ in lexicographical order such that $g_{i',j}^* \neq g_{i',j}$. If no such indexes are found or if $i' \neq i$, Adv_i aborts. Otherwise, Adv_i uses the operations $\mathsf{Add}, \mathsf{Sub}, \mathsf{Mult}$ on the encodings in CRS, Π, Π^* to obtain:
$$[\beta]_{\mathbf{v}+\mathbf{u}_{i-1}} \quad , \quad [\gamma]_{\mathbf{v}+\mathbf{u}_{i-1}+(j-1)\cdot\mathbf{e}_{m_1+i}} \quad ,$$

where

$$\beta = g^*_{i,j} - g_{i,j} \quad , \quad \gamma = \sum_{j'<j} (g_{i,j'} - g^*_{i,j'}) r_i^{j'} \ ,$$

and indeed:

$$\begin{aligned}(\mathbf{v} + \mathbf{u}_{i-1})\,[m_1 + i] &= 0 \\ (\mathbf{v} + \mathbf{u}_{i-1} + (j-1)\cdot \mathbf{e}_{m_1+i})\,[m_1+i] &< j\end{aligned}$$

Finally Adv_i outputs $[\beta]_{\mathbf{v}+\mathbf{u}_{i-1}}, [\gamma]_{\mathbf{v}+\mathbf{u}_{i-1}+(j-1)\cdot\mathbf{e}_{m_1+i}}$.

Next we prove that for some $i \in [\ell_2]$, Adv_i wins the security game in Assumption 3.10 with probability $\frac{1}{p(n)\cdot \ell_2}$. For $i \in [\ell_2]$ let:

$$g^*_i(\xi) = \sum_{j\in[0,\delta]} g^*_{i,j} \cdot \xi^j \quad , \quad g_i(\xi) = \sum_{j\in[0,\delta]} g_{i,j} \cdot \xi^j \ ,$$

and let E_i be that event that $g^*_i \not\equiv g_i$. Conditioned on the event CHEAT we have that:

$$\sum_{\mathbf{w}\in\{0,1\}^{\ell_2}} f(\mathbf{z}, \mathbf{w}) = g_1(0) + g_1(1) \neq c \ .$$

Conditioned on the event CHEAT, we also have that and that the verifier's Test 36 passes and

$$g^*_1(0) + g^*_1(1) = c \ .$$

Therefore, $g^*_1 \not\equiv g_1$, and the event E_1 must hold. Thus, Adv_i finds some (maximal) indices (i', j) as described above.

Let G_i be the event that CHEAT occurs and that $i' = i$ (that is, Adv_i does not abort). Conditioned on G_i, $i = i'$ is the maximal index such that E_i holds. Therefore, in the case that $i < \ell_2$ we have that $\neg E_{i+1}$ holds, and since the verifier's Test 37 passes we have that:

$$g^*_i(r_i) = g^*_{i+1}(0) + g^*_{i+1}(1) = g_{i+1}(0) + g_{i+1}(1) = g_i(r_i) \ .$$

In the case that $i = \ell_2$ we have that conditioned on G_i:

$$\begin{aligned} g^*_{\ell_2}(r_{\ell_2}) &= c^* && (40)\\ &= f(\mathbf{z}, \mathbf{w}^*) && (41)\\ &= f(\mathbf{z}, r_1, \dots, r_{\ell_2}) && (42)\\ &= f(\mathbf{z}, \mathbf{w}^{\mathsf{out}}) = c^{\mathsf{out}} = g_{\ell_2}(r_{\ell_2}) \ , && (43)\end{aligned}$$

where (40) follows from the fact that the verifier's Test 39 passes, (41) follows from the event CHEAT, (42) follows from the fact that the verifier's Test 38 passes, and (43) follows from the definition of the honest prove P_{SC}. Overall, for every $i \in [\ell_2]$, we conclude that if we condition on G_i, we have that:

$$g^*_i \not\equiv g_i \quad , \quad g^*_i(r_i) = g_i(r_i) \ .$$

Thus

$$0 = \sum_{j'\in[0,\delta]} (g_{i,j'} - g^*_{i,j'}) r_i^{j'} .$$

Since j is the maximal index such that $g^*_{i,j} \neq g_{i,j}$, we have that:

$$0 = \sum_{j'\in[0,j]} (g_{i,j'} - g^*_{i,j'}) r_i^{j'} \implies (g^*_{i,j} - g_{i,j}) r_i^j = \sum_{j'<j} (g_{i,j'} - g^*_{i,j'}) r_i^{j'} ,$$

and we conclude that conditioned on G_i:

$$\beta \cdot r_i^j = (g^*_{i,j} - g_{i,j}) r_i^j = \sum_{j' < j} (g_{i,j'} - g^*_{i,j'}) r_i^{j'} = \gamma \ ,$$

and $\beta \neq 0$. Hence, Adv_i wins the security game.

It is left to show that (for some $i \in [\ell_2]$) the event G_i occurs with noticeable probability. Recall that the challenge CRS generated by Adv_i is distributed exactly as the output of $\mathsf{Gen}_{SC}(1^n, \mathsf{pp}, (m_2, m_1))$, and by our assumption, the event CHEAT occurs in the execution of Adv_i with probability at least $\frac{1}{p(n)}$. Conditioned on CHEAT, we know that E_1 occurs, and thus $G_{i'}$ occurs for some $i' \in [\ell_2]$. Since the view of P_1^* and P_2^* and the event CHEAT are all independent of i, we conclude that conditioned on CHEAT, there exists $i \in [\ell_2]$ such that the event G occurs with probability at least $\frac{1}{p(n) \cdot \ell_2}$. □

5.2 Non-Interactive 2-to-1 Sub-Protocol

This section describes a non-interactive 2-to-1 sub-protocol $(\mathsf{Gen}_{2\to1}, \mathsf{P}_{2\to1}, \mathsf{V}_{2\to1})$. Let $n \in \mathbb{N}$ be a security parameter. Let $\delta = \delta(n), \kappa = \kappa(n)$ be polynomials and let:

$$\mathsf{pp} \leftarrow \mathsf{InstGen}(1^n, \kappa, \delta) \ ,$$

be parameters for an asymmetric graded encoding scheme. Let $\mathbb{Z}_p$ be the underlying field for the public parameters pp. The 2-to-1 sub-protocol will depend on the public parameters pp, and the soundness of protocol will only hold for honestly generated parameters. Let:

$$f : \mathbb{Z}_p^{\ell_1} \times \mathbb{Z}_p^{\ell_2} \to \mathbb{Z}_p \ ,$$

be a multi-linear polynomial (represented as an arithmetic circuit).

Informally, the 2-to-1 sub-protocol takes as input encodings of $\mathbf{z} \in \mathbb{Z}_p^{\ell_1}, \mathbf{w}^1, \mathbf{w}^2 \in \mathbb{Z}_p^{\ell_2}, c^1, c^2 \in \mathbb{Z}_p$ and outputs encodings of $\mathbf{w}^{\mathsf{out}} \in \mathbb{Z}_p^{\ell}, c^{\mathsf{out}} \in \mathbb{Z}_p$. When the verifier accepts, she is guaranteed that with overwhelming probability it is the case that:

$$\left(c^1 \neq f(\mathbf{z}, \mathbf{w}^1) \quad \vee \quad c^2 \neq f(\mathbf{z}, \mathbf{w}^2)\right) \quad \Rightarrow \quad c^{\mathsf{out}} \neq f(\mathbf{z}, \mathbf{w}^{\mathsf{out}}) \ .$$

The verifier in the 2-to-1 sub-protocol will only require as input the encodings of $\mathbf{w}^1, \mathbf{w}^2, c^1, c^2$, but not the encodings of $\mathbf{z}$. Therefore, the complexity of the verifier is independent of ℓ_1.

The inputs and outputs of the 2-to-1 sub-protocol are encoded under the public parameters pp. The parameter m describes the number of dimensions used in the levels of input encodings. The $(m+1)$-th coordinate is used internally by the protocol and its outputs. In particular, the level of each input encoding is zero in all coordinates higher than m.

5.2.1 Interface

In this section we define the interfaces of the protocol and formulate its soundness property.

The challenge generator $\mathsf{Gen}_{2\to1}$**.**

Input:

1. A security parameter 1^n and a parameter m.
2. Public parameters for an asymmetric graded encoding:

$$\mathsf{pp} \leftarrow \mathsf{InstGen}(1^n, \kappa, \delta) \ .$$

We require that $\kappa \geq m+1$ and that $\delta \geq \ell_2$.

Output: A challenge CRS.

Complexity: $\mathsf{Gen}_{2\to 1}$'s running time and output length are $\mathrm{poly}(n)$.

The prover $\mathsf{P}_{2\to 1}$.

Input:

1. The public parameters pp.
2. The challenge CRS.
3. A multi-linear arithmetic circuit f taking $\ell_1+\ell_2$ inputs. f should contain only addition subtraction and multiplication gates and use only the constants $\{0,1\}$.
4. An encoded input:

$$x = \left(\left\{[z_i]_{\mathbf{v}_i}\right\}_{i\in[\ell_1]}, \left\{\left[w_i^1\right]_{\mathbf{v}_{\ell_1+i}}, \left[w_i^2\right]_{\mathbf{v}_{\ell_1+i}}\right\}_{i\in[\ell_2]}, \left[c^1\right]_{\mathbf{v}}, \left[c^2\right]_{\mathbf{v}}\right) ,$$

where:

$$\left\{[z_i]_{\mathbf{v}_i}\right\}_{i\in[\ell_1]} \cup \left\{\left[w_i^1\right]_{\mathbf{v}_{\ell_1+i}}\right\}_{i\in[\ell_2]} \quad , \quad \left\{[z_i]_{\mathbf{v}_i}\right\}_{i\in[\ell_1]} \cup \left\{\left[w_i^2\right]_{\mathbf{v}_{\ell_1+i}}\right\}_{i\in[\ell_2]} ,$$

are a valid partial input for f and $\mathbf{v}$ is the output level of f for these partial inputs (recall the definitions of a valid encoded input and output level in Section 3.1). Additionally we require that for every $i \in [\ell_1+\ell_2]$, $\mathbf{v}_i$ is zero in all coordinates higher than m.

Output:

1. An encoded output:

$$y = \left(\left\{\left[w_i^{\mathsf{out}}\right]_{\mathbf{v}_{(\ell_1+i)}+\mathbf{e}_{m+1}}\right\}_{i\in[\ell_2]}, \left[c^{\mathsf{out}}\right]_{\mathbf{v}+\ell_2\cdot\mathbf{e}_{m+1}}\right) .$$

2. A proof Π for the fact that:

$$\left(c^1 \neq f(\mathbf{z},\mathbf{w}^1) \quad \vee \quad c^2 \neq f(\mathbf{z},\mathbf{w}^2)\right) \quad \Rightarrow \quad c^{\mathsf{out}} \neq f(\mathbf{z},\mathbf{w}^{\mathsf{out}}) .$$

Complexity: The running time of $\mathsf{P}_{2\to 1}$ is $\mathrm{poly}(n,\ell,|f|)$, the lengths of y and Π are $\ell\cdot\mathrm{poly}(n)$.

The verifier $\mathsf{V}_{2\to 1}$.

Input:

1. The public parameters pp.
2. The challenge CRS.
3. Part of the encoded input x:

$$x_{\mathsf{V}} = \left(\left\{\left[w_i^1\right]_{\mathbf{v}_{\ell_1+i}}, \left[w_i^2\right]_{\mathbf{v}_{\ell_1+i}}\right\}_{i\in[\ell_2]}, \left[c^1\right]_{\mathbf{v}}, \left[c^2\right]_{\mathbf{v}}\right) ,$$

4. The encoded output y.
5. The proof Π.

Output: 1 if the proof is accepted and 0 otherwise.

Complexity: The running time of $\mathsf{V}_{2\to 1}$ is $\mathrm{poly}(n,\ell_2)$.

5.2.2 Completeness and Soundness

The protocol's completeness and soundness properties are stated in Claims 5.4 and 5.5 below. Completeness guarantees that if the input statement (defined by the encoded inputs and the function f) is correct, then the output statement (defined by the encoded output) is also correct. Soundness guarantees that if the input statement is incorrect, then w.h.p. either the verifier rejects, or the output statement is also incorrect (we prove adaptive soundness for adversarially chosen input statements).

Claim 5.4 (Completeness)**.** *The 2-to-1 protocol described in Section 5.2.3 satisfies the following property. Let f be an arithmetic circuit, and let $\kappa, \delta, \ell_1, \ell_2, m$ be parameters as defined in the protocol interface (Section 5.2.1). Let* $\mathsf{InpSamp}$ *be an algorithm that gets as input public parameters* pp *and outputs an encoded input:*

$$x = \left(\left\{ [z_i]_{\mathbf{v}_i} \right\}_{i \in [\ell_1]}, \left\{ \left[w_i^1\right]_{\mathbf{v}_{\ell_1+i}}, \left[w_i^2\right]_{\mathbf{v}_{\ell_1+i}} \right\}_{i \in [\ell_2]}, \left[c^1\right]_{\mathbf{v}}, \left[c^2\right]_{\mathbf{v}} \right) \quad ,$$

as defined in the protocol interface and such that:

$$c^1 = f(z_1, \ldots, z_{\ell_1}, w_1^1, \ldots, w_{\ell_2}^1) \quad , \quad c^2 = f(z_1, \ldots, z_{\ell_1}, w_1^2, \ldots, w_{\ell_2}^2) \quad ,$$

where the evaluation of f are over the field $\mathbb{Z}_p$ underlying pp*.*

Then:

$$\Pr \left[\begin{array}{l} \mathsf{pp} \leftarrow \mathsf{InstGen}(1^n, \kappa, \delta); \\ \mathsf{CRS} \leftarrow \mathsf{Gen}_{2\to1}(1^n, \mathsf{pp}, m); \\ x \leftarrow \mathsf{InpSamp}(\mathsf{pp}); \\ y, \Pi \leftarrow \mathsf{P}_{2\to1}(\mathsf{pp}, \mathsf{CRS}, f, x); \\ 1 \leftarrow \mathsf{V}_{2\to1}(\mathsf{pp}, \mathsf{CRS}, x_{\mathsf{V}}, y, \Pi); \\ \mathsf{OUT} \end{array} \right] = 1 \quad ,$$

where x_{V} is as defined above, where:

$$y = \left(\left\{ \left[w_i^{\mathsf{out}}\right]_{\mathbf{v}_i + \mathbf{e}_{m+1}} \right\}_{i \in [\ell_2]}, \left[c^{\mathsf{out}}\right]_{\mathbf{v} + \ell \cdot \mathbf{e}_{m+1}} \right) \quad ,$$

and where OUT *is the event that:*

$$c^{\mathsf{out}} = f(z_1, \ldots, z_{\ell_1}, w_1^{\mathsf{out}}, \ldots, w_{\ell_2}^{\mathsf{out}}) \quad ,$$

where f is evaluated over the underlying field $\mathbb{Z}_p$ of pp*.*

The adaptive soundness of the 2-to-1 sub-protocol says that if the statement defined by the encoded input is incorrect then either the verifier rejects, or the statement defined by the encoded output is also incorrect.

Claim 5.5 (Adaptive Soundness)**.** *Assuming an asymmetric graded encoding scheme satisfying Assumption 3.10, the 2-to-1 protocol described in Section 5.2.3 satisfies the following property. for every pair of poly-size circuits $\mathsf{P}_1^*, \mathsf{P}_2^*$ there exists a negligible function μ such that for every $n \in \mathbb{N}$:*

$$\Pr \left[\begin{array}{l} \mathsf{pp} \leftarrow \mathsf{InstGen}(1^n, \kappa, \delta); \\ \mathsf{CRS} \leftarrow \mathsf{Gen}_{2\to1}(1^n, \mathsf{pp}, m); \\ f, x \leftarrow \mathsf{P}_1^*(\mathsf{pp}, \mathsf{CRS}); \\ y^*, \Pi^* \leftarrow \mathsf{P}_2^*(\mathsf{pp}, \mathsf{CRS}, f, x); \\ 1 \leftarrow \mathsf{V}_{2\to1}(\mathsf{pp}, \mathsf{CRS}, x_{\mathsf{V}}, y^*, \Pi^*); \\ \mathsf{CHEAT} \end{array} \right] \leq \mu(n) \quad ,$$

where x_{V} is as defined above, where:

$$x = \left(\left\{ [z_i]_{\mathbf{v}_i} \right\}_{i \in [\ell_1]}, \left\{ \left[w_i^1\right]_{\mathbf{v}_{\ell_1+i}}, \left[w_i^2\right]_{\mathbf{v}_{\ell_1+i}} \right\}_{i \in [\ell_2]}, \left[c^1\right]_{\mathbf{v}}, \left[c^2\right]_{\mathbf{v}} \right) ,$$
$$y^* = \left(\left\{ [w_i^*]_{\mathbf{v}_{(\ell_1+i)}+\mathbf{e}_{m+1}} \right\}_{i \in [\ell_2]}, [c^*]_{\mathbf{v}+\ell_2 \cdot \mathbf{e}_{m+1}} \right) ,$$

f is as required in the protocol interface (Section 5.2.1), and where CHEAT *is the event that:*

$$\left(c^1 \neq f(\mathbf{z}, w_1^1, \ldots, w_{\ell_2}^1) \quad \vee \quad c^2 \neq f(\mathbf{z}, w_1^2, \ldots, w_{\ell_2}^2)\right) \quad \wedge \quad c^* = f(\mathbf{z}, w_1^*, \ldots, w_{\ell_2}^*) \ ,$$

where $\mathbf{z} = (z_1, \ldots, z_{\ell_1})$ and f is evaluated over the underlying field $\mathbb{Z}_p$ of pp.

Remark 5.6. Similarly to Remark 3.11, observe that the events OUT and CHEAT can be tested in polynomial time using the public parameters pp, and using the operations $\mathsf{Add}, \mathsf{Sub}, \mathsf{Mult}, \mathsf{isZero}$ on the input and output encodings x, y. For simplicity, we define these events asconditions on the values:

$$\mathbf{z}, \mathbf{w}^1, c^1, \mathbf{w}^2, c^2, \mathbf{w}^*, c^* \ ,$$

(which cannot be efficiently computed from the input and output encoding), rather than explicitly describing the efficient testing procedure.

5.2.3 Construction

In this section we specify the strategies $\mathsf{Gen}_{2\to1}, \mathsf{P}_{2\to1}$ and $\mathsf{V}_{2\to1}$.

The challenge generator $\mathsf{Gen}_{2\to1}$. Given the public parameters pp for the graded encoding scheme and the parameter m, $\mathsf{Gen}_{2\to1}$ uses the operation Samp to obtain the encoding:

$$\mathsf{CRS} = [t]_{\mathbf{e}_{m+1}} \ ,$$

where t is a uniform element in $\mathbb{Z}_p$ (the underlying field of pp). $\mathsf{Gen}_{2\to1}$ outputs CRS.

The prover $\mathsf{P}_{2\to1}$. Recall that the prover's input includes the public parameters pp, the challenge CRS, the arithmetic circuit f, and the encoded input x where:

$$\mathsf{CRS} = [t]_{\mathbf{e}_{m+1}} \ ,$$
$$x = \left(\left\{ [z_i]_{\mathbf{v}_i} \right\}_{i \in [\ell_1]}, \left\{ \left[w_i^1\right]_{\mathbf{v}_{\ell_1+i}}, \left[w_i^2\right]_{\mathbf{v}_{\ell_1+i}} \right\}_{i \in [\ell_2]}, \left[c^1\right]_{\mathbf{v}}, \left[c^2\right]_{\mathbf{v}} \right) .$$

Let $\vec{\gamma}(\xi)$ be the line:

$$\vec{\gamma}(\xi) = \xi \cdot \mathbf{w}^1 + (1 - \xi) \cdot \mathbf{w}^2 \ ,$$

that is, for every $i \in [\ell_2]$:

$$\vec{\gamma}(\xi)[i] = \xi \cdot w_i^1 + (1 - \xi) \cdot w_i^2 \ . \tag{44}$$

For the elements $z_1, \ldots, z_{\ell_1}$ defined by the input encoding, let $g(\xi)$ be the univariate polynomial:

$$g(\xi) = f(z_1, \ldots, z_{\ell_1}, \vec{\gamma}(\xi)) \ .$$

Since f is multi-linear, g is of degree ℓ_2. For every $j \in [0, \ell_2]$ let g_j be g's coefficients, i.e.:

$$g(\xi) = \sum_{j \in [0,\ell_2]} g_j \cdot \xi^j \ . \tag{45}$$

The proof consists of encodings of these coefficients:

$$\Pi = \left\{ [g_j]_{\mathbf{v}} \right\}_{j \in [0,\ell_2]} \ .$$

See below on how the prover computes these encodings. Using the operations $\mathsf{Add}, \mathsf{Sub}, \mathsf{Mult}$ on the proof encodings and the encoding in CRS, $\mathsf{P}_{2\to 1}$ obtains the output encodings:

$$y = \left(\left\{ \left[w_i^{\mathsf{out}} \right]_{\mathbf{v}_{(\ell_1+i)} + \mathbf{e}_{m+1}} \right\}_{i \in [\ell_2]}, \left[c^{\mathsf{out}} \right]_{\mathbf{v} + \ell_2 \cdot \mathbf{e}_{m+1}} \right) \ ,$$

where:

$$w_i^{\mathsf{out}} = \vec{\gamma}(t)[i] \quad , \quad c^{\mathsf{out}} = g(t) \ .$$

$\mathsf{P}_{2\to 1}$ outputs the proof Π and the output encoding y.

To compute the proof encodings $\left\{ [g_j]_{\mathbf{v}} \right\}$, we use arithmetic circuits $\{f_j\}_{j \in [0,\ell_2]}$ taking ℓ_1 inputs, such that:

$$f(z_1, \dots, z_{\ell_1}, \vec{\gamma}(\xi)) = \sum_{j \in [0,\ell_2]} f_j(z_1, \dots, z_{\ell_1}) \cdot \xi^j \ .$$

Note that a circuit for f_j can be efficiently computed from f and that the encoded input for f is valid also for f_j, see Section 3.7. Now, for every $j \in [0, \ell_2]$, using the circuit for f_j and the operations $\mathsf{Add}, \mathsf{Sub}, \mathsf{Mult}$ on the input encodings $\mathsf{P}_{2\to 1}$ obtains $[g_j]_{\mathbf{v}}$ where:

$$g_j = f_j(z_1, \dots, z_{\ell_1}) \ .$$

The verifier $\mathsf{V}_{2\to 1}$**.** Recall that the verifier's input includes:

1. The public parameters pp.
2. The challenge:
$$\mathsf{CRS} = [t]_{\mathbf{e}_{m+1}} \ .$$
3. Part of the encoded input:
$$x_{\mathsf{V}} = \left(\left\{ \left[w_i^1 \right]_{\mathbf{v}_{\ell_1+i}}, \left[w_i^2 \right]_{\mathbf{v}_{\ell_1+i}} \right\}_{i \in [\ell_2]}, \left[c^1 \right]_{\mathbf{v}}, \left[c^2 \right]_{\mathbf{v}} \right) \ .$$
4. The encoded output:
$$y = \left(\left\{ \left[w_i^{\mathsf{out}} \right]_{\mathbf{v}_{(\ell_1+i)} + \mathbf{e}_{m+1}} \right\}_{i \in [\ell_2]}, \left[c^{\mathsf{out}} \right]_{\mathbf{v} + \ell_2 \cdot \mathbf{e}_{m+1}} \right) \ .$$
5. The proof:
$$\Pi = \left\{ [g_j]_{\mathbf{v}} \right\}_{j \in [0,\ell_2]} \ .$$

Following Equations (44) and (45), and using the operations $\mathsf{Add}, \mathsf{Sub}, \mathsf{Mult}$ on the input encodings and on the proof, $\mathsf{V}_{2\to 1}$ obtains:

$$\left\{ [\vec{\gamma}(t)[i]]_{\mathbf{v}_i + \mathbf{e}_{m+1}} \quad , \quad [\vec{\gamma}(0)[i]]_{\mathbf{v}_i} \quad , \quad [\vec{\gamma}(1)[i]]_{\mathbf{v}_i} \right\}_{i \in [\ell]}$$
$$[g(t)]_{\mathbf{v} + \ell_2 \cdot \mathbf{e}_{m+1}} \quad , \quad [g(0)]_{\mathbf{v}} \quad , \quad [g(1)]_{\mathbf{v}} \ .$$

Using the operations $\mathsf{Sub}, \mathsf{isZero}$ on the above encodings, and the input encodings, $\mathsf{V}_{2\to 1}$ tests that:

$$\begin{array}{llll} \forall i \in [\ell] : & \vec{\gamma}(t)[i] = w_i^{\mathsf{out}} & \vec{\gamma}(0)[i] = w_i^2 & \vec{\gamma}(1)[i] = w_i^1 \\ & g(t) = c^{\mathsf{out}} & g(0) = c^2 & g(1) = c^1 \end{array}$$

$\mathsf{V}_{2\to 1}$ accepts iff all tests pass.

5.2.4 Proof of Claim 5.5

Proof. Assume towards contradiction that there exists a pair of poly-size circuits $(\mathsf{P}_1^*, \mathsf{P}_2^*)$ and a polynomial p such that for infinitely many values of $n \in \mathbb{N}$, the event CHEAT occurs with probability at least $\frac{1}{p(n)}$. Fix such n. We show that there exist an adversary Adv that wins the security game in Assumption 3.10 with probability $\frac{1}{p(n)}$. Consider the adversary Adv that gets public parameters pp and an encoding $[t]_{\mathbf{e}_{m+1}}$ of a random element t. Adv first obtains a the challenge:

$$\mathsf{CRS} = [t]_{\mathbf{e}_{m+1}} \quad .$$

Note that CRS is distributed exactly as the output of $\mathsf{Gen}_{2\to 1}(1^n, \mathsf{pp}, m)$. Next, Adv executes $\mathsf{P}_1^*(\mathsf{pp}, \mathsf{CRS})$ and obtains:

$$f \quad , \quad x = \left(\left\{ [z_i]_{\mathbf{v}_i} \right\}_{i\in[\ell_1]}, \left\{ [w_i^1]_{\mathbf{v}_{\ell_1+i}}, [w_i^2]_{\mathbf{v}_{\ell_1+i}} \right\}_{i\in[\ell_2]}, [c^1]_{\mathbf{v}}, [c^2]_{\mathbf{v}} \right) \quad .$$

Adv then executes $\mathsf{P}_2^*(\mathsf{pp}, \mathsf{CRS}, f, x)$ and obtains output encodings and a proof:

$$y^* = \left(\left\{ [w_i^*]_{\mathbf{v}_{(\ell_1+i)}+\mathbf{e}_{m+1}} \right\}_{i\in[\ell_2]}, [c^*]_{\mathbf{v}+\ell_2\cdot\mathbf{e}_{m+1}} \right) \quad , \quad \Pi = \left\{ [g_j^*]_{\mathbf{v}} \right\}_{j\in[0,\ell_2]} \quad .$$

Adv also executes the honest prover $\mathsf{P}_{2\to 1}(\mathsf{pp}, \mathsf{CRS}, f, x)$ and obtains output encodings and a proof:

$$y = \left(\left\{ [w_i^{\mathsf{out}}]_{\mathbf{v}_{(\ell_1+i)}+\mathbf{e}_{m+1}} \right\}_{i\in[\ell_2]}, [c^{\mathsf{out}}]_{\mathbf{v}+\ell_2\cdot\mathbf{e}_{m+1}} \right) \quad , \quad \Pi = \left\{ [g_j]_{\mathbf{v}} \right\}_{j\in[0,\ell_2]} \quad .$$

Next, Adv uses the operation $\mathsf{Sub}, \mathsf{isZero}$, finds the maximal index $j \in [0, \ell_2]$ such that $g_j^* \neq g_j$. If no such index is found, Adv aborts. Otherwise, Adv uses the operations $\mathsf{Add}, \mathsf{Sub}, \mathsf{Mult}$ on the encodings in CRS, Π, Π^* to obtain:

$$[\beta]_{\mathbf{v}} \quad , \quad [\gamma]_{\mathbf{v}+(j-1)\cdot\mathbf{e}_{m+1}} \quad ,$$

where:

$$\beta = g_j^* - g_j \quad , \quad \gamma = \sum_{j'<j} (g_{j'} - g_{j'}^*) t^{j'} \quad .$$

Note that indeed:

$$\begin{aligned} \mathbf{v}[m+1] &= 0 \\ (\mathbf{v} + (j-1)\cdot\mathbf{e}_{m+1})\,[m+1] &< j. \end{aligned}$$

Finally Adv outputs $[\beta]_{\mathbf{v}}, [\gamma]_{\mathbf{v}+(j-1)\cdot\mathbf{e}_{m+1}}$.

We prove that Adv_i wins the security game in Assumption 3.10 with probability $\frac{1}{p(n)}$. Let:

$$g^*(x) = \sum_{j\in[0,\ell_2]} g_j^* \cdot x^j \quad , \quad g(x) = \sum_{j\in[0,\ell_2]} g_j \cdot x^j \quad ,$$

and let E be that event that $g^*(x) \not\equiv g(x)$. Conditioned on the event CHEAT we have that:

$$g(1) = f(\mathbf{z}, \mathbf{w}^1) \neq c^1 \quad \vee \quad g(0) = f(\mathbf{z}, \mathbf{w}^2) \neq c^2 \quad .$$

Conditioned on the event CHEAT we also have that and that the verifier's tests pass and

$$g^*(1) = c^1 \quad \vee \quad g^*(0) = c^2 \quad .$$

Therefore the event E must hold and Adv must find an index j as described. Conditioned on the event CHEAT we also have that:

$$\begin{aligned} g^*(t) &= c^* && (46)\\ &= f(\mathbf{z}, \mathbf{w}^*) && (47)\\ &= f(\mathbf{z}, \vec{\gamma}(t)[1], \dots, \vec{\gamma}(t)[\ell_2]) && (48)\\ &= f(\mathbf{z}, \mathbf{w}^{\mathsf{out}}) = c^{\mathsf{out}} = g(t) \ , && (49) \end{aligned}$$

where (46) and (48) follow from the fact that the verifier's tests pass and (47) follows from the event CHEAT. Overall, we conclude that conditioned on CHEAT:

$$g^* \not\equiv g \quad \wedge \quad g^*(t) = g(t) \ .$$

Therefore, since j is the maximal index such that $g_j^* \neq g_j$, we have that conditioned on CHEAT:

$$\beta \cdot t^j = (g_j^* - g_j)t^j = \sum_{j' < j} (g_{j'} - g_{j'}^*)t^{j'} = \gamma \ ,$$

and $\beta \neq 0$. Hence, Adv wins the security game.

The challenge CRS generated by Adv is distributed exactly as the output of $\mathsf{Gen}_{2\to1}(1^n, \mathsf{pp}, m)$. Thus, if indeed the cheating provers can make the event CHEAT occur with probability $\frac{1}{p(n)}$, then Adv wins the security game in Assumption 3.10 with probability $\frac{1}{p(n)}$, contradicting the assumption. □

5.3 Non-Interactive Bare-Bones Delegation Protocol

This section describes a bare-bones delegation protocol $(\mathsf{Gen}_{BB}, \mathsf{P}_{BB}, \mathsf{V}_{BB})$. Let $n \in \mathbb{N}$ be a security parameter. Let $\delta = \delta(n), \kappa = \kappa(n)$ be polynomials and let:

$$\mathsf{pp} \leftarrow \mathsf{InstGen}(1^n, \kappa, \delta) \ ,$$

be parameters for an asymmetric graded encoding scheme. Let $\mathbb{Z}_p$ be the underlying field of the public parameters pp. The sum-check sub-protocol depends on the public parameters pp, and the soundness of protocol holds only for honestly generated parameters.

Let f be a layered fan-in 2, strongly ring-independent arithmetic circuit with k inputs. Let S be the size of f and D be the depth of f and take $m = \log S$. (For simplicity and without loss of generality we assume that S is a power of 2.) We index every gate of f by a string in $\{0,1\}^m$. We number the layers of f such that layer 0 is the output layer and layer D is the input layer. For every $i \in [D]$ let:

$$\mathsf{add}_i, \mathsf{sub}_i, \mathsf{mult}_i : \{0,1\}^{3m} \to \{0,1\} \ ,$$

be functions such that $\mathsf{add}_i(g_1, g_2, g_3) = 1$ if g_1 is a gate in layer $i-1$, g_2, g_3 are gates is in layer i and g_1 is adding the output of g_2 and g_3. Otherwise $\mathsf{add}_i(g_1, g_2, g_3) = 0$. sub_i and mult_i are defined similarly. Let δ' be a degree parameter and let $\widetilde{\mathsf{add}}_i, \widetilde{\mathsf{sub}}_i, \widetilde{\mathsf{mult}}_i$ be ring-independent arithmetic circuits of individual degree δ' computing the functions $\mathsf{add}_i, \mathsf{sub}_i, \mathsf{mult}_i$.

Informally, the bare-bones delegation protocol takes as input encodings of $\mathbf{x} = (x_1, \dots, x_k) \in \mathbb{Z}_p^k$, $c \in \mathbb{Z}_p$, and outputs encodings of ring elements:

$$\left\{\mathbf{z}_i, c_i^+, c_i^-, c_i^\times\right\}_{i \in [D]} \ ,$$

When the verifier accepts, she is guaranteed that with overwhelming probability it is the case that:

$$f(\mathbf{x}) \neq c \quad \Rightarrow \quad \left(\exists i \in [D] : \left(\widetilde{\mathsf{add}}_i(\mathbf{z}_i) \neq c_i^+\right) \vee \left(\widetilde{\mathsf{sub}}_i(\mathbf{z}_i) \neq c_i^-\right) \vee \left(\widetilde{\mathsf{mult}}_i(\mathbf{z}_i) \neq c_i^\times\right)\right) \ ,$$

The inputs and outputs of the bare-bones delegation protocol are encoded under the public parameters pp. The parameter ℓ describes the number of dimensions used in the levels of input encodings. The protocol will internally use $D \cdot m'$ additional dimensions where $m' = 3m + 1$. We require that the input encoding is valid for f (see Section 3.1) to ensure that the honest prover can evaluate f on the encoded input. In fact we will need to choose δ to be twice as larger as what would be required to encode the output level of f.

5.3.1 Interface

In this section we define the interfaces of the protocol and formulate its soundness property.

The challenge generator Gen_{BB}.

Input:

1. A security parameter 1^n and parameters δ', m, D, ℓ.
2. Public parameters for an asymmetric graded encoding:

$$\mathsf{pp} \leftarrow \mathsf{InstGen}(1^n, \kappa, \delta) \enspace ,$$

such that $\kappa \geq \ell + D \cdot m'$ (where $m' = 3m + 1$), $\delta \geq \delta' + 1$ and $\delta \geq m$.

Output: A challenge CRS.

Complexity: The running time of Gen_{BB} is $\mathrm{poly}(n, \delta', m)$. The length of CRS is $\mathrm{poly}(n, m, \delta', D)$.

The prover P_{BB}.

Input:

1. The public parameters pp.
2. The challenge CRS.
3. Ring-independent arithmetic circuits:

$$\left\{\widetilde{\mathsf{add}}_i, \widetilde{\mathsf{sub}}_i, \widetilde{\mathsf{mult}}_i\right\}_{i\in[D]} \enspace ,$$

each of degree δ, describing a layered, fan-in 2, strongly ring-independent arithmetic circuit f of size $S = 2^m$, depth D, and input length k. We assume throughout that these circuits are part of f's description and are all of size at most $\mathrm{poly}(S)$.

4. An encoded input:

$$x = \left(\left\{[x_i]_{\mathbf{v}_i}\right\}_{i\in[k]}, [c]_{\mathbf{v}}\right) \enspace ,$$

where $\left\{[x_i]_{\mathbf{v}_i}\right\}_{i\in[k]}$ is a valid partial input for f and $\mathbf{v}$ is the output level of f for this partial input (recall the definitions of a valid encoded input and output level in Section 3.1). Additionally we require that $\mathbf{v} \leq \frac{\delta}{2} \cdot \mathbf{s}_\ell$. That is, $\mathbf{v}$ is at most $\frac{\delta}{2}$ in its first ℓ coordinates and is zero in the remaining coordinates.

Output:

1. An encoded output:

$$y = \left(\left\{ [w_{i,j}]_{\mathbf{e}_{\ell+(i-1)\cdot m'+j}} \right\}_{i\in[D],j\in[m'-1]}, \left\{ \left[c_i^+\right]_{\delta'\cdot\mathbf{u}_i}, \left[c_i^-\right]_{\delta'\cdot\mathbf{u}_i}, \left[c_i^\times\right]_{\delta'\cdot\mathbf{u}_i} \right\}_{i\in[D]} \right) ,$$

where for every $i \in [D]$:

$$\mathbf{u}_i = \sum_{j\in[m'-1]} \mathbf{e}_{\ell+(i-1)\cdot m'+j} \ .$$

2. A proof Π for the fact that:

$$f(\mathbf{x}) \neq c \quad \Rightarrow \quad \left(\exists i \in [D] : \left(\widetilde{\mathsf{add}}_i(\mathbf{w}_i) \neq c_i^+\right) \vee \left(\widetilde{\mathsf{sub}}_i(\mathbf{w}_i) \neq c_i^-\right) \vee \left(\widetilde{\mathsf{mult}}_i(\mathbf{w}_i) \neq c_i^\times\right)\right) ,$$

where $\mathbf{x} = (x_1, \ldots, x_k)$ and for every $i \in [D]$, $\mathbf{w}_i = \left(w_{i,1}, \ldots, w_{i,m'-1}\right)$.

Complexity: The running time of P_{BB} is $\mathrm{poly}(n, 2^m, \delta')$. The length of Π is $\mathrm{poly}(n, m, \delta', D)$.

The verifier V_{BB}.

Input:

1. The public parameters pp.
2. The challenge CRS.
3. The encoded input x.
4. The encoded output y.
5. The proof Π.

Output: 1 if the proof is accepted and 0 otherwise.

Complexity: The running time of V_{BB} is $k \cdot \mathrm{poly}(n, \delta', m)$.

5.3.2 Completeness and Soundness

The protocol's completeness and soundness properties are stated in Claims 5.7 and 5.8 below. Completeness guarantees that if the input statement ($f(\mathbf{x}) = c$) is correct, then the output statement (defined by the encoded outputs) is also correct. Soundness guarantees that if the input statement is incorrect, then w.h.p. either the verifier rejects, or the output statement is also incorrect. We prove adaptive soundness for adversarially chosen f, $\mathbf{x}$ and c.

Claim 5.7 (Completeness)**.** *The bare-bones delegation protocol described in Section 5.3.3 satisfies the following property. Let f be an arithmetic circuit, and let $\kappa, \delta, \delta', m, D, \ell$ be parameters as defined in the protocol interface (Section 5.1.1). Let* InpSamp *be an algorithm that gets as input public parameters* pp *and outputs an encoded input:*

$$x = \left(\left\{ [x_i]_{\mathbf{v}_i} \right\}_{i\in[k]}, [c]_{\mathbf{v}} \right) ,$$

as defined in the protocol interface, and such that:

$$c = f(x_1, \ldots, x_k) \ ,$$

where the evaluation of f is over the field $\mathbb{Z}_p$ underlying pp. *Then:*

$$\Pr\left[\begin{array}{l}\mathsf{pp} \leftarrow \mathsf{InstGen}(1^n, \kappa, \delta);\\ \mathsf{CRS} \leftarrow \mathsf{Gen}_{BB}(1^n, \mathsf{pp}, (\delta', m, D, \ell));\\ x \leftarrow \mathsf{InpSamp}(\mathsf{pp});\\ y, \Pi \leftarrow \mathsf{P}_{BB}(\mathsf{pp}, \mathsf{CRS}, f, x);\\ 1 \leftarrow \mathsf{V}_{BB}(\mathsf{pp}, \mathsf{CRS}, x, y, \Pi);\\ \mathsf{OUT}\end{array}\right] = 1 \ ,$$

where

$$y = \left(\left\{[w_{i,j}]_{\mathbf{e}_{(i-1)\cdot m'+j}}\right\}_{i\in[D], j\in[m'-1]}, \left\{[c_i^+]_{\delta'\cdot\mathbf{u}_i}, [c_i^-]_{\delta'\cdot\mathbf{u}_i}, [c_i^\times]_{\delta'\cdot\mathbf{u}_i}\right\}_{i\in[D]}\right) \ ,$$

and OUT *is the event that:*

$$\forall i \in [D] : \left(\widetilde{\mathsf{add}}_i(\mathbf{w}_i) = c_i^+\right) \wedge \left(\widetilde{\mathsf{sub}}_i(\mathbf{w}_i) = c_i^-\right) \wedge \left(\widetilde{\mathsf{mult}}_i(\mathbf{w}_i) = c_i^\times\right) \ ,$$

where for every $i \in [D]$, recall that $\mathbf{w}_i = \left(w_{i,1}, \ldots, w_{i,m'-1}\right)$, and where the arithmetic circuits are evaluated over the field $\mathbb{Z}_p$ underlying pp.

Claim 5.8 (Adaptive Soundness)**.** *Assuming an asymmetric graded encoding scheme satisfying Assumption 3.10, the bare-bones delegation protocol described in Section 5.3.3 satisfies the following property. for every pair of poly-size circuits $\mathsf{P}_1^*, \mathsf{P}_2^*$ there exists a negligible function μ such that for every $n \in \mathbb{N}$:*

$$\Pr\left[\begin{array}{l}\mathsf{pp} \leftarrow \mathsf{InstGen}(1^n, \kappa, \delta);\\ \mathsf{CRS} \leftarrow \mathsf{Gen}_{BB}(1^n, \mathsf{pp}, (\delta', m, D, \ell));\\ f, x \leftarrow \mathsf{P}_1^*(\mathsf{pp}, \mathsf{CRS});\\ y^*, \Pi^* \leftarrow \mathsf{P}_2^*(\mathsf{pp}, \mathsf{CRS}, f, x);\\ 1 \leftarrow \mathsf{V}_{BB}(\mathsf{pp}, \mathsf{CRS}, x, y^*, \Pi^*);\\ \mathsf{CHEAT}\end{array}\right] \le \mu(n) \ ,$$

where f is given by the arithmetic circuits:

$$\left\{\widetilde{\mathsf{add}}_i, \widetilde{\mathsf{sub}}_i, \widetilde{\mathsf{mult}}_i\right\}_{i\in[D]} \ ,$$

and f satisfies the conditions stated in the interface (Section 5.3.1), and

$$y^* = \left(\left\{[w_{i,j}]_{\mathbf{e}_{\ell+(i-1)\cdot m'+j}}\right\}_{i\in[D], j\in[m'-1]}, \left\{[c_i^+]_{\delta'\cdot\mathbf{u}_i}, [c_i^-]_{\delta'\cdot\mathbf{u}_i}, [c_i^\times]_{\delta'\cdot\mathbf{u}_i}\right\}_{i\in[D]}\right) \ ,$$

and CHEAT *is the event that:*

$$f(\mathbf{x}) \neq c \quad \wedge \quad \left(\forall i \in [D] : \left(\widetilde{\mathsf{add}}_i(\mathbf{w}_i) = c_i^+\right) \wedge \left(\widetilde{\mathsf{sub}}_i(\mathbf{w}_i) = c_i^-\right) \wedge \left(\widetilde{\mathsf{mult}}_i(\mathbf{w}_i) = c_i^\times\right)\right) \ ,$$

where recall that $\mathbf{x} = (x_1, \ldots, x_k)$ and for every $i \in [D]$, $\mathbf{w}_i = \left(w_{i,1}, \ldots, w_{i,m'-1}\right)$, and where the arithmetic above is over the underlying field $\mathbb{Z}_p$ of the public parameters pp.

Remark 5.9. Similarly to Remark 3.11, observe that the events OUT and CHEAT can be tested in polynomial time using the public parameters pp, using the arithmetic circuits:

$$\left\{\widetilde{\mathsf{add}}_i, \widetilde{\mathsf{sub}}_i, \widetilde{\mathsf{mult}}_i\right\}_{i\in[D]} \ ,$$

and using the operations Add, Sub, Mult, isZero on the output encodings y. For simplicity, we define these events as conditions on the values $\left\{\mathbf{w}_i, c_i^+, c_i^-, c_i^\times\right\}_{i\in[D]}$ (which cannot be efficiently computed from the output encoding), rather than explicitly describing the efficient testing procedure.

5.3.3 Construction

In this section we specify the strategies $\mathsf{Gen}_{BB}, \mathsf{P}_{BB}$ and V_{BB}.

The challenge generator Gen_{BB}. Gen_{BB} is given the security parameter 1^n, public parameters pp and the parameters δ, m, D, ℓ. For every $i \in [D]$, Gen_{BB} samples challenges $\mathsf{CRS}^i_{SC}, \mathsf{CRS}^i_{2\to1}$ for the sum-check and 2-to-1 sub-protocols (Sections 5.1,5.2). The D executions of the sub-protocols use encodings in different levels. Specifically, the protocol uses the last $D \cdot m'$ coordinates of the level vector, which we think of as divided into D groups of m' coordinates each.

1. The i-th sum-check sub-protocol uses the first $m' - 1 = 3m$ coordinates of the i-th group (the input is encoded in the previous coordinates).

$$\mathsf{CRS}^i_{SC} \leftarrow \mathsf{Gen}_{SC}(1^n, \mathsf{pp}, (\ell + (i-1)\cdot m', m'-1)) \ .$$

2. The i-th 2-to-1 sub-protocol uses the last coordinate of the i-th group (the input is encoded in the previous coordinates).

$$\mathsf{CRS}^i_{2\to1} \leftarrow \mathsf{Gen}_{2\to1}(1^n, \mathsf{pp}, \ell + (i \cdot m') - 1) \ .$$

Gen_{BB} outputs the challenge CRS:

$$\mathsf{CRS} = \Big(\mathsf{pp}, \big\{\big(\mathsf{CRS}^i_{SC}, \mathsf{CRS}^i_{2\to1}\big)\big\}_{i\in[D]}\Big) \ .$$

The prover P_{BB}. Recall that the Prover's input includes the public parameters pp, the challenge CRS, the function f described by the ring-independent arithmetic circuits: $\{\widetilde{\mathsf{add}}_i, \widetilde{\mathsf{sub}}_i, \widetilde{\mathsf{mult}}_i\}$ for each $i \in [D]$, and the encoded input:

$$x = \Big(\big\{[x_i]_{\mathbf{v}_i}\big\}_{i\in[k]}, [c]_{\mathbf{v}}\Big) \ .$$

Since the encoded input is valid for f, the prover P_{BB} can use the operations $\mathsf{Add}, \mathsf{Sub}, \mathsf{Mult}$ to evaluate f on the encodings of the input $\mathbf{x} = (x_1, \ldots, x_k)$. For every layer $i \in [0, D]$ and for every wire $\mathbf{w} \in \{0,1\}^m$, if $\mathbf{w}$ is a wire of level i, let $[x_{i,\mathbf{w}}]_{\mathbf{v}_{i,\mathbf{w}}}$ be the encoding that P_{BB} obtains for the wire $\mathbf{z}$ in the evaluation of $f(\mathbf{x})$. If $\mathbf{w}$ is not a wire of level i, let $x_{i,\mathbf{w}} = 0$ and let let $\mathbf{v}_{i,\mathbf{w}}$ be the all-zero level. For every $i \in [0, D], \mathbf{w} \in \{0,1\}^m$, we have that $\mathbf{v}_{i,\mathbf{w}} \leq \mathbf{v} \leq \frac{\delta}{2} \cdot \mathbf{s}_\ell$. For every $i \in [0, D]$, let X_i be a vector of size 2^m containing the values $x_{i,\mathbf{w}}$ for every $\mathbf{w} \in \{0,1\}^m$ in some canonical order.

Let $\widetilde{\mathsf{V}}_i$ be a multi-linear, ring-independent arithmetic circuit taking two inputs: X_i of length 2^m and $\mathbf{z}$ of length m (respectively). $\widetilde{\mathsf{V}}_i$ computes the multilinear extension of X_i at coordinate $\mathbf{z}$ as follows:

$$\widetilde{\mathsf{V}}_i(X_i, \mathbf{z}) = \sum_{\mathbf{w}\in\{0,1\}^m} \beta(\mathbf{z}, \mathbf{w}) \cdot x_{i,\mathbf{w}} \ . \tag{50}$$

(See Section 3.5 for the definition of β and for the proof that $\widetilde{\mathsf{V}}_i$ is ring-independent.) Note that for every $\mathbf{z} \in \{0,1\}^m$, $\widetilde{\mathsf{V}}_i(X_i, \mathbf{z}) = x_{i,\mathbf{z}}$.

Let $\mathbf{z}_0 \in \{0,1\}^m$ be the output wire of f, where we assume without loss of generality that $\mathbf{z}_0 = 0^m$. Let $c_0 = \widetilde{\mathsf{V}}_0(X_0, \mathbf{z}_0)$. To prove the statement $f(x) = c$, it is sufficient to prove the equivalent statement $c = c_0$. To this end, for every $i \in [D]$, P_{BB} obtains encoded values $\mathbf{z}_i, c_i$ and provides a proof Π_i for the fact that:

$$c_{i-1} \neq \widetilde{\mathsf{V}}_{i-1}(X_{i-1}, \mathbf{z}_{i-1}) \quad \Rightarrow \quad c_i \neq \widetilde{\mathsf{V}}_i(X_i, \mathbf{z}_i) \ .$$

These proofs let us proceed from each layer $i-1$ to the next layer i. The last statement $c_D = \widetilde{\mathsf{V}}_D(X_D, \mathbf{z}_D)$ has only to do with the circuit's input, and can be verified directly by V_{BB} in time $\tilde{O}(k) \cdot \mathrm{poly}(n)$.

The prover proceeds from the top (output) layer towards the bottom (input). For each layer $i \in [0, D]$ in order, P_{BB} obtains encodings of:

$$\mathbf{z}_i = (z_{i,1}, \ldots, z_{i,m}) \in \{0,1\}^m \ .$$

For every $j \in [m]$, we use $\mathbf{t}_{i,j}$ to denote the level under which the value $z_{i,j}$ is encoded and $\mathbf{t}_i$ be to denote under which the value c_i is encoded. P_{BB} maintains the invariant that the non-zero coordinates of $\mathbf{t}_{i,j}$ are in the range $[\ell+1, \ell+(i-1)\cdot m']$ that is:

$$\forall k \notin [\ell+1, \ell+(i-1)\cdot m'] : \mathbf{t}_{i,j}[k] = 0 \ . \tag{51}$$

P_{BB} also maintains that:

$$\mathbf{t}_i \geq \sum_{j \in [m]} \mathbf{t}_{i,j} \ . \tag{52}$$

P_{BB} sets $\mathbf{t}_0 = \mathbf{v}$ and for every $j \in [m]$, P_{BB} sets $\mathbf{t}_{0,j}$ to the all-zero level. (Note that Equations (51) and (52) are satisfied.) P_{BB} obtains the encodings:

$$\left\{ [z_{0,j}]_{\mathbf{t}_{0,j}} \right\}_{j \in [m]} \quad , \quad [c_0]_{\mathbf{t}_0} \ .$$

(For every $j \in [m]$, the encoding $[z_{0,j}]_{\mathbf{t}_{0,j}}$ is obtained using the fact that $z_{0,j} = 0$ and $[c_0]_{\mathbf{t}_0}$ is just the input encoding $[c]_{\mathbf{v}}$.)

Proceeding from layer $i-1$ to layer i. For every $i \in [D]$ starting from $i = 1$, P_{BB} obtains the encoded values $\mathbf{z}_i, c_i$ and the proof Π_i as follows:

1. **Sum-Check from $i-1$ to i:** Let g_i be an arithmetic circuit computing the following expression:

$$g_i(X_i, \mathbf{w}_1) = \sum_{\mathbf{w}_2, \mathbf{w}_3 \in \{0,1\}^m} \begin{pmatrix} & \widetilde{\mathsf{add}}_i(\mathbf{w}_1, \mathbf{w}_2, \mathbf{w}_3) \cdot \left(\widetilde{\mathsf{V}}_i(X_i, \mathbf{w}_2) + \widetilde{\mathsf{V}}_i(X_i, \mathbf{w}_3) \right) \\ + & \widetilde{\mathsf{sub}}_i(\mathbf{w}_1, \mathbf{w}_2, \mathbf{w}_3) \cdot \left(\widetilde{\mathsf{V}}_i(X_i, \mathbf{w}_2) - \widetilde{\mathsf{V}}_i(X_i, \mathbf{w}_3) \right) \\ + & \widetilde{\mathsf{mult}}_i(\mathbf{w}_1, \mathbf{w}_2, \mathbf{w}_3) \cdot \left(\widetilde{\mathsf{V}}_i(X_i, \mathbf{w}_2) \cdot \widetilde{\mathsf{V}}_i(X_i, \mathbf{w}_3) \right) \end{pmatrix} \ .$$

   If follows from the definition of the functions $\widetilde{\mathsf{V}}_i, \widetilde{\mathsf{V}}_{i-1}, \mathsf{add}_i, \mathsf{sub}_i, \mathsf{mult}_i$ and from the fact that the circuits $\widetilde{\mathsf{add}}_i, \widetilde{\mathsf{sub}}_i, \widetilde{\mathsf{mult}}_i$ are ring-independent that for for every $\mathbf{z} \in \{0,1\}^m$:

$$g_i(X_i, \mathbf{z}) = x_{i-1,\mathbf{z}} \ . \tag{53}$$

   when the circuit g_i is evaluated over the field $\mathbb{Z}_p$ underlying pp.

   Let:

$$f_i(X_i, \mathbf{z}, \mathbf{w}_1, \mathbf{w}_2, \mathbf{w}_3) = \\ = \beta(\mathbf{z}, \mathbf{w}_1) \cdot \begin{pmatrix} & \widetilde{\mathsf{add}}_i(\mathbf{w}_1, \mathbf{w}_2, \mathbf{w}_3) \cdot \left(\widetilde{\mathsf{V}}_i(X_i, \mathbf{w}_2) + \widetilde{\mathsf{V}}_i(X_i, \mathbf{w}_3) \right) \\ + & \widetilde{\mathsf{sub}}_i(\mathbf{w}_1, \mathbf{w}_2, \mathbf{w}_3) \cdot \left(\widetilde{\mathsf{V}}_i(X_i, \mathbf{w}_2) - \widetilde{\mathsf{V}}_i(X_i, \mathbf{w}_3) \right) \\ + & \widetilde{\mathsf{mult}}_i(\mathbf{w}_1, \mathbf{w}_2, \mathbf{w}_3) \cdot \left(\widetilde{\mathsf{V}}_i(X_i, \mathbf{w}_2) \cdot \widetilde{\mathsf{V}}_i(X_i, \mathbf{w}_3) \right) \end{pmatrix} \ . \tag{54}$$

   Combining Equation (53) and the definition of $\widetilde{\mathsf{V}}_{i-1}$ (see Equation (50)), we get that:

$$\widetilde{\mathsf{V}}_{i-1}(X_{i-1}, \mathbf{z}) \equiv \sum_{\mathbf{w}_1, \mathbf{w}_2, \mathbf{w}_3 \in \{0,1\}^n} f_i(X_i, \mathbf{z}, \mathbf{w}_1, \mathbf{w}_2, \mathbf{w}_3) \ . \tag{55}$$

P_{BB} executes the sum-check prover:

$$\mathsf{P}_{SC}\left(\mathsf{pp}, \mathsf{CRS}^i_{SC}, f_i, x^i_{SC}\right) \ ,$$

where:

$$x^i_{SC} = \left(\left\{[x_{i,\mathbf{w}}]_{\mathbf{v}_{i,\mathbf{w}}}\right\}_{\mathbf{w}\in\{0,1\}^m} \cup \left\{[z_{i-1,j}]_{\mathbf{t}_{i-1,j}}\right\}_{j\in[m]}, [c_{i-1}]_{\mathbf{t}_{i-1}}\right) \ .$$

Note that for every $\mathbf{w} \in \{0,1\}^m$, we have that $\mathbf{v}_{i,\mathbf{w}} \leq \frac{\delta}{2} \cdot \mathbf{s}_\ell$. Since f_i is of individual degree at most 2 in every variable $x_{i,\mathbf{w}}$ and since Equation (51) and Equation (52) hold, the partial encoded input:

$$\left\{[x_{i,\mathbf{w}}]_{\mathbf{v}_{i,\mathbf{w}}}\right\}_{\mathbf{w}\in\{0,1\}^m} \cup \left\{[z_{i-1,j}]_{\mathbf{t}_{i-1,j}}\right\}_{j\in[m]} \ ,$$

is valid for f_i. Since f_i is of individual degree at most $\delta' + 1$ in its last $3m$ variables, it follows that the input to P_{SC} satisfies the requirements described in Section 5.1.

P_{SC} returns an encoded output:

$$y^i_{SC} = \left(\left\{[w_{i,j}]_{\mathbf{e}_{\ell+(i-1)\cdot m'+j}}\right\}_{j\in[3m]}, \left[c^{\mathsf{out}}_i\right]_{\mathbf{t}_{i-1}+\delta\cdot\mathbf{u}_i}\right) \ ,$$

where:

$$\mathbf{u}_i = \sum_{j\in[3m]} \mathbf{e}_{\ell+(i-1)\cdot m'+j} \ .$$

P_{SC} also returns the sum-check proof Π^i_{SC} for the fact that:

$$c_{i-1} \neq \sum_{\mathbf{w}_1,\mathbf{w}_2,\mathbf{w}_3\in\{0,1\}^m} f_i(X_i, \mathbf{z}_{i-1}, \mathbf{w}_1, \mathbf{w}_2, \mathbf{w}_3) \quad \Rightarrow \quad c^{\mathsf{out}}_i \neq f_i(X_i, \mathbf{z}_{i-1}, \mathbf{w}^0_i, \mathbf{w}^1_i, \mathbf{w}^2_i)$$

where:

$$\begin{aligned} \mathbf{w}^0_i &= (w_{i,1}, \ldots, w_{i,m}) \ , \\ \mathbf{w}^1_i &= (w_{i,m+1}, \ldots, w_{i,2m}) \ , \\ \mathbf{w}^2_i &= (w_{i,2m+1}, \ldots, w_{i,3m}) \ . \end{aligned}$$

2. **Consistency of c^{out}_i:** To prove that:

$$c^{\mathsf{out}}_i = f_i(X_i, \mathbf{z}_{i-1}, \mathbf{w}^0_i, \mathbf{w}^1_i, \mathbf{w}^2_i) \ ,$$

P_{BB} uses the arithmetic circuits $\widetilde{\mathsf{add}}_i, \widetilde{\mathsf{sub}}_i, \widetilde{\mathsf{mult}}_i$ and the operations $\mathsf{Add}, \mathsf{Sub}, \mathsf{Mult}$ to obtain the encodings:

$$\left[c^+_i\right]_{\delta'\cdot\mathbf{u}_i}, \left[c^-_i\right]_{\delta'\cdot\mathbf{u}_i}, \left[c^\times_i\right]_{\delta'\cdot\mathbf{u}_i}, \left[c^1_i\right]_{\mathbf{t}_{i-1}+\mathbf{u}_i}, \left[c^2_i\right]_{\mathbf{t}_{i-1}+\mathbf{u}_i}$$

where:

$$\begin{aligned} c^+_i &= \widetilde{\mathsf{add}}_i(\mathbf{w}^0_i, \mathbf{w}^1_i, \mathbf{w}^2_i) \ , \\ c^-_i &= \widetilde{\mathsf{sub}}_i(\mathbf{w}^0_i, \mathbf{w}^1_i, \mathbf{w}^2_i) \ , \\ c^\times_i &= \widetilde{\mathsf{mult}}_i(\mathbf{w}^0_i, \mathbf{w}^1_i, \mathbf{w}^2_i) \ . \end{aligned}$$

and:

$$\begin{aligned} c^1_i &= \widetilde{\mathsf{V}}_i(X_i, \mathbf{w}^1_i) \ , \\ c^2_i &= \widetilde{\mathsf{V}}_i(X_i, \mathbf{w}^2_i) \ . \end{aligned} \tag{56}$$

3. **2-to-1:** P_{BB} obtains the encodings:

$$\left\{[w_{i,m+j}]_{\mathbf{u}_{i,j}}, [w_{i,2m+j}]_{\mathbf{u}_{i,j}}\right\}_{j\in[m]} \ ,$$

where:

$$\mathbf{u}_{i,j} = \mathbf{e}_{\ell+(i-1)\cdot m'+j} + \mathbf{e}_{\ell+(i-1)\cdot m'+m+j} + \mathbf{e}_{\ell+(i-1)\cdot m'+2m+j} \ .$$

Note that:

$$\sum_{j\in[m]} \mathbf{u}_{i,j} = \mathbf{u}_i \ . \tag{57}$$

(Note that we have an encoding of $w_{i,j}$ under the level $\mathbf{e}_{\ell+(i-1)\cdot m'+j}$. We use encodings in the higher level $\mathbf{u}_{i,j}$ so that Equation (57) is satisfied.)

To prove that Equation (56) holds, P_{BB} executes the 2-to-1 prover:

$$\mathsf{P}_{2\to1}\left(\mathsf{pp}, \mathsf{CRS}^i_{2\to1}, \widetilde{\mathsf{V}}_i, x^i_{2\to1}\right) \ ,$$

where:

$$x^i_{2\to1} = \left(\left\{[x_{i,\mathbf{w}}]_{\mathbf{v}_{i,\mathbf{w}}}\right\}_{\mathbf{w}\in\{0,1\}^m}, \left\{[w_{i,m+j}]_{\mathbf{u}_{i,j}}, [w_{i,2m+j}]_{\mathbf{u}_{i,j}}\right\}_{j\in[m]}, \left[c^1_i\right]_{\mathbf{t}_{i-1}+\mathbf{u}_i}, \left[c^2_i\right]_{\mathbf{t}_{i-1}+\mathbf{u}_i}\right) \ .$$

Note that since for every $\mathbf{z} \in \{0,1\}^m$, we have that $\mathbf{v}_{i,\mathbf{z}}$ is zero in all coordinates higher than ℓ and since Equation (57) hold, the partial encoded inputs:

$$\left\{[x_{i,\mathbf{w}}]_{\mathbf{v}_{i,\mathbf{w}}}\right\}_{\mathbf{w}\in\{0,1\}^m} \cup \left\{[w_{i,m+j}]_{\mathbf{u}_{i,j}}\right\}_{j\in[m]} \quad , \quad \left\{[x_{i,\mathbf{w}}]_{\mathbf{v}_{i,\mathbf{w}}}\right\}_{\mathbf{w}\in\{0,1\}^m} \cup \left\{[w_{i,2m+j}]_{\mathbf{u}_{i,j}}\right\}_{j\in[m]} \ ,$$

are valid for f_i and the output level of f_i for these inputs is indeed $\mathbf{t}_{i-1} + \mathbf{u}_i$.

$\mathsf{P}_{2\to1}$ returns an encoded output:

$$y^i_{2\to1} = \left(\left\{[z_{i,j}]_{\mathbf{t}_{i,j}}\right\}_{j\in[m]}, [c_i]_{\mathbf{t}_i}\right) \ ,$$

where:

$$\mathbf{t}_{i,j} = \mathbf{u}_{i,j} + \mathbf{e}_{\ell+i\cdot m'} \quad , \quad \mathbf{t}_i = \mathbf{t}_{i-1} + \mathbf{u}_i + m\cdot\mathbf{e}_{\ell+i\cdot m'} \ .$$

These encodings are used in the next iteration. (Note that Equations (51) and (52) are satisfied.)

$\mathsf{P}_{2\to1}$ also returns a 2-to-1 proof $\Pi^i_{2\to1}$ for the fact that:

$$\left(c^1_i \neq \widetilde{\mathsf{V}}_i(\mathbf{w}^1_i) \quad \vee \quad c^2_i \neq \widetilde{\mathsf{V}}_i(\mathbf{w}^2_i)\right) \quad \Rightarrow \quad c_i \neq \widetilde{\mathsf{V}}_i(\mathbf{z}_i) \ ,$$

P_{BB} outputs the encoded output:

$$y = \left(\left\{[w_{i,j}]_{\mathbf{e}_{\ell+(i-1)\cdot m'+j}}\right\}_{i\in[D],j\in[m'-1]}, \left\{\left[c^+_i\right]_{\delta'\cdot\mathbf{u}_i}, \left[c^-_i\right]_{\delta'\cdot\mathbf{u}_i}, \left[c^\times_i\right]_{\delta'\cdot\mathbf{u}_i}\right\}_{i\in[D]}\right) \ ,$$

and the proof:

$$\Pi = \{\Pi_i\}_{i\in[D]} = \left\{\left(x^i_{\mathsf{V}_{SC}}, y^i_{SC}, \Pi^i_{SC}, x^i_{\mathsf{V}_{2\to1}}, y^i_{2\to1}, \Pi^i_{2\to1}\right)\right\}_{i\in[D]} \ ,$$

where:

$$x^i_{\mathsf{V}_{SC}} = \left(\left\{[z_{i-1,j}]_{\mathbf{t}_{i-1,j}}\right\}_{j\in[m]}, [c_{i-1}]_{\mathbf{t}_{i-1}}\right) \ ,$$

$$x^i_{\mathsf{V}_{2\to1}} = \left(\left\{[w_{i,m+j}]_{\mathbf{u}_{i,j}}, [w_{i,2m+j}]_{\mathbf{u}_{i,j}}\right\}_{j\in[m]}, \left[c^1_i\right]_{\mathbf{t}_{i-1}+\mathbf{u}_i}, \left[c^2_i\right]_{\mathbf{t}_{i-1}+\mathbf{u}_i}\right) \ .$$

The verifier V_{BB}**.** Recall that the verifier's input includes:

1. The public parameters pp.

2. The challenge:
$$\mathsf{CRS} = \left(\mathsf{pp}, \left\{\left(\mathsf{CRS}^i_{SC}, \mathsf{CRS}^i_{2\to1}\right)\right\}_{i\in[D]}\right) \quad .$$

3. The encoded input:
$$x = \left(\left\{[x_i]_{\mathbf{v}_i}\right\}_{i\in[k]}, [c]_{\mathbf{v}}\right) \quad .$$

4. The encoded output:
$$y = \left(\left\{[w_{i,j}]_{\mathbf{e}_{\ell+(i-1)\cdot m'+j}}\right\}_{i\in[D],j\in[m'-1]}, \left\{\left[c_i^+\right]_{\delta'\cdot\mathbf{u}_i}, \left[c_i^-\right]_{\delta'\cdot\mathbf{u}_i}, \left[c_i^\times\right]_{\delta'\cdot\mathbf{u}_i}\right\}_{i\in[D]}\right) \quad ,$$

5. The proof:
$$\Pi = \left\{\left(x^i_{\mathsf{V}_{SC}}, y^i_{SC}, \Pi^i_{SC}, x^i_{\mathsf{V}_{2\to1}}, y^i_{2\to1}, \Pi^i_{2\to1}\right)\right\}_{i\in[D]} \quad .$$

Let $\mathbf{z}_0 = 0^m$ be the output gate of f and let $c_0 = c$. V_{BB} obtains the output layer encodings:
$$\left\{[z_{0,j}]_{\mathbf{t}_{0,j}}\right\}_{j\in[m]} \quad , \quad [c_0]_{\mathbf{t}_0} \quad .$$
(For every $j \in [m]$, the encoding $[z_{0,j}]_{\mathbf{t}_{0,j}}$ is obtained using the fact that $z_{0,j} = 0$ and $[c_0]_{\mathbf{t}_0}$ is just the input encoding $[c]_{\mathbf{v}}$.)

The verifier proceeds to check the proofs for consistency. We note that the proofs supplied to the verifier include encoded inputs and outputs, together with proofs that if the inputs specified false statements, then (w.h.p.) the outputs also specify false statements. Recall, however, that the verifier does not trust the prover who generated these proofs, and in particular the inputs and outputs are untrusted: the input for each sub-protocol must be compared with the output of the previous sub-protocol to guarantee consistency. With this in mind, the verifier operates as follows. For every $i \in [D]$, let:
$$\begin{aligned}
x^i_{\mathsf{V}_{SC}} &= \left(\left\{[\bar{z}_{i-1,j}]_{\mathbf{t}_{i-1,j}}\right\}_{j\in[m]}, [\bar{c}_{i-1}]_{\mathbf{t}_{i-1}}\right) \quad ,\\
y^i_{SC} &= \left(\left\{[\bar{w}_{i,j}]_{\mathbf{e}_{ell+(i-1)\cdot m'+j}}\right\}_{j\in[3m]}, \left[c_i^{\mathsf{out}}\right]_{\mathbf{t}_{i-1}+\delta\cdot\mathbf{u}_i}\right) \quad ,\\
x^i_{\mathsf{V}_{2\to1}} &= \left(\left\{[\bar{\bar{w}}_{i,m+j}]_{\mathbf{u}_{i,j}}, [\bar{\bar{w}}_{i,2m+j}]_{\mathbf{u}_{i,j}}\right\}_{j\in[m]}, \left[c_i^1\right]_{\mathbf{t}_{i-1}+\mathbf{u}_i}, \left[c_i^2\right]_{\mathbf{t}_{i-1}+\mathbf{u}_i}\right) \quad ,\\
y^i_{2\to1} &= \left(\left\{[z_{i,j}]_{\mathbf{t}_{i,j}}\right\}_{j\in[m]}, [c_i]_{\mathbf{t}_i}\right) \quad .
\end{aligned}$$

For every $i \in [D]$, V_{BB} performs the following tests:

1. V_{BB} uses the operations $\mathsf{Sub}, \mathsf{isZero}$ to verify that:
$$\forall j \in [m] : \bar{z}_{i-1,j} = z_{i-1,j} \quad , \quad \bar{c}_{i-1} = c_{i-1} \quad .$$

2. V_{BB} verifies that the i-th sum-check proof is accepting:
$$1 \leftarrow \mathsf{V}_{SC}(\mathsf{pp}, \mathsf{CRS}^i_{SC}, [\bar{c}_{i-1}]_{\mathbf{t}_{i-1}}, y^i_{SC}, \Pi^i_{SC}) \quad .$$

3. V_{BB} uses the operations $\mathsf{Sub}, \mathsf{isZero}$ to verify that:
$$\forall j \in [3m] : \bar{w}_{i,j} = w_{i,j} \quad .$$

4. V_{BB} uses the operations $\mathsf{Add}, \mathsf{Sub}, \mathsf{Mult}, \mathsf{isZero}$ to verify that:

$$c_i^{\mathsf{out}} = \beta(\bar{z}_{i-1,1}, \ldots, \bar{z}_{i-1,m}, w_{i,1}, \ldots, w_{i,m}) \cdot \left(c_i^+ \cdot \left(c_i^1 + c_i^2\right) + c_i^- \cdot \left(c_i^1 - c_i^2\right) + c_i^\times \cdot \left(c_i^1 \cdot c_i^2\right)\right) \ .$$

5. V_{BB} uses the operations $\mathsf{Sub}, \mathsf{isZero}$ to verify that:

$$\forall j \in [m] : \bar{\bar{w}}_{i,m+j} = \bar{w}_{i,m+j} \quad \wedge \quad \bar{\bar{w}}_{i,2m+j} = \bar{w}_{i,2m+j} \ .$$

6. V_{BB} verifies that the i-th 2-to-1 proof is accepting:

$$1 \leftarrow \mathsf{V}_{2\to1}(\mathsf{pp}, \mathsf{CRS}^i_{2\to1}, x^i_{\mathsf{V}_{2\to1}}, y^i_{2\to1}, \Pi^i_{2\to1}) \ .$$

Finally, following Equation (50), V_{BB} uses the input encodings and the operations $\mathsf{Add}, \mathsf{Sub}, \mathsf{Mult}, \mathsf{isZero}$ to verify that:

$$c_D = \widetilde{\mathsf{V}}_D(X_D, z_{D,1}, \ldots, z_{D,m}) \ . \tag{58}$$

Recall that X_D is simply the circuit's input, which is available to the verifier (albeit in encoded form). Note that in Equation (50), only k of the summands are non-zero, and so this final verification step can be completed by the verifier in time that only depends linearly on k (and polynomially on n and D).

V_{BB} accepts iff all tests pass.

5.3.4 Proof of Claim 5.8

Proof. Let $(\mathsf{P}_1^*, \mathsf{P}_2^*)$ be a pair of poly-size circuits. To prove Claim 5.8, recall that we need to show that there exists a negligible function μ such that for every security parameter $n \in \mathbb{N}$:

$$\Pr\left[\begin{array}{l} \mathsf{pp} \leftarrow \mathsf{InstGen}(1^n, \kappa, \delta); \\ \mathsf{CRS} \leftarrow \mathsf{Gen}_{BB}(1^n, \mathsf{pp}, (\delta', m, D, \ell)); \\ f, x \leftarrow \mathsf{P}_1^*(\mathsf{pp}, \mathsf{CRS}); \\ y^*, \Pi^* \leftarrow \mathsf{P}_2^*(\mathsf{pp}, \mathsf{CRS}, f, x); \\ 1 \leftarrow \mathsf{V}_{BB}(\mathsf{pp}, \mathsf{CRS}, x, y^*, \Pi^*); \\ \mathsf{CHEAT} \end{array}\right] \leq \mu(n) \ ,$$

where:

$$y^* = \left(\left\{[w_{i,j}]_{\mathbf{e}_{\ell+(i-1)\cdot m'+j}}\right\}_{i\in[D], j\in[m'-1]}, \left\{\left[c_i^+\right]_{\delta'\cdot\mathbf{u}_i}, \left[c_i^-\right]_{\delta'\cdot\mathbf{u}_i}, \left[c_i^\times\right]_{\delta'\cdot\mathbf{u}_i}\right\}_{i\in[D]}\right) \ ,$$

and CHEAT is the event that:

$$f(\mathbf{x}) \neq c \quad \wedge \quad \left(\forall i \in [D] : \left(\widetilde{\mathsf{add}}_i(\mathbf{w}_i) = c_i^+\right) \wedge \left(\widetilde{\mathsf{sub}}_i(\mathbf{w}_i) = c_i^-\right) \wedge \left(\widetilde{\mathsf{mult}}_i(\mathbf{w}_i) = c_i^\times\right)\right) \ ,$$

where recall that $\mathbf{x} = (x_1, \ldots, x_k)$ and for every $i \in [D]$ $\mathbf{w}_i = \left(w_{i,1}, \ldots, w_{i,m'-1}\right)$, and where the arithmetic above and in what follows is over the underlying field $\mathbb{Z}_p$ of the public parameters pp.

Recall also the notation used in the construction. In the above experiment:

$$\Pi^* = \left\{\left(x^i_{\mathsf{V}_{SC}}, y^i_{SC}, \Pi^i_{SC}, x^i_{\mathsf{V}_{2\to1}}, y^i_{2\to1}, \Pi^i_{2\to1}\right)\right\}_{i\in[D]} \ ,$$

for every $i \in [D]$, let:

$$\begin{aligned}
x^i_{\mathsf{V}_{SC}} &= \left(\left\{[\bar{z}_{i-1,j}]_{\mathbf{t}_{i-1,j}}\right\}_{j\in[m]}, [\bar{c}_{i-1}]_{\mathbf{t}_{i-1}}\right) \ , \\
y^i_{SC} &= \left(\left\{[\bar{w}_{i,j}]_{\mathbf{e}_{ell+(i-1)\cdot m'+j}}\right\}_{j\in[3m]}, \left[c^{\mathsf{out}}_i\right]_{\mathbf{t}_{i-1}+\delta\cdot\mathbf{u}_i}\right) \ , \\
x^i_{\mathsf{V}_{2\to1}} &= \left(\left\{[\bar{\bar{w}}_{i,m+j}]_{\mathbf{u}_{i,j}}, [\bar{\bar{w}}_{i,2m+j}]_{\mathbf{u}_{i,j}}\right\}_{j\in[m]}, \left[c^1_i\right]_{\mathbf{t}_{i-1}+\mathbf{u}_i}, \left[c^2_i\right]_{\mathbf{t}_{i-1}+\mathbf{u}_i}\right) \ , \\
y^i_{2\to1} &= \left(\left\{[z_{i,j}]_{\mathbf{t}_{i,j}}\right\}_{j\in[m]}, [c_i]_{\mathbf{t}_i}\right) \ .
\end{aligned}$$

and let:

$$\begin{aligned}
\mathbf{z}_i &= (z_{i,1}, \dots, z_{i,m}) \ , \\
\bar{\mathbf{z}}_i &= (\bar{z}_{i,1}, \dots, \bar{z}_{i,m}) \ , \\
\bar{\mathbf{w}}_i &= (\bar{w}_{i,1}, \dots, \bar{w}_{i,3m}) \ , \\
\bar{\bar{\mathbf{w}}}^1_i &= (\bar{\bar{w}}_{i,m+1}, \dots, \bar{\bar{w}}_{i,2m}) \ , \\
\bar{\bar{\mathbf{w}}}^2_i &= (\bar{\bar{w}}_{i,2m1}, \dots, \bar{\bar{w}}_{i,3m}) \ .
\end{aligned}$$

For every $i \in [0, D]$ let X_i and $\widetilde{\mathsf{V}}_i$ be defined as in the honest prover strategy P_{BB}. For every $i \in [D]$ let f_i be defined as in Equation (54).

Define the following events:

$$\begin{aligned}
\mathsf{CHEAT}^i_{SC}: &\quad \left(\bar{c}_{i-1} \neq \sum_{\mathbf{w}\in\{0,1\}^{3m}} f_i(\bar{\mathbf{z}}_{i-1}, \mathbf{w})\right) \wedge \left(c^{\mathsf{out}}_i = f_i(\bar{\mathbf{z}}_{i-1}, \bar{\mathbf{w}}_i)\right) \\
\mathsf{CHEAT}^i_{2\to1}: &\quad \left(\left(c^1_i \neq \widetilde{\mathsf{V}}_i(X_i, \bar{\bar{\mathbf{w}}}_{i,2})\right) \vee \left(c^2_i \neq \widetilde{\mathsf{V}}_i(X_i, \bar{\bar{\mathbf{w}}}_{i,3})\right)\right) \wedge \left(c_i = \widetilde{\mathsf{V}}_i(X_i, \mathbf{z}_i)\right) \\
\mathsf{CHEAT}^i: &\quad \left(c_{i-1} \neq \widetilde{\mathsf{V}}_{i-1}(X_{i-1}, \mathbf{z}_{i-1})\right) \wedge \left(c_i = \widetilde{\mathsf{V}}_i(X_i, \mathbf{z}_i)\right) \\
\mathsf{OUT}^i: &\quad \left(\widetilde{\mathsf{add}}_i(\mathbf{w}_i) = c^+_i\right) \wedge \left(\widetilde{\mathsf{sub}}_i(\mathbf{w}_i) = c^-_i\right) \wedge \left(\widetilde{\mathsf{mult}}_i(\mathbf{w}_i) = c^\times_i\right)
\end{aligned}$$

Let ACCEPT be the event that in the above experiment, the verifier V_{BB} accepts the proof Π^*. When ACCEPT occurs, Equation (58) holds:

$$c_D = \widetilde{\mathsf{V}}_D(X_D, \mathbf{z}_D) \ .$$

We will show that there exist negligible function μ_1 such that for every $i \in [D]$:

$$\Pr\left[\mathsf{ACCEPT} \wedge \mathsf{OUT}^i \wedge \mathsf{CHEAT}^i\right] \leq \mu_1(n) \ . \tag{59}$$

By a Union Bound, we then conclude that:

$$\begin{aligned}
\Pr\left[\mathsf{ACCEPT} \wedge \mathsf{CHEAT}\right] &= \Pr\left[\mathsf{ACCEPT} \wedge \mathsf{OUT}^1 \wedge \dots \wedge \mathsf{OUT}^D \quad \wedge \quad \left(c \neq \widetilde{\mathsf{V}}_0(X_0, \mathbf{z}_0)\right)\right] \\
&\leq D \cdot \mu_1(n) \ ,
\end{aligned}$$

as required.

In the rest of the proof we will prove Equation (59). It is sufficient to prove that:

$$\Pr\left[c_i = \widetilde{\mathsf{V}}_i(X_i, \mathbf{z}_i) \mid \mathsf{ACCEPT} \wedge \mathsf{OUT}^i \wedge \left(c_{i-1} \neq \widetilde{\mathsf{V}}_{i-1}(X_{i-1}, \mathbf{z}_{i-1})\right)\right] \leq \mu_1(n) \ .$$

To this end, we assume that the event:

$$\mathsf{ACCEPT} \wedge \mathsf{OUT}^i \wedge \left(c_{i-1} \neq \widetilde{\mathsf{V}}_{i-1}(X_{i-1}, \mathbf{z}_{i-1})\right) \quad ,$$

holds, and we prove that:

$$\Pr\left[c_i = \widetilde{\mathsf{V}}_i(X_i, \mathbf{z}_i)\right] \leq \mu_1(n) \ .$$

Since the event ACCEPT holds all of V_{BB}'s tests passes. Since Test 1 passes:

$$\bar{c}_{i-1} \neq \widetilde{\mathsf{V}}_{i-1}(X_{i-1}, \bar{\mathbf{z}}_{i-1}) \ .$$

By Equation (55):

$$\bar{c}_{i-1} \neq \sum_{\mathbf{w} \in \{0,1\}^{3m}} f_i(X_i, \bar{\mathbf{z}}_{i-1}, \mathbf{w}_i) \ .$$

Since V_{BB} accepts the sum-check proof Π^i_{SC} (Test 2), it follows from Claim 5.2 that for some negligible function μ_2:

$$\begin{aligned}\Pr\left[\mathsf{CHEAT}^i_{SC}\right] &= \Pr\left[\left(\bar{c}_{i-1} \neq \sum_{\mathbf{w} \in \{0,1\}^{3m}} f_i(X_i, \bar{\mathbf{z}}_{i-1}, \mathbf{w})\right) \wedge \left(c_i^{\mathsf{out}} = f_i(X_i, \bar{\mathbf{z}}_{i-1}, \bar{\mathbf{w}}_i)\right)\right] \\ &\leq \mu_2(n) \ .\end{aligned}$$

(Otherwise we can turn P_1^* and P_2^* into adversaries that break the adaptive soundness of the sum-check protocol.) Therefore:

$$\Pr\left[c_i^{\mathsf{out}} = f_i(X_i, \bar{\mathbf{z}}_{i-1}, \bar{\mathbf{w}}_i)\right] \leq \mu_2(n) \ ,$$

and since V_{BB}'s Test 3 passes:

$$\Pr\left[c_i^{\mathsf{out}} = f_i(X_i, \bar{\mathbf{z}}_{i-1}, \mathbf{w}_i)\right] \leq \mu_2(n) \ .$$

By the definition of f_i (Equation (54)) and since he event OUT^i holds and Test 4 passes:

$$\Pr\left[c_i^1 = \widetilde{\mathsf{V}}_i(X_i, \mathbf{w}_i^1) \quad \wedge \quad c_i^2 = \widetilde{\mathsf{V}}_i(X_i, \mathbf{w}_i^2)\right] \leq \mu_2(n) \ ,$$

where

$$\mathbf{w}_i^1 = (w_{i,m+1}, \ldots, w_{i,2m}) \quad , \quad \mathbf{w}_i^2 = (w_{i,2m+1}, \ldots, w_{i,3m}) \ .$$

Since V_{BB}'s Test 5 passes:

$$\Pr\left[\widetilde{\mathsf{V}}_i(X_i, \bar{\bar{\mathbf{w}}}_i^1) = c_i^1 \quad \wedge \quad \widetilde{\mathsf{V}}_i(X_i, \bar{\bar{\mathbf{w}}}_i^2) = c_i^2\right] \leq \mu_2(n) \ .$$

Since V_{BB} accepts the 2-to-1 proof $\Pi^i_{2\to1}$ (Test 6), it follows from Claim 5.5 that for some negligible function μ_3:

$$\begin{aligned}\Pr\left[\mathsf{CHEAT}^i_{2\to1}\right] &= \Pr\left[\left(\left(c_i^1 \neq \widetilde{\mathsf{V}}_i(X_i, \bar{\bar{\mathbf{w}}}_{i,2})\right) \vee \left(c_i^2 \neq \widetilde{\mathsf{V}}_i(X_i, \bar{\bar{\mathbf{w}}}_{i,3})\right)\right) \wedge \left(c_i = \widetilde{\mathsf{V}}_i(X_i, \mathbf{z}_i)\right)\right] \\ &\leq \mu_3(n) \ .\end{aligned}$$

(Otherwise we can turn P_1^* and P_2^* into adversaries that break the adaptive soundness of the 2-to-1 protocol.) By union bound, we have that for $\mu_1 = \mu_2 + \mu_3$:

$$\Pr\left[c_i = \widetilde{\mathsf{V}}_i(X_i, \mathbf{z}_i)\right] \leq \mu_1(n) \ .$$

□

5.4 Non-Interactive Arguments for Bounded-Depth Computations

In this section we instantiate the bare-bones delegation protocol $(\mathsf{Gen}_{BB}, \mathsf{P}_{BB}, \mathsf{V}_{BB})$ from Section 5.3. We obtain non-interactive publicly verifiable arguments for functions that are computable by log-space uniform bounded-depth circuits.

Recall that in the Bare-Bones protocol, $\widetilde{\mathsf{add}}_i, \widetilde{\mathsf{sub}}_i, \widetilde{\mathsf{mult}}_i$ are bounded-degree and ring-independent extensions of the functions that define the circuit computing f. The output of the bare-bones protocol is a collection of encodings, and claims for the values of $\widetilde{\mathsf{add}}_i, \widetilde{\mathsf{sub}}_i, \widetilde{\mathsf{mult}}_i$ on those encodings. Completeness and soundness of the bare-bones guarantee that if $f(x) = 1$ and the proof is generated properly, then the claims are correct. If $f(x) \neq 1$, then w.h.p at least one of these claims will be false.

Thus, to instantiate the base-bones protocol, all that remains is to show how the verifier can check the claimed values of $\widetilde{\mathsf{add}}_i, \widetilde{\mathsf{sub}}_i, \widetilde{\mathsf{mult}}_i$. We can do this in two ways:

1. In Section 5.4.1 we consider functions f that are computable in (nondeterministic) logarithmic space. These functions can be computed by circuits of polylogarithmic depth, where $\widetilde{\mathsf{add}}_i, \widetilde{\mathsf{sub}}_i, \widetilde{\mathsf{mult}}_i$ can be computed by (uniform) circuits of polylogarithmic size and polylogarithmic degree δ' (an arithmetic circuit variant of the notion of constructible boolean circuits, see Definition 3.20).

   For this class of functions, we can directly instantiate the non-interactive bare-bones protocol on the un-encoded input x (there is no need to consider an encoded input). The verifier can explicitly compute the values of $\widetilde{\mathsf{add}}_i, \widetilde{\mathsf{sub}}_i, \widetilde{\mathsf{mult}}_i$ on the encoded outputs of the bare-bones protocol. This gives a publicly verifiable non-interactive argument for any language in $\mathcal{NL}$.

   This result is stated in Theorem 5.11. It is included mainly to facilitate exposition of our full result for bounded-depth computations (see below).

2. In Section 5.4.2 we consider functions f that are computable by bounded-depth circuits that are log-space uniform, i.e. the circuits' description can be produced by a logspace Turing Machine (in polynomial time).

   Here we take the functions $\widetilde{\mathsf{add}}_i, \widetilde{\mathsf{sub}}_i, \widetilde{\mathsf{mult}}_i$ to be the unique multilinear extensions of the functions $\mathsf{add}_i, \mathsf{sub}_i, \mathsf{mult}_i$ that specify the circuit (see Section 3.5). For logspace-uniform circuits, these multilinear extensions can also be computed in logarithmic space, as was done in [GKR08]. We emphasize that the *time* to compute these functions may be as large as the circuit size (or larger). Thus, the verifier cannot compute these functions on her own. Instead, she runs the bare-bones protocol, and obtains claims about these functions' outputs (as given in the bare-bones protocol's output). These claims are verified using a separate tailored delegation protocol. The verifier receives a separate proof, and uses it to verify the claimed values of these functions. This (again) is as was done in [GKR08].

   In our setting of non-interactive delegation, however, there is an additional difficulty. The functions $\widetilde{\mathsf{add}}_i, \widetilde{\mathsf{sub}}_i, \widetilde{\mathsf{mult}}_i$ can be computed in logarithmic-space, and so we might hope to use the protocol of Theorem 5.11 to delegate their computation (as was done in [GKR08]). In our setting, however, the verifier only knows *encodings of* the inputs on which $\widetilde{\mathsf{add}}_i, \widetilde{\mathsf{sub}}_i, \widetilde{\mathsf{mult}}_i$ were evaluated. Thus, we need to use the bare-bones protocol on encoded inputs. Note that this requires the prover to be able to evaluate the delegated functions on the encoded inputs. This is a (highly) non-trivial condition, and indeed the bare-bones protocol cannot be used to delegate arbitrary log-space computations on encoded inputs, since those computations might be of very high degree.

   Fortunately, the computation of $\widetilde{\mathsf{add}}_i, \widetilde{\mathsf{sub}}_i, \widetilde{\mathsf{mult}}_i$ are not arbitrary log-space computations. In fact, they can be computed by low-depth circuits whose outputs (and internal gates) are *multilinear* in the inputs. Moreover, we show that these low-degree circuits that compute the $\widetilde{\mathsf{add}}_i, \widetilde{\mathsf{sub}}_i, \widetilde{\mathsf{mult}}_i$ are themselves *constructible* (see Definition 3.20). Their wiring functions can be computed by (uniform) circuits of polylogarithmic size and degree. Thus, we can use the Bare-Bones protocol for encoded inputs to verify the computation of these functions. This gives publicly verifiable non-interactive arguments for log-space uniform bounded-depth computations.

This result is stated in Theorem 5.13

Notation and setup. Before proceeding to show both of these instantiations, we briefly recall the notation from Section 5.3. Let f be a layered fan-in 2, strongly ring-independent arithmetic circuit with k inputs. (For simplicity and without loss of generality we assume that S is a power of 2.) Index every gate of f by a string in $\{0,1\}^m$. Number the layers of f such that layer 0 is the output layer and layer D is the input layer.

For every $i \in [D]$ let:

$$\mathsf{add}_i, \mathsf{sub}_i, \mathsf{mult}_i : \{0,1\}^{3m} \to \{0,1\} \ ,$$

be the functions that $\mathsf{add}_i(g_1, g_2, g_3) = 1$ if g_1 is a gate in layer $i-1$, g_2, g_3 are gates is in layer i and g_1 is adding the output of g_2 and g_3. Otherwise $\mathsf{add}_i(g_1, g_2, g_3) = 0$. sub_i and mult_i are defined similarly.

Let δ be a degree parameter and let $\widetilde{\mathsf{add}}_i, \widetilde{\mathsf{sub}}_i, \widetilde{\mathsf{mult}}_i$ be ring-independent arithmetic circuits of degree $\delta' < \delta$ computing the functions $\mathsf{add}_i, \mathsf{sub}_i, \mathsf{mult}_i$. We emphasize that these "algebraic extensions" of the (boolean) functions $\mathsf{add}_i, \mathsf{sub}_i, \mathsf{mult}_i$ need not compute the *unique multilinear extensions*. Rather, they compute some arbitrary functions that agree with $\mathsf{add}_i, \mathsf{sub}_i, \mathsf{mult}_i$ over boolean inputs. (In some cases, however, we *will* use $\mathsf{add}_i, \mathsf{sub}_i, \mathsf{mult}_i$ that compute the unique multilinear extensions of their respective functions).

The bare-bones delegation protocol of Section 5.3 takes an input x for f and outputs encodings:

$$\left(\left\{ [w_{i,j}]_{\mathbf{e}_{(i-1)\cdot m'+j}} \right\}_{i\in[D], j\in[m'-1]}, \left\{ [c_i^+]_{\delta'\cdot\mathbf{u}_i}, [c_i^-]_{\delta'\cdot\mathbf{u}_i}, [c_i^\times]_{\delta'\cdot\mathbf{u}_i} \right\}_{i\in[D]} \right) \ ,$$

The soundness guarantee is that if $f(x) \neq 1$, then the verifier is guaranteed that (w.h.p.) either she will reject, or one of these output encodings specifies a false claim (see Claim 5.8):

$$\left(\exists i \in [D] : \left(\widetilde{\mathsf{add}}_i(\mathbf{w}_i) \neq c_i^+ \right) \vee \left(\widetilde{\mathsf{sub}}_i(\mathbf{w}_i) \neq c_i^- \right) \vee \left(\widetilde{\mathsf{mult}}_i(\mathbf{w}_i) \neq c_i^\times \right) \right) \ .$$

where for every $i \in [D]$, recall that: $\mathbf{w}_i = \left(w_{i,1}, \ldots, w_{i,m'-1} \right)$, and where the arithmetic above and in what follows is over the underlying field $\mathbb{Z}_p$ of the public parameters pp.

Observe that each encoding of an input $w_{i,j}$ to the functions $\widetilde{\mathsf{add}}_i, \widetilde{\mathsf{sub}}_i, \widetilde{\mathsf{mult}}_i$ is encoded at level $\mathbf{e}_{(i-1)\cdot m'+j}$. Thus, the graded encoding supports computations of degree $\delta > \delta'$ in each of these inputs. The functions $\widetilde{\mathsf{add}}_i, \widetilde{\mathsf{sub}}_i, \widetilde{\mathsf{mult}}_i$ are of degree δ' in each variable, and thus the graded encoding scheme supports the computation of these functions on the encoded values $w_{i,j}$. If the verifier can verify the claims (on encoded inputs), either on her own, or using a non-interactive argument, then we obtain a complete and sound argument system for verifying that $f(x) = 1$.

5.4.1 Non-Interactive Arguments for $\mathcal{NL}$

We show that functions that can be computed in $\mathcal{NL}$ have low-depth circuits where there exist extensions $\widetilde{\mathsf{add}}_i, \widetilde{\mathsf{sub}}_i, \widetilde{\mathsf{mult}}_i$ that can be computed in polylogarithmic time and degree. This is stated in Lemma 5.10 below. Combining this with the completeness and soundness of the bare-bones argument system (Claims 5.7 and 5.8), we obtain a publicly-verifiable argument system for any language in $\mathcal{NL}$. This result is stated in Theorem 5.11 below.

Lemma 5.10. *Let $\mathcal{L}$ be a language computed by a non-deterministic Turing Machine T in time* $\mathrm{poly}(k)$ *and space* $O(\log k)$ *(where k is the input length). There exists an ensemble of ring-independent arithmetic circuits $\{f_k\}$ for computing $\mathcal{L}$ with* $O(\log^2 k)$*-depth and* $\mathrm{poly}(k)$*-size.*

Fix an input length k, a circuit $f = f_k$ of depth D and size S, and take $m = \log S$. For every $i \in [D]$, there exist ring-independent *circuits* $\widetilde{\mathsf{add}}_i, \widetilde{\mathsf{sub}}_i, \widetilde{\mathsf{mult}}_i$ *for computing* $\mathsf{add}_i, \mathsf{sub}_i, \mathsf{mult}_i$. *These circuits have degree $\delta' = \mathrm{polylog}(k)$, can be constructed in $\mathrm{polylog}(k)$-time, and can be evaluated (on an input in some ring) using $\mathrm{polylog}(k)$ ring operations (additions, subtractions and multiplications).*

Proof. The lemma follows from Lemma 4.1 in [GKR08]. They show the existence of an ensemble as above of circuits over $GF[2]$, where the functions $\mathsf{add}_i, \mathsf{sub}_i, \mathsf{mult}_i$ are computed by constant depth AC^0 circuits of $\mathrm{polylog}(k)$ size, which can be generated and evaluated in $\mathrm{polylog}(k)$ time. The only difference for us is that we want a *ring-independent* circuit ensemble computing the function, and we want *ring-independent* circuits for computing $\widetilde{\mathsf{add}}_i, \widetilde{\mathsf{sub}}_i, \widetilde{\mathsf{mult}}_i$:

Ring-independent ensemble. For an input length k, the circuit $f = f_k$ computes the adjacency matrix of the graph comprising all $\mathrm{poly}(n)$ Turing Machine states on the given input x, and then checks for a path from the starting state to an accepting state in this graph. This is done by a kind of repeated squaring of the adjacency matrix.

To compute the initial adjacency matrix B (for the fixed input x), observe that each edge is either always present (constant 1), never present (constant 0), equal to the i-th bit of x, or to the negation of the i-th bit (i.e. $1 - x_i$). All these assignments are ring-independent.

For “repeated squaring”, we use the matrix $B^{2^{c-1}}$, where the $[u, v]$-th entry is 1 if there is a path of length at most 2^{c-1} from u to v, and compute the matrix B^{2^c}, defined similarly but for paths of length at most 2^c. This computation is done using the ring-independent equation:

$$B^{2^c}[u, v] = 1 - \left(\prod_w \left[1 - (B^{2^{c-1}}[u, w] \cdot B^{2^{c-1}}[w, v]) \right] \right),$$

which guarantees that the $[u, v]$-th entry of B^{2^c} is 1 if there exists an intermediate node w such that there are paths of length at most 2^{c-1} from u to w and from w to v. Otherwise, the $[u, v]$-th of B^{2^c} is 0.

Ring-independent $\widetilde{\mathsf{add}}_i, \widetilde{\mathsf{sub}}_i, \widetilde{\mathsf{mult}}_i$. Lemma 4.1 of [GKR08] gives AC^0 boolean circuits computing $\mathsf{add}_i, \mathsf{sub}_i, \mathsf{mult}_i$. These can be converted to ring-independent circuits by replacing AND gates with multiplication, and replacing NOT gates with subtraction from 1. The resulting ring-independent circuit has degree $\mathrm{polylog}(k)$, can be constructed in $\mathrm{polylog}(k)$ time, and evaluated in $\mathrm{polylog}(k)$ ring operations (multiplication and subtraction). Note that converting the AC^0 circuit to operate over the ring maintains the time to construct (and evaluate) the circuit. $\square$

Theorem 5.11. *Let $\mathcal{L}$ be a language in $\mathcal{NL}$. The Bare-Bones protocol $(\mathsf{Gen}_{BB}, \mathsf{P}_{BB}, \mathsf{V}_{BB})$ from Section 5.3, instantiated with the circuits $\widetilde{\mathsf{add}}_i, \widetilde{\mathsf{sub}}_i, \widetilde{\mathsf{mult}}_i$ from Lemma 5.10, gives an adaptively sound publicly-verifiable non-interactive argument system for $\mathcal{L}$. For security parameter n and input length k:*

- *The challenge generator runs in time* $\mathrm{poly}(n)$.
- *The (honest) prover runs in time* $\mathrm{poly}(n, k)$.
- *The verifier runs in time* $k \cdot \mathrm{poly}(n)$.

5.4.2 Bounded Depth Computations

We show that for functions that can be computed by log-space uniform bounded-depth circuits, there exist circuits implementing the functions $\widetilde{\mathsf{add}}_i, \widetilde{\mathsf{sub}}_i, \widetilde{\mathsf{mult}}_i$ whose computations can themselves be delegated. In particular, $\widetilde{\mathsf{add}}_i, \widetilde{\mathsf{sub}}_i, \widetilde{\mathsf{mult}}_i$ are implemented by *constructible* circuits of polylog depth and polynomial size (see Definition 3.20). This is stated in Lemma 5.12. We then use these implementations of $\widetilde{\mathsf{add}}_i, \widetilde{\mathsf{sub}}_i, \widetilde{\mathsf{mult}}_i$ to obtain a non-interactive argument for such low depth computations, as stated in Theorem 5.13 below.

Lemma 5.12. *Let $\mathcal{L}$ be a language computed by logspace-uniform circuits of polynomial size. There exist* ring-independent *circuits* $\widetilde{\mathsf{add}}_i, \widetilde{\mathsf{sub}}_i, \widetilde{\mathsf{mult}}_i$ *for computing the multilinear extensions of* $\mathsf{add}_i, \mathsf{sub}_i, \mathsf{mult}_i$ *as follows. The circuits for computing* $\widetilde{\mathsf{add}}_i, \widetilde{\mathsf{sub}}_i, \widetilde{\mathsf{mult}}_i$ *are* constructible *(see Definition 3.20), of* polylog *depth, and their internal gates (and output gate) are all multilinear in the input.*

Proof Sketch. We sketch a proof for add_i. The constructions for $\mathsf{sub}_i, \mathsf{mult}_i$ are similar. Recall that:

$$\widetilde{\mathsf{add}}_i(\mathbf{z}) = \sum_{\mathbf{y} \in \{0,1\}^{3m}} \beta(\mathbf{z}, \mathbf{y}) \cdot \mathsf{add}_i(\mathbf{y}) \ . \tag{60}$$

The circuit for computing $\widetilde{\mathsf{add}}_i$ can compute the values $\{\mathsf{add}_i(\mathbf{y})\}_{\mathbf{y} \in \{0,1\}^{3m}}$. These computations may have high degree (and there are polynomially many of them), but they are completely independent of the circuit's input! For each $\mathbf{y} \in \{0,1\}^{3m}$, the circuit then computes the value $\beta_n(\mathbf{z}, \mathbf{y}) \cdot \mathsf{add}_i(\mathbf{y})$, a multilinear function of $\mathbf{z}$. Finally, taking the sum as in Equation (60) maintains multilinearity in the input.

To argue constructibility, we implement the log-space computations of the values $\{\mathsf{add}_i(\mathbf{y})\}_{\mathbf{y} \in \{0,1\}^{3m}}$ using the constructible circuits for log-space computations, as built in the proof of Lemma 5.10. Given these values, we can perform the multiplications and additions for Equation (60), while maintaining constructibility in the straightforward way. □

Theorem 5.13. *Let $\mathcal{L}$ be a language computable by logspace-uniform circuits of Depth $D(n)$ and polynomial size. The Bare-Bones protocol $(\mathsf{Gen}_{BB}, \mathsf{P}_{BB}, \mathsf{V}_{BB})$ from Section 5.3, instantiated with the sub-protocol for delegating $\widetilde{\mathsf{add}}_i, \widetilde{\mathsf{sub}}_i, \widetilde{\mathsf{mult}}_i$ on encoded inputs from Lemma 5.12, gives an adaptively sound publicly-verifiable non-interactive argument system for $\mathcal{L}$. For security parameter n and input length k:*

- *The challenge generator runs in time* $\mathrm{poly}(D, n)$.
- *The (honest) prover runs in time* $\mathrm{poly}(k, n)$.
- *The verifier runs in time* $k \cdot \mathrm{poly}(D, n)$.

Proof Sketch. There are two parts to the non-interactive argument. First, we run the Bare-Bones Protocol on the original (log-space uniform) circuit C, as specified by the (unique) multilinear extensions $\widetilde{\mathsf{add}}_i, \widetilde{\mathsf{sub}}_i, \widetilde{\mathsf{mult}}_i$ of the functions $\mathsf{add}_i, \mathsf{sub}_i, \mathsf{mult}_i$. This gives us a CRS CRS, a proof Π, and an encoded output:

$$y = \left(\left\{ [w_{i,j}]_{\mathbf{e}_{\ell+(i-1)\cdot m'+j}} \right\}_{i \in [D], j \in [m'-1]}, \left\{ [c_i^{+}]_{\delta' \cdot \mathbf{u}_i}, [c_i^{-}]_{\delta' \cdot \mathbf{u}_i}, [c_i^{\times}]_{\delta' \cdot \mathbf{u}_i} \right\}_{i \in [D]} \right) \ ,$$

where for every $i \in [D]$:

$$\mathbf{u}_i = \sum_{j \in [m'-1]} \mathbf{e}_{\ell+(i-1)\cdot m'+j} \ ,$$

and where verifier needs to verify that:

$$\left(\forall i \in [D] : \left(\widetilde{\mathsf{add}}_i(\mathbf{w}_i) = c_i^{+} \right) \wedge \left(\widetilde{\mathsf{sub}}_i(\mathbf{w}_i) = c_i^{-} \right) \wedge \left(\widetilde{\mathsf{mult}}_i(\mathbf{w}_i) = c_i^{\times} \right) \right)$$

This verification is performed using $3D$ further executions of the bare-bones protocol on the aobve encoded inputs, one for each claim about the values of $\mathsf{add}_i, \mathsf{sub}_i, \mathsf{mult}_i$. These are composed in parallel with each other and with the above initial run of the bare-bones protocol on the original circuit. The challenge generator generates a CRS CRS' for running the bare-bones protocol on the circuits for

$\mathsf{add}_i, \mathsf{sub}_i, \mathsf{mult}_i$. The prover then uses CRS' together with the encoded inputs (and outputs) in y to generate proofs and outputs:

$$\{\Pi_{\mathsf{add},i}, \Pi_{\mathsf{sub},i}, \Pi_{\mathsf{mult},i}\}_{i\in[D]} \quad , \quad \{y_{\mathsf{add},i}, y_{\mathsf{sub},i}, y_{\mathsf{mult},i}\}_{i\in[D]}$$

The verifier can verify the claims in these $3D$ proofs and outputs on her own. To do so, we use the fact proved in Lemma 5.12, that $\mathsf{add}_i, \mathsf{sub}_i, \mathsf{mult}_i$ are multilinear in the encoded inputs, and are also themselves constructible. Thus the prover and verifier can run the Bare-Bones protocol on the encoded inputs, and the verifier can compute low-degree extensions of their wiring predicates in polylogarithmic time and degree.

This gives a non-interactive argument for delegating the computation of the original circuit. The composed CRS is simply

$$(\mathsf{CRS}, \mathsf{CRS}'),$$

the composed proof contains:

$$\left(\Pi, y, \{\Pi_{\mathsf{add},i}, \Pi_{\mathsf{sub},i}, \Pi_{\mathsf{mult},i}, y_{\mathsf{add},i}, y_{\mathsf{sub},i}, y_{\mathsf{mult},i}\}_{i\in[D]}\right)$$

and the composed verifier simply verifies both the original protocol and the $3D$ resulting claims. □

6 Acknowledgements

We thank Zvika Brakerski for suggesting the secure sampling procedure in Remark 3.5, and for patiently and tirelessly answering our questions about graded encodings. We also thank Yael Kalai and Ron Rothblum for many helpful and illuminating conversations. Finally, thanks are also due to Mercury Cafe, for ambiance, for inspiration and for happy hour.

References

[ABOR00] William Aiello, Sandeep N. Bhatt, Rafail Ostrovsky, and Sivaramakrishnan Rajagopalan. Fast verification of any remote procedure call: Short witness-indistinguishable one-round proofs for np. In *ICALP*, pages 463–474, 2000.

[AIK10] Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. From secrecy to soundness: Efficient verification via secure computation. In *Automata, Languages and Programming, 37th International Colloquium, ICALP 2010, Bordeaux, France, July 6-10, 2010, Proceedings, Part I*, pages 152–163, 2010.

[BCCT13] Nir Bitansky, Ran Canetti, Alessandro Chiesa, and Eran Tromer. Recursive composition and bootstrapping for snarks and proof-carrying data. In *STOC*, pages 111–120, 2013.

[BCI$^+$13] Nir Bitansky, Alessandro Chiesa, Yuval Ishai, Rafail Ostrovsky, and Omer Paneth. Succinct non-interactive arguments via linear interactive proofs. In *TCC*, pages 315–333, 2013.

[BFLS91] László Babai, Lance Fortnow, Leonid A. Levin, and Mario Szegedy. Checking computations in polylogarithmic time. In *Proceedings of the 23rd Annual ACM Symposium on Theory of Computing, May 5-8, 1991, New Orleans, Louisiana, USA*, pages 21–31, 1991.

[BGT14] Nir Bitansky, Sanjam Garg, and Sidharth Telang. Succinct randomized encodings and their applications. Cryptology ePrint Archive, Report 2014/771, 2014. `http://eprint.iacr.org/`.

[BR14] Zvika Brakerski and Guy N. Rothblum. Virtual black-box obfuscation for all circuits via generic graded encoding. In *Theory of Cryptography - 11th Theory of Cryptography Conference, TCC 2014, San Diego, CA, USA, February 24-26, 2014. Proceedings*, pages 1–25, 2014.

[Bra14] Zvika Brakerski. Personal communication, 2014.

[BWZ14] Dan Boneh, David J. Wu, and Joe Zimmerman. Immunizing multilinear maps against zeroizing attacks. Cryptology ePrint Archive, Report 2014/930, 2014. `http://eprint.iacr.org/`.

[CHJV14] Ran Canetti, Justin Holmgren, Abhishek Jain, and Vinod Vaikuntanathan. Indistinguishability obfuscation of iterated circuits and ram programs. Cryptology ePrint Archive, Report 2014/769, 2014. `http://eprint.iacr.org/`.

[CHL$^+$14] Jung Hee Cheon, Kyoohyung Han, Changmin Lee, Hansol Ryu, and Damien Stehlé. Cryptanalysis of the multilinear map over the integers. Cryptology ePrint Archive, Report 2014/906, 2014. `http://eprint.iacr.org/`.

[CKV10] Kai-Min Chung, Yael Tauman Kalai, and Salil P. Vadhan. Improved delegation of computation using fully homomorphic encryption. In *Advances in Cryptology - CRYPTO 2010, 30th Annual Cryptology Conference, Santa Barbara, CA, USA, August 15-19, 2010. Proceedings*, pages 483–501, 2010.

[CLT13] Jean-Sébastien Coron, Tancrède Lepoint, and Mehdi Tibouchi. Practical multilinear maps over the integers. In *CRYPTO (1)*, pages 476–493, 2013.

[CLT14] Jean-Sebastien Coron, Tancrede Lepoint, and Mehdi Tibouchi. Cryptanalysis of two candidate fixes of multilinear maps over the integers. Cryptology ePrint Archive, Report 2014/975, 2014. `http://eprint.iacr.org/`.

[CTY11] Graham Cormode, Justin Thaler, and Ke Yi. Verifying computations with streaming interactive proofs. *PVLDB*, 5(1):25–36, 2011.

[DFH12] Ivan Damgård, Sebastian Faust, and Carmit Hazay. Secure two-party computation with low communication. In *Theory of Cryptography - 9th Theory of Cryptography Conference, TCC 2012, Taormina, Sicily, Italy, March 19-21, 2012. Proceedings*, pages 54–74, 2012.

[DLN$^+$04] Cynthia Dwork, Michael Langberg, Moni Naor, Kobbi Nissim, and Omer Reingold. Succinct proofs for NP and spooky interactions. Unpublished manuscript, 2004. `http://www.cs.bgu.ac.il/~kobbi/papers/spooky_sub_crypto.pdf`.

[GGH13] Sanjam Garg, Craig Gentry, and Shai Halevi. Candidate multilinear maps from ideal lattices. In *EUROCRYPT*, pages 1–17, 2013.

[GGH14] Craig Gentry, Sergey Gorbunov, and Shai Halevi. Graph-induced multilinear maps from lattices. Cryptology ePrint Archive, Report 2014/645, 2014. `http://eprint.iacr.org/`.

[GGP10] Rosario Gennaro, Craig Gentry, and Bryan Parno. Non-interactive verifiable computing: Outsourcing computation to untrusted workers. In *Advances in Cryptology - CRYPTO 2010, 30th Annual Cryptology Conference, Santa Barbara, CA, USA, August 15-19, 2010. Proceedings*, pages 465–482, 2010.

[GGPR13] Rosario Gennaro, Craig Gentry, Bryan Parno, and Mariana Raykova. Quadratic span programs and succinct nizks without pcps. In *Advances in Cryptology - EUROCRYPT 2013, 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Athens, Greece, May 26-30, 2013. Proceedings*, pages 626–645, 2013.

[GHMS14] Craig Gentry, Shai Halevi, Hemanta K. Maji, and Amit Sahai. Zeroizing without zeroes: Cryptanalyzing multilinear maps without encodings of zero. Cryptology ePrint Archive, Report 2014/929, 2014. `http://eprint.iacr.org/`.

[GKR08] Shafi Goldwasser, Yael Tauman Kalai, and Guy N. Rothblum. Delegating computation: interactive proofs for muggles. In *Proceedings of the 40th Annual ACM Symposium on Theory of Computing, Victoria, British Columbia, Canada, May 17-20, 2008*, pages 113–122, 2008.

[GLSW14] Craig Gentry, Allison Lewko, Amit Sahai, and Brent Waters. Indistinguishability obfuscation from the multilinear subgroup elimination assumption. Cryptology ePrint Archive, Report 2014/309, 2014. `http://eprint.iacr.org/`.

[Gro10] Jens Groth. Short pairing-based non-interactive zero-knowledge arguments. In *ASIACRYPT*, pages 321–340, 2010.

[GW11] Craig Gentry and Daniel Wichs. Separating succinct non-interactive arguments from all falsifiable assumptions. In *Proceedings of the 43rd Annual ACM Symposium on Theory of Computing*, pages 99–108, 2011.

[Kil92] Joe Kilian. A note on efficient zero-knowledge proofs and arguments. In *Proceedings of the 24th Annual ACM Symposium on Theory of Computing*, pages 723–732, 1992.

[KLW14] Venkata Koppula, Allison Bishop Lewko, and Brent Waters. Indistinguishability obfuscation for turing machines with unbounded memory. Cryptology ePrint Archive, Report 2014/925, 2014. `http://eprint.iacr.org/`.

[KR09] Yael Tauman Kalai and Ran Raz. Probabilistically checkable arguments. In *Advances in Cryptology - CRYPTO 2009, 29th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2009. Proceedings*, pages 143–159, 2009.

[KRR13] Yael Tauman Kalai, Ran Raz, and Ron D. Rothblum. Delegation for bounded space. In *STOC*, pages 565–574, 2013.

[KRR14] Yael Tauman Kalai, Ran Raz, and Ron D. Rothblum. How to delegate computations: the power of no-signaling proofs. In *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014*, pages 485–494, 2014.

[LFKN90] Carsten Lund, Lance Fortnow, Howard J. Karloff, and Noam Nisan. Algebraic methods for interactive proof systems. In *31st Annual Symposium on Foundations of Computer Science, St. Louis, Missouri, USA, October 22-24, 1990, Volume I*, pages 2–10, 1990.

[Lip12] Helger Lipmaa. Progression-free sets and sublinear pairing-based non-interactive zero-knowledge arguments. In *Theory of Cryptography - 9th Theory of Cryptography Conference, TCC 2012, Taormina, Sicily, Italy, March 19-21, 2012. Proceedings*, pages 169–189, 2012.

[LP14] Huijia Lin and Rafael Pass. Succinct garbling schemes and applications. Cryptology ePrint Archive, Report 2014/766, 2014. `http://eprint.iacr.org/`.

[Mic94] Silvio Micali. CS proofs (extended abstracts). In *35th Annual Symposium on Foundations of Computer Science, Santa Fe, New Mexico, USA, 20-22 November 1994*, pages 436–453, 1994.

[Nao03] Moni Naor. On cryptographic assumptions and challenges. In *Proceedings of the 23rd Annual International Cryptology Conference*, pages 96–109, 2003.

[PF79] Nicholas Pippenger and Michael J. Fischer. Relations among complexity measures. *J. ACM*, 26(2):361–381, 1979.

[PRV12] Bryan Parno, Mariana Raykova, and Vinod Vaikuntanathan. How to delegate and verify in public: Verifiable computation from attribute-based encryption. In *Theory of Cryptography - 9th Theory of Cryptography Conference, TCC 2012, Taormina, Sicily, Italy, March 19-21, 2012. Proceedings*, pages 422–439, 2012.

[RVW13] Guy N. Rothblum, Salil P. Vadhan, and Avi Wigderson. Interactive proofs of proximity: delegating computation in sublinear time. In *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013*, pages 793–802, 2013.

[WB13] Michael Walfish and Andrew J. Blumberg. Verifying computations without reexecuting them: from theoretical possibility to near-practicality. *Electronic Colloquium on Computational Complexity (ECCC)*, 20:165, 2013.