On Publicly Verifiable Delegation From Standard Assumptions

Yael Tauman Kalai [*] Omer Paneth [†] Lisa Yang [‡]

August 25, 2018

Abstract

We construct a publicly verifiable non-interactive delegation scheme for log-space uniform bounded depth computations in the common reference string (CRS) model, where the CRS is long (as long as the time it takes to do the computation).

The soundness of our scheme relies on the assumption that there exists a group with a bilinear map, such that given group elements g, h, h^t, h^{t^2}, it is hard to output g^a, g^b, g^c and h^a, h^b, h^c such that $a \cdot t^2 + b \cdot t + c = 0$, but a, b, c are not all zero.

Previously, such a result was only known under knowledge assumptions (or in the Random Oracle model), or under non-standard assumptions related to obfuscation or zero-testable homomorphic encryption.

We obtain our result by converting the interactive delegation scheme of Goldwasser, Kalai and Rothblum (J. ACM 2015) into a publicly verifiable non-interactive one. As a stepping stone, we give a publicly verifiable non-interactive version of the sum-check protocol of Lund, Fortnow, Karloff, Nisan (J. ACM 1992).

[*]Microsoft Research, email `yael@microsoft.com`.

[†]MIT, email `omerpa@mit.edu`. Supported by NSF Grants CNS-1350619 and CNS-1414119, and the Defense Advanced Research Projects Agency (DARPA) and the U.S. Army Research Office under contracts W911NF-15-C-0226 and W911NF-15-C-0236.

[‡]MIT, email `lisayang@mit.edu`. Part of this work was done at Microsoft Research.

Contents

1 Introduction

This work considers the setting where a prover wishes to prove the validity of a computational statement to a verifier who is too weak to perform the computation on its own. The exploration of such proofs have transformed computer science, giving rise to notions like NP proofs [Coo71, Kar72], interactive proofs [GMR88], multi-prover interactive proofs [BGKW88], probabilistically checkable proofs [FGL$^+$91, BFLS91, AS92, ALM$^+$98], and zero knowledge proofs [GMR88]. Such proofs have a pivotal role in secure cloud computing and other online applications such as anonymous distributed payment systems [BCG$^+$14].

In many applications, it is crucial that proofs are non-interactive and can be verified by anyone at any time. Concretely, we are interested in achieving the following two key properties in a proof system:

Non-interactive: Proofs can be generated offline , posted and then verified without any additional interaction between the prover and verifier.

Publicly verifiable: Proofs can be verified by anyone, and verification does not require knowing any secret information.

For example, the standard notion of an NP proof satisfies both of these properties.

In this work we focus on the following setting: given a program M and an input x, the prover would like to convince the verifier that $M(x) = y$. The verifier is given x, and a representation of M (for example, as a Turing machine), but it is either not capable or not willing to spend the computational resources required to evaluate $M(x)$. The prover will, therefore, provide the verifier with the output $y = M(x)$ together with a proof of correctness Π. Importantly, verifying this proof should be much easier then evaluating $M(x)$. Additionally, the resources required to generate the proof should not be much greater than the resources required to perform the computation. In the literature, such proofs are referred to as doubly-efficient proofs, proofs for delegating computation, or efficiently verifiable computation.

It is well known that, under standard complexity theoretic assumptions, non-interactive delegation schemes require both computational assumptions and a common reference string. The common reference string (CRS) is generated once and used to generate and verify proofs. While for every CRS there exist accepting proofs for false statements, it should be computationally infeasible to find any such proof given an honestly generated CRS.

Prior work. Many delegation schemes have been proposed in the literature. These schemes can roughly be divided into three groups:

Schemes from non-standard assumptions. Extensive work, starting from the seminal work of Micali [Mic94], and continuing with [Gro10, Lip12, DFH12, GGPR13, BCI$^+$13, BCCT13, BCC$^+$14], constructed publicly verifiable non-interactive delegation schemes that can even prove non-deterministic computations. However, the soundness of these schemes is proven either in the Random Oracle model [BR93], or based on non-standard hardness assumptions known as "knowledge assumptions".[1] Such assumptions have been criticized for being non-falsifiable and for yielding non-explicit security reductions [Nao03]. We mention that some of these works form the basis of several efficient implementations. These schemes, however, have long CRS proportional to the runtime of the computation.

Other schemes (for deterministic computations) are known based on non-standard assumptions related to obfuscation [CHJV15, KLW15, BGL$^+$15, CH16, ACC$^+$16, CCC$^+$16] or to zero-testable homomorphic encryption [PR17].

Designated verifier schemes. A line of works starting from [KRR13, KRR14], and continuing with [KP16, BHK17, BKK$^+$17, HR18], designed delegation schemes based on standard assumptions (such as computational private-information retrieval). However these schemes are not publicly verifiable.

[1] For example, the Knowledge-of-Exponent assumption [Dam92] asserts that any efficient adversary that is given two random generators (g, h) and outputs (g^z, h^z), must also "know" the exponent z.

The CRS is generated together with a secret verification key required to verify the proof. Moreover, an adversary that is able to learn if its proofs are accepted or not can eventually recover the secret verification key.

Interactive schemes. In the interactive setting we can achieve publicly verifiable schemes under standard assumptions, and even unconditionally. For example, [GKR15] and [RRR16] give interactive delegation schemes for bounded depth and bounded space computations with unconditional soundness. The work of [Kil92] gives a four message protocol from collision-resistant hashing, and [PRV12] use attribute-based encryption to delegate low-depth circuits in two messages in addition to a (hard to compute) CRS.

We therefore ask:

Do publicly verifiable non-interactive delegation schemes exist under standard assumptions?

1.1 This Work

In this work we construct a publicly verifiable non-interactive delegation scheme for low-depth circuits based on a hardness assumption on groups with bilinear maps.

The delegation scheme. Our scheme supports computations represented as log-space uniform circuits. That is, circuits that can be generated by a log-space Turing machine. The cost of verification (and proof length) is proportional to the depth of the circuit (and depends polynomially on the security parameter), and is independent of the circuit size. The length of the CRS is polynomially related to the size of the circuit, but verification only depends on a small portion of the CRS. Our proofs are adaptively sound meaning that soundness holds even if the statement being proved depends on the CRS.

The assumption. Our scheme is based on a group G of prime order $p \in \{0,1\}^\kappa$ (κ being the security parameter) equipped with a non-degenerate bilinear map $e : G \times G \to G_T$. We make the following assumption stated informally:

Assumption 1.1. *For a pair of random group elements $g, h \in G$ and a random $t \in \mathbb{Z}_p$, no polynomial-size adversary that is given g, h, h^t, h^{t^2} can output group elements*

$$g^a, g^b, g^c \;,\; h^a, h^b, h^c \qquad \text{such that} \qquad (a,b,c) \neq (0,0,0) \quad \text{and} \quad a \cdot t^2 + b \cdot t + c = 0 \;,$$

with non-negligible probability.

This arguably simple constant size assumption is falsifiable, and holds in the generic group model.

Informal theorem. *For every $d = d(\kappa)$ (κ being the security parameter) there exists a publicly verifiable non-interactive delegation scheme for any log-space uniform depth d computation, with verification time $d \cdot \mathsf{poly}(\kappa)$ and adaptive soundness and under Assumption 1.1.*

Our delegation scheme is obtained by removing interaction from the interactive proofs of Goldwasser, Kalai and Rothblum [GKR15] and the interactive sum-check protocol [LFKN92].

1.2 Our Techniques

Our starting point is the interactive delegation scheme of [GKR15] (henceforth denoted by GKR). This scheme is an interactive public-coin delegation scheme for low-depth log-space uniform circuits. We show how to convert this scheme into a publicly verifiable non-interactive delegation scheme.

At the heart of the GKR delegation scheme is the interactive sum-check protocol. Thus, as a first step, we show how to convert the interactive sum-check protocol into a publicly verifiable non-interactive protocol.

1.2.1 The Interactive Sum-Check Protocol.

We start with a quick overview of the celebrated interactive sum-check protocol of Lund, Fortnow, Karloff, and Nisan [LFKN92]. Let $\mathbb{F}$ be a field of size $\kappa^{\omega(1)}$ (where κ is the security parameter) and let $\mathbb{H} \subset \mathbb{F}$ be a subset of size $\mathsf{poly}(\kappa)$. Let $f : \mathbb{F}^\ell \to \mathbb{F}$ be a polynomial with individual degree $d = \mathsf{poly}(\kappa)$. The prover and verifier are both given a representation of f as a small arithmetic circuit C. The prover claims that the evaluations of f over the entire hypercube $\mathbb{H}^\ell$ sum to an element $A \in \mathbb{F}$:

$$\sum_{x_1,\ldots,x_\ell \in \mathbb{H}} f(x_1, \ldots, x_\ell) = A \ .$$

While computing the above sum naively requires time $O(|\mathbb{H}|^\ell \cdot |C|)$, the running time of the verifier is only $O(\ell \cdot d \cdot |\mathbb{H}| + |C|)$ (assuming constant-time field operations).

The protocol proceeds as follows: In the first round of the protocol, the prover computes a degree-d univariate polynomial S_1 that maps every $t \in \mathbb{F}$ to the partial sum of f where x_1 is fixed to t and the rest of the variables range over $\mathbb{H}^{\ell-1}$:

$$S_1(t) \equiv \sum_{x_2,\ldots,x_\ell \in \mathbb{H}} f(t, x_2, \ldots, x_\ell) \ .$$

The prover sends the polynomial S_1 represented as a list of d coefficients, and the verifier checks that

$$A = \sum_{x_1 \in \mathbb{H}} S_1(x_1) \ . \tag{1}$$

Next, the verifier samples a random element $t_1 \in \mathbb{F}$ and sends it to the prover. In the rest of the protocol the prover must convince the verifier that the polynomial $f(t_1, \cdot)$ indeed sums to $S_1(t_1)$ over $\mathbb{H}^{\ell-1}$:

$$S_1(t_1) = \sum_{x_2,\ldots,x_\ell \in \mathbb{H}} f(t_1, x_2, \ldots, x_\ell) \ . \tag{2}$$

In the second round of the protocol the prover computes the polynomial S_2 and sends it to the verifier:

$$S_2(t) \equiv \sum_{x_3,\ldots,x_\ell \in \mathbb{H}} f(t_1, t, x_3, \ldots, x_\ell) \ .$$

The verifier checks that S_2 indeed sums to $S_1(t_1)$ over $\mathbb{H}$:

$$S_1(t_1) = \sum_{x_2 \in \mathbb{H}} S_2(x_2) \ . \tag{3}$$

Then the verifier sends a random element $t_2 \in \mathbb{F}$ to the prover. Then it remains to verify that $f(t_1, t_2, \cdot)$ indeed sums to $S_2(t_2)$ over $\mathbb{H}^{\ell-2}$.

Finally, after ℓ such rounds have been completed, it remains to verify that:

$$S_\ell(t_\ell) = f(t_1, \ldots, t_\ell) \ . \tag{4}$$

The verifier checks this on its own by evaluating the circuit C on inputs $(t_1, \ldots, t_\ell)$.

Soundness. Consider a cheating prover claiming an incorrect sum $A' \neq A$ for $f(\cdot)$. To pass the verifier's test in (1), such a prover must send an incorrect polynomial $S_1' \not\equiv S_1$. If S_1' and S_1 are distinct polynomials of degree at most d, then they disagree on the random element t_1 with overwhelming probability at least $1 - d/|\mathbb{F}|$. If $S_1'(t_1) \neq S_1(t_1)$, then the cheating prover claims an incorrect sum for $f(t_1, \cdot)$. To pass the verifier's test in (3), the prover must send an incorrect polynomial $S_2' \not\equiv S_2$. By induction, with overwhelming probability, the prover must send an incorrect polynomial $S_\ell' \not\equiv S_\ell$ in the last round. Then the final claim in (4) must be incorrect causing the verifier to reject.

1.2.2 A Non-interactive Sum-Check Protocol.

Next we give a high-level overview of our publicly verifiable non-interactive sum-check protocol, which is the main component in our delegation scheme.

As a mental experiment, consider the following (unsound) non-interactive sum-check protocol: the elements $t_1, \ldots, t_\ell$ which are sent by the original sum-check verifier one at a time, are now published ahead of time in the CRS and the proof contains all of the original prover's answers in one message. This protocol is clearly unsound since a cheating prover that knows t_1 before sending its message can cheat by sending a polynomial $S'_1 \not\equiv S_1$ such that $S'_1(t_1) = S_1(t_1)$ and then compute the rest of the polynomials honestly.

Main idea. Our solution is based on the following observation: while the honest prover computes the polynomial S_1 independently of t_1, in a cheating proof where $S'_1 \not\equiv S_1$ and $S'_1(t_1) = S_1(t_1)$, the polynomial S_1 must depend on t_1, as t_1 is one of the d roots of $S'_1 - S_1$. In our solution the CRS includes a "cryptographic encoding" of t_1. Intuitively, this encoding still allows the verifier to check the proof, but it does not allow a cheating prover to find a non-zero polynomial that vanishes at t_1.

Specifically, we use a prime order group G, sample a random element $g \in G$ and encode t_1 in the exponent of g. To allow the verifier to evaluate the degree-d polynomial S_1 on t_1 we also encode the powers $t_1^2, \ldots, t_1^d$ in the exponent. This is sufficient since the verifier can obtain any linear combination of these powers in the exponent. For soundness we need to assume that given such encodings it is hard to find a non-zero polynomial that vanishes at t_1.

Subsequent rounds. In the second round of the interactive sum-check protocol, the prover sends the polynomial S_2:

$$S_2(t) \equiv \sum_{x_3,\ldots,x_\ell \in \mathbb{H}} f(t_1, t, x_2, \ldots, x_\ell) \ .$$

Since S_2 does depend on t_1, the (non-interactive) prover cannot compute S_2 in the clear. However, since the coefficients of S_2 can themselves be expressed as a degree-d polynomial in t_1, the prover can send encodings of the coefficients of S_2.

The next round bring additional challenges. The prover now needs to convinces the verifier that:

$$S_2(t_2) = \sum_{x_3,\ldots,x_\ell \in \mathbb{H}} f(t_1, t_2, x_3, \ldots, x_\ell) \ .$$

We again add encodings of the element t_2 and its powers to the CRS. However these encodings are no longer sufficient.

The prover needs to compute encoded coefficients of the polynomial S_3, but now these coefficients depend on t_1, t_2, and their products. To this end, we add encodings of all possible degree-d monomials in t_1 and t_2 to the CRS. On the verification side, the verifier needs to compute $S_2(t_2)$, but now both t_2 and S_2's coefficients are encoded. To this end, we require that the group G is equipped with a bilinear map $e : G \times G \to G_T$. The verifier can now obtain an encoding of $S_2(t_2)$ in the target group G_T and perform the consistency checks there.

The final round. Continuing the above construction through the last round, the CRS will contain encodings of all degree-d monomials in the elements $t_1, \ldots, t_\ell$. Therefore the length of our CRS grows with d^ℓ. However, note that the running time of the honest prover is exponential in ℓ anyways since it needs to sum f over all inputs. Looking ahead, for the sum-check parameters used in the GKR protocol, d^ℓ will be polynomial in the width of the circuit.

Recall that in the original interactive sum-check protocol, the verifier has a representation of f as a small arithmetic circuit C, and in the final round it evaluates C to obtain $f(t_1, \ldots, t_\ell)$. However, now the CRS only contains encoded monomials of $t_1, \ldots, t_\ell$, and the time required to compute an encoding of $f(t_1, \ldots, t_\ell)$ is proportional to the number of monomials in f, which may be exponentially larger than $|C|$.

The complexity of this final verification step, therefore, depends on the specific polynomial f in question. For example, for the polynomial f used in the GKR protocol we show how to perform the final verification step efficiently.

To allow for maximal flexibility in applications, we formulate our non-interactive sum-check without the final verification step. Instead, the prover will include an encoding of the value $B = f(t_1, \ldots, t_\ell)$ as part of the proof. If the prover claims that the sum of f is A', and the verifier accepts the proof, then it does not guarantee that A' is correct. Rather, we are only guaranteed that if A' is incorrect then the proof must contain an encoding of an incorrect value B', namely:

$$B' = f(t_1, \ldots, t_\ell) \quad \Rightarrow \quad A' = \sum_{x_1, \ldots, x_\ell \in \mathbb{H}} f(x_1, \ldots, x_\ell) \ .$$

Soundness. To prove soundness, we hope to argue that given the encoded monomials of $t_1, \ldots, t_\ell$, it is hard to generate encoded coefficients of a degree-d polynomial that vanishes on some t_i. Alas, finding such encodings is actually easy. For example, the polynomial $S(t) = t_1 \cdot t - t_1 \cdot t_2$ vanishes on t_2 and encodings of its coefficients t_1 and $t_1 \cdot t_2$ are given in the CRS.

To overcome this problem, we modify the protocol to use encodings under different random group elements $g_0, \ldots, g_\ell \in G$. For every $i > 0$, the CRS will contain all of the monomials of $t_1, \ldots, t_i$ encoded in the exponent of g_i. The proof will contain the coefficients of the polynomial S_i encoded under g_{i-1} (recall that these coefficients only depend on t_j for $j < i$).

Intuitively, we can obtain soundness under the assumption that given the encodings of powers of t_i under g_i, it is hard to find encodings of coefficients of a non-zero polynomial in the exponent of g_{i-1} such that the polynomial vanishes at t_i. However, it is still easy to find such encodings: simply encode such coefficients in the exponent of g_i. The same encodings viewed as exponents with base g_{i-1} encode a scaled version of the same polynomial and must vanish at t_i as well. Thus to obtain soundness, we make one last modification to the protocol: we ask for the proof to contain the coefficients of S_i encoded in the exponent of both g_{i-1} and g_i.

We prove the soundness of the non-interactive sum-check for polynomials of degree d under Assumption 1.2 which is stated informally below. We note that our delegation scheme only relies on sum-check for quadratic polynomials and therefore we can base its security on Assumption 1.1 which is a special case of Assumption 1.2 with $d = 2$.

Assumption 1.2 (Informal)**.** *For a pair of random group elements $g, h \in G$ and a random $t \in \mathbb{Z}_p$, no polynomial-size adversary that is given g and $h^{t^0}, \ldots, h^{t^d}$ can output group elements*

$$\{g^{a_i}, h^{a_i}\}_{0 \le i \le d} \qquad \text{such that} \qquad \exists i : \ a_i \neq 0 \quad \text{and} \quad \sum_{i=0}^{d} a_i \cdot t^i = 0 \ ,$$

with non-negligible probability.

1.2.3 The Non-interactive Delegation Scheme.

With a new sum-check protocol at hand, we proceed to describe our publicly verifiable non-interactive delegation scheme for low-depth log-space uniform circuits. The scheme mostly follows the blueprint of the interactive GKR protocol with a few important modifications and simplifications that we discuss next.

Roughly speaking, the interactive GKR protocol for delegating a computation $C(x)$ of depth D, is as follows. The prover evaluates $C(x)$, and for every $1 \le i \le D$ it encodes the i-th layer of the computation (with $i = 1$ being the output layer) as a multilinear polynomial V_i.[2] The output $C(x)$ is encoded by the value $V_1(s_1)$ for some particular point s_1.

[2]We note that the GKR protocol considers the more general case where V_i is a low-degree polynomial (not necessarily multilinear). In our work we use the multilinear version, since it results with a weaker assumption.

The protocol proceeds in $D-1$ phases, where in phase i, the prover makes a claim $V_i(s_i) = v_i$ on the value of a particular point in the i'th layer. This claim is then reduced to a new claim $V_{i+1}(s_{i+1}) = v_{i+1}$ on the value of a point in the $(i+1)$'st layer. After the final phase, the verifier can check the final claim $V_D(s_D) = v_D$ on the value of a point in the input layer on its own.

Oversimplifying, the value $V_i(s_i)$ can be expressed as a sum of the form:

$$V_i(s_i) = \sum_{x_i, x'_i} V_{i+1}(x_i) \cdot V_{i+1}(x'_i) \cdot f_C(x_i, x'_i) \ ,$$

where (x_i, x'_i) range over all pairs of points in layer $i+1$ and where f_C is some multilinear polynomial that encodes the structure of C's wires. The prover convinces the verifier that $V_i(s_i)$ is indeed the value of the sum above via the interactive sum-check protocol. Recall that in order to complete the sum-check, the verifier must compute $V_{i+1}(t_i) \cdot V_{i+1}(t'_i) \cdot f_C(t_i, t'_i)$ for a single pair of points (t_i, t'_i). The verifier cannot simply compute this value efficiently and, therefore, in the GKR protocol the verifier obtains this value in two steps:

- The prover sends the values $V_{i+1}(t_i) = u_i$ and $V_{i+1}(t'_i) = u'_i$. Then, the prover and verifier run a so-called "2-to-1" interactive protocol to reduce these two claims to a single claim of the form $V_{i+1}(s_{i+1}) = v_{i+1}$. This is the starting point for the protocol's $(i+1)$'st phase.
- The verifier needs to evaluate $f_C(t_i, t'_i)$ on the two claimed values. Recall that the verifier holds a log-space Turing machine that outputs C, but it does not have the time to evaluate this machine, write C and evaluate f_C. Instead it delegated this computation to the prover using the GKR protocol itself. Fortunately, since this computation is very simple (in log-space), it can be shown that its verification can be performed efficiently without further help from the prover.

To obtain our delegation scheme, we replace each of the D interactive sum-checks with our non-interactive sum-check. We also need to modify the two steps described above:

Avoiding the "2-to-1" protocol. One approach to dealing with the interactive "2-to-1" protocol is making it non-interactive as we do for sum-check. Instead we show how to avoid the use of a "2-to-1" protocol altogether both in the interactive and non-interactive settings.

Our solution is to let the prover make a claim on the value of two points in every phase instead of on just one point. Of course if we do this naively, each claim would be reduced to a claim about two new points in the next layer and the number of points would grow exponentially.

Our idea is to make sure that each of the claims in the i'th phase is reduced to a claim about the same two points in the next layer. We observe that in the sum-check protocol, the verifier's final evaluation points t_i and t'_i depend only on its own randomness and not on the prover's messages. Therefore, if we use the *same randomness* in the two parallel sum-check executions (or the same CRS in the non-interactive case) moving from layer i to layer $i+1$, then the verification in both sum-check executions will depend on the same pair of points (t_i, t'_i) in the next layer. We note that reusing the verifier's randomness in this manner does not increase the soundness error too much.

Avoiding the delegation of f_C. Recall that unlike the case in the interactive sum-check protocol, in our non-interactive sum-check, the complexity of the verifier's final evaluation grows with the number of monomials in f and not with $f's$ circuit complexity.

As a result, even when delegating simple log-space computations such as the evaluation of f_C, we can no longer show that verification is efficient. Instead we show how to avoid the need to delegate f_C altogether in the non-interactive setting.

The basic idea is to place (encodings of) the values $f_C(t_i, t'_i)$ in the CRS. As mentioned above, the values t_i and t'_i depend only on the CRS for the sum-check protocol. However, the function f_C depends on the delegated computation C and therefore the CRS cannot depend on it. To resolve this, we first consider delegation only for a fixed universal circuit U and include in the CRS (encodings of) the values $f_U(t_i, t'_i)$.

The circuit U outputs $C(x)$ given a description of a log-space Turing machine that generates a circuit C of depth D and any input x for C. We rely on the fact that there exists such circuits U with depth not much larger than D.

1.2.4 Discussion

While at first glance our approach may seem tailored to the sum-check and GKR protocol, we note that our techniques can be generalized and used to remove interaction (in the CRS model) from a wider class of "algebraic" protocols where the prover and verifier strategies are given by low-degree polynomials. We leave the task of formalizing this class of protocols to future work.

Next we discuss other known techniques for "compilers" that give non-interactive delegation and compare these techniques with our approach.

The Fiat-Shamir heuristic. Fiat and Shamir [FS86] suggested a heuristic for reducing interaction in constant-round public-coin protocols. Compared to our approach, their transformation is simple, highly efficient, applies to a wide class of protocols, and uses a random CRS with no structure. The main downside is that, as its name suggests, the security of the resulting protocol is heuristic and we do not know how to base it on any standard assumption. Very recently, [KRR16, CCRR18] show how to securely instantiate this heuristic under non-standard and exponentially-strong hardness assumptions. However, these instantiations fall short of reducing interaction in the GKR protocol which has a super-constant number of rounds.

The Kalai-Raz transformation. Kalai and Raz [KR09] show how to remove interaction from any public-coin interactive proof assuming the existence of a fully homomorphic encryption scheme with sub-exponential security. However, the resulting non-interactive protocol is not publicly verifiable: a secret decryption key is needed to verify the proof. Although our techniques are not as generic as those of [KR09], we achieve a publicly verifiable protocol.

It is instructive to compare our approach to that of [KR09]. Both protocols have the same high-level structure: for every round i, the verifier's messages up to the i-th round are given in the CRS under some cryptographic encoding. This encoding can be homomorphically manipulated so that the prover can compute its i-th answer under the encoding. However to maintain soundness, the encoding must not reveal too much about the verifier's future messages. The encoding in [KR09] is simply a fully homomorphic encryption, whereas we use an encoding based on groups with bilinear maps.

While the group structure and bilinear map enable public verification, they only support a limited set of homomorphic operations, namely quadratic polynomials. As a result, we can only handle protocols where both the prover and verifier strategies only evaluate quadratic polynomials over random elements. In particular, to handle the sum-check and GKR protocols, where the prover and verifier's strategies have degree above 2, we need to include all the monomials as part of the CRS which results in a long CRS.

Delegation from linear PCPs. The work of [BCI$^+$13] shows how to convert linear PCPs into publicly verifiable non-interactive delegation schemes. In a linear PCP the answers are obtained by applying a linear function to the queries. Their work considers linear PCPs where the verification has degree 2 in the queries and their answers. They transform such a PCP into a delegation protocol by encoding the verifier's PCP queries in the exponent of a group with a bilinear map. Our protocols can be described in a similar framework. Specifically, the interactive GKR prover just evaluates a low-degree polynomial over the verifier queries. By changing the verifier's queries to include all possible monomials, we can think of the prover strategy as a linear PCP. In our protocol the CRS contains an encodings of this extended query.

The crucial difference between our approach and that of [BCI$^+$13] is in the security proof. Their soundness proof relies on a non-standard and non-falsifiable knowledge assumption on groups with bilinear maps, whereas our security proof is based on a falsifiable assumption on such groups. However, we can only support deterministic bounded depth computations, whereas [BCI$^+$13] support any non-deterministic computation.

We note that the need to restrict our attention to deterministic computations is known to be inherent, given that our security proof is via a black-box reduction to a falsifiable assumption [GW11].

2 Preliminaries

2.1 Low Degree Extension

This subsection is taken almost verbatim from [GKR15]. Let $\mathbb{F}$ be a field and let $\mathbb{H} \subseteq \mathbb{F}$. Throughout this work with take $\mathbb{H} = \{0, 1\}$. We always assume (without loss of generality) that field operations can be performed in time that is poly-logarithmic in the field size, and space that is logarithmic in the field size. Fix an integer $m \in \mathbb{N}$. In what follows, we define the low degree extension of a k-element string $(w_0, w_1, \ldots, w_{k-1}) \in \mathbb{F}^k$ with respect to $\mathbb{F}, \mathbb{H}, m$, where $k \leq |\mathbb{H}|^m$.

Fix $\alpha : \mathbb{H}^m \to \{0, 1, \ldots, |\mathbb{H}^m| - 1\}$ to be any (efficiently computable) one-to-one function. In this paper, we take α to be the lexicographic order of $\mathbb{H}^m$. We can view $(w_0, w_1, \ldots, w_{k-1})$ as a function $W : \mathbb{H}^m \to \mathbb{F}$, where

$$W(z) = \begin{cases} w_{\alpha(z)} & \text{if } \alpha(z) \leq k-1 \\ 0 & \text{o.w.} \end{cases} \tag{5}$$

A basic fact is that there exists a unique extension of W into a function $\tilde{W} : \mathbb{F}^m \to \mathbb{F}$ (which agrees with W on $\mathbb{H}^m$: $\tilde{W}|_{\mathbb{H}^m} \equiv W$), such that $\tilde{W}$ is an m-variate polynomial of degree at most $|\mathbb{H}| - 1$ in each variable. Moreover, as is formally stated in the proposition below, the function $\tilde{W}$ can be expressed as

$$\tilde{W}(t_1, \ldots, t_m) = \sum_{i=0}^{k-1} \tilde{\beta}_i(t_1, \ldots, t_m) \cdot w_i,$$

where each $\tilde{\beta}_i : \mathbb{F}^m \to \mathbb{F}$ is an m-variate polynomial, that depends only on the parameters $\mathbb{H}$, $\mathbb{F}$, and m (and is independent of w), of size $\mathsf{poly}(|\mathbb{H}|, m)$ and of degree at most $|\mathbb{H}| - 1$ in each variable.

The function $\tilde{W}$ is called the *low degree extension* of $w = (w_0, w_1, \ldots, w_{k-1})$ with respect to $\mathbb{H}, \mathbb{F}, m$, and is denoted by $\mathrm{LDE}_{\mathbb{H},\mathbb{F},m}(w)$.

Proposition 2.1. *There exists a Turing machine that takes as input a field $\mathbb{F}$,[3] and a subset $\mathbb{H} \subseteq \mathbb{F}$, and an integer m. The machine runs in time $\mathsf{poly}(|\mathbb{H}|, m)$. It outputs the unique $2m$-variate polynomial $\tilde{\beta} : \mathbb{F}^m \times \mathbb{F}^m \to \mathbb{F}$ of degree $|\mathbb{H}| - 1$ in each variable (represented as an arithmetic circuit of degree $|\mathbb{H}| - 1$ in each variable), such that for every $(w_0, w_1, \ldots, w_{k-1}) \in \mathbb{F}^k$ with $k \leq |\mathbb{H}|^m$, and for every $z \in \mathbb{F}^m$,*

$$\tilde{W}(z) = \sum_{p \in \mathbb{H}^m} \tilde{\beta}(z, p) \cdot W(p), \tag{6}$$

where $W : \mathbb{H}^m \to \mathbb{F}$ is the function corresponding to $(w_0, w_1, \ldots, w_{k-1})$ as defined in Equation (5), and $\tilde{W} : \mathbb{F}^m \to \mathbb{F}$ is its low degree extension (i.e., the unique extension of $W : \mathbb{H}^m \to \mathbb{F}$ of degree at most $|\mathbb{H}| - 1$ in each variable).

Moreover, $\tilde{\beta}$ can be evaluated in time $\mathsf{poly}(|\mathbb{H}|, m)$. Namely, there exists a Turing machine with the above time bound, that takes as input parameters $\mathbb{H}, \mathbb{F}, m$ (as above), and a pair $(z, p) \in \mathbb{F}^m \times \mathbb{F}^m$, and outputs $\tilde{\beta}(z, p)$.

Claim 2.2. *There exists a Turing machine that takes as input a field $\mathbb{F}$, a subset $\mathbb{H} \subseteq \mathbb{F}$, an integer m, a sequence $w = (w_0, w_1, \ldots, w_{k-1}) \in \mathbb{F}^k$ such that $k \leq |\mathbb{H}|^m$, and a coordinate $z \in \mathbb{F}^m$. It outputs the value $\tilde{W}(z)$, where $\tilde{W}$ is the unique low-degree extension of w (with respect to $\mathbb{H}, \mathbb{F}, m$). The machine's running time is $|\mathbb{H}|^m \cdot \mathsf{poly}(|\mathbb{H}|, m)$.*

[3] Throughout this work, when we refer to a machine that takes as input a field, we mean the machine is given a short (poly-logarithmic in the field size) description of the field, that permits field operations to be computed in time that is poly-logarithmic in the field size and space that is logarithmic in the field size.

2.2 The Sum-Check Protocol

Let $\mathbb{F}$ be a field and let $\mathbb{H} \subseteq \mathbb{F}$. Let $f(x_1, \ldots, x_\ell)$ be a multivariate polynomial of degree at most d in each variable, with coefficients in $\mathbb{F}$, and let $v \in \mathbb{F}$. The sum-check protocol is an interactive protocol for proving that

$$\sum_{(h_1,\ldots,h_\ell)\in\mathbb{H}^\ell} f(h_1, \ldots, h_\ell) = v.$$

It consists of ℓ rounds of interaction between a prover and a verifier, and is defined below.

Protocol 2.3 (The Sum-Check Protocol)**.**

- The verifier samples random elements $t_1, \ldots, t_\ell \leftarrow \mathbb{F}$.
- In the ith round, the verifier sends t_{i-1} and the prover responds with $\alpha_{i,0}, \alpha_{i,1}, \ldots, \alpha_{i,d}$ such that

$$\sum_{j=0}^{d} \alpha_{i,j} x_i^j = \sum_{(h_{i+1},\ldots,h_\ell)\in\mathbb{H}^{\ell-i}} f(t_1, \ldots, t_{i-1}, x_i, h_{i+1}, \ldots, h_\ell).$$

  Where we define $t_0 = \emptyset$.

  The verifier performs the following checks depending on the round:

  – In the first round, the verifier checks that $\sum_{h\in\mathbb{H}} \sum_{j=0}^{d} \alpha_{1,j} h^j = v$.

  – In the ith round for $i = 2, \ldots, \ell$, the verifier checks that $\sum_{h\in\mathbb{H}} \sum_{j=0}^{d} \alpha_{i,j} h^j = \sum_{j=0}^{d} \alpha_{i-1,j} t_{i-1}^j$.

  – Finally, the verifier checks that $\sum_{j=0}^{d} \alpha_{\ell,j} t_\ell^j = f(t_1, \ldots, t_\ell)$.
- The verifier accepts if and only if all checks pass.

Lemma 2.4. *[LFKN92, Sha92] Let $f : \mathbb{F}^\ell \to \mathbb{F}$ be an ℓ-variate polynomial of degree d in each variable. The sum-check protocol (P, V) described in Protocol 2.3 satisfies the following properties.*

- **Completeness:** *If $\sum_{(h_1,\ldots,h_\ell)\in\mathbb{H}^\ell} f(h_1, \ldots, h_\ell) = v$ then*

$$\Pr\left[\left(P(f), V^f(v)\right) = 1\right] = 1.$$

- **Soundness:** *If $\sum_{(h_1,\ldots,h_\ell)\in\mathbb{H}^\ell} f(h_1, \ldots, h_\ell) \neq v$ then for every (unbounded) interactive Turing machine $\tilde{P}$,*

$$\Pr\left[\left(\tilde{P}(f), V^f(v)\right) = 1\right] \leq \frac{\ell d}{|\mathbb{F}|}.$$

- **Efficiency:** *$P(f)$ is an interactive Turing machine, and $V^f(v)$ is a probabilistic interactive Turing machine with oracle access to $f : \mathbb{F}^m \to \mathbb{F}$. The prover $P(f)$ runs in time $\leq \mathsf{poly}(|\mathbb{H}|^\ell)$.[4] The verifier $V^f(v)$ runs in time $\leq \mathsf{poly}(|\mathbb{H}|, \log|\mathbb{F}|, \ell, d)$, and queries the oracle f at a single point. The communication complexity is $\leq \mathsf{poly}(|\mathbb{H}|, \log|\mathbb{F}|, \ell, d)$, and the total number of bits sent from the verifier to the prover is $O(\ell \cdot \log|\mathbb{F}|)$. Moreover, this protocol is public-coin; i.e., all the messages sent by the verifier are truly random and consist of the verifier's random coin tosses.*

3 Publicly Verifiable Non-interactive Sum-Check

In this section we show how to convert the interactive sum-check protocol to a publicly verifiable non-interactive one. We start by defining this notion.

[4]Here we assume the prover's input is a description of the function f, from which f can be computed (on any input) in time $\leq \mathsf{poly}(|\mathbb{H}^\ell|)$.

3.1 Definitions

We specify the syntax of the publicly verifiable non-interactive sum-check protocol. As discussed in Section 1.2.2 of the introduction, this is a non-interactive encoded version of the sum-check protocol.

Notations We use the following notations throughout this work. We let κ denote the security parameter. We let $\mathbb{F} = \mathbb{F}_\kappa$ be a field of prime order $p = p_\kappa = \Theta(2^\kappa)$, and let $\mathbb{H} = \mathbb{H}_\kappa \subset \mathbb{F}_\kappa$ be a subset of size $\mathsf{poly}(\kappa)$. Let $G = G_\kappa$ be a group of order p_κ with a non-degenerate bilinear map $e : G \times G \to G_T$.[5] For $t \in \mathbb{F}$ and $g \in G$ let $\langle t \rangle_g$ denote the element $g^t \in G$. For a vector $\mathbf{t} = (t_1, \ldots, t_\ell) \in \mathbb{F}^\ell$ let $\langle \mathbf{t} \rangle_g$ denote the vector $\left(\langle t_i \rangle_g\right)_{i \in [\ell]}$.

The monomial encoding. The interface of the sum-check protocol uses a simple monomial encoding scheme. Let $d \in \mathbb{N}$ be a constant degree parameter. For a vector $\mathbf{s} = (s_1, \ldots, s_m) \in \mathbb{F}^m$ and a vector of degrees $\boldsymbol{\delta} = (\delta_1, \ldots, \delta_m) \in [0, d]^m$, let $\mathbf{s}^{\boldsymbol{\delta}}$ denote the product $\prod_{i \in [m]} s_i^{\delta_i}$.

Definition 3.1 (Monomial encoding)**.** The degree d monomial encoding of the vector $\mathbf{s} \in \mathbb{F}^m$ under an element $g \in G$ is denoted by $[\mathbf{s}]_g^d$ and it consists of the elements:

$$[\mathbf{s}]_g^d = \left\{ \left\langle \mathbf{s}^{\boldsymbol{\delta}} \right\rangle_g \right\}_{\boldsymbol{\delta} \in [0,d]^m} .$$

Fact 3.2. *Let $\mathbf{s} \in \mathbb{F}^m$ and $\mathbf{t} \in \mathbb{F}^\ell$ be vectors. Let $g \in G$ be an element, and let $h = \langle x \rangle_g$ for some $x \in \mathbb{F}$. Let $f : \mathbb{F}^m \to \mathbb{F}$ be a multivariate polynomial of individual degree at most d represented by a list of its coefficients. Then:*

Extention: *The encodings $[\mathbf{s}|\mathbf{t}]_g^d$ and $[\mathbf{t}|\mathbf{s}]_g^d$ can be efficiently computed given $[\mathbf{s}]_g^d$ and $\mathbf{t}$.*

Rerandomization: *The encoding $[\mathbf{s}]_h^d$ can be efficiently computed given $[\mathbf{s}]_g^d$ and x.*

Evaluation: *The element $\langle f(\mathbf{s}) \rangle_g$ can be efficiently computed given $[\mathbf{s}]_g^d$ and f.*

We proceed to define the notion of a publicly verifiable non-interactive sum-check protocol.

Syntax A publicly verifiable non-interactive sum-check protocol consists of a randomized algorithm $\mathsf{SC.S}$ and a pair of deterministic polynomial-time algorithms $(\mathsf{SC.P}, \mathsf{SC.V})$ with the following syntax.

- The setup algorithm $\mathsf{SC.S}$ takes as input:
  - The security parameter κ.
  - A parameter ℓ.
  - A monomial encoding $[\mathbf{s}]_g^d$ of a vector $\mathbf{s} \in \mathbb{F}^m$.

  The setup algorithm outputs:
  - A monomial encoding $[\mathbf{s}|\mathbf{t}]_h^d$ for a vector $\mathbf{t} \in \mathbb{F}^\ell$.
  - A pair of proving and verification keys $(\mathsf{pk}, \mathsf{vk})$.
- The prover algorithm $\mathsf{SC.P}$ takes as input:
  - The proving key pk.
  - A multivariate polynomial $f : \mathbb{F}^{m+\ell} \to \mathbb{F}$ of individual degree at most d represented by a list of its coefficients.

[5]We can use also an asymmetric map, and we chose to use a symmetric one only for the sake of simplicity.

The prover algorithm outputs:

- A pair of elements $(A, B) \in G$.
- A proof Π.

- The verifier algorithm $\mathsf{SC.V}$ takes as input:
  - The verification key vk.
  - The elements $(A, B) \in G$.
  - The proof Π.

  The verifier algorithm outputs an acceptance bit.

Definition 3.3 (Publicly Verifiable Non-interactive Sum-Check)**.** A publicly verifiable non-interactive sum-check protocol $(\mathsf{SC.S}, \mathsf{SC.P}, \mathsf{SC.V})$ satisfies the following properties:

Completeness. For every $\kappa \in \mathbb{N}$, multivariate polynomial $f : \mathbb{F}^{m+\ell} \to \mathbb{F}$ of individual degree at most d, vector $\mathbf{s} \in \mathbb{F}^m$ and element $g \in G$:

$$\Pr\left[\begin{array}{l} 1 = \mathsf{SC.V}(\mathsf{vk}, (A, B), \Pi) \\ B = \langle f(\mathbf{s}, \mathbf{t})\rangle_h \\ A = \left\langle \sum_{\mathbf{x}\in\mathbb{H}^\ell} f(\mathbf{s}, \mathbf{x})\right\rangle_g \end{array} \middle| \begin{array}{l} \left([\mathbf{s}|\mathbf{t}]^d_h, (\mathsf{pk}, \mathsf{vk})\right) \leftarrow \mathsf{SC.S}\left(\kappa, \ell, [\mathbf{s}]^d_g\right) \\ ((A, B), \Pi) \leftarrow \mathsf{SC.P}(\mathsf{pk}, f) \end{array}\right] = 1 \ .$$

Soundness. For every poly-size cheating prover P^* and polynomials $m = m(\kappa)$ and $\ell = \ell(\kappa)$, there exists a negligible function negl such that for every $\kappa \in \mathbb{N}$ and vector $\mathbf{s} \in \mathbb{F}^m$:

$$\Pr\left[\begin{array}{l} 1 = \mathsf{SC.V}(\mathsf{vk}, (A, B), \Pi) \\ B = \langle f(\mathbf{s}, \mathbf{t})\rangle_h \\ A \neq \left\langle \sum_{\mathbf{x}\in\mathbb{H}^\ell} f(\mathbf{s}, \mathbf{x})\right\rangle_g \end{array} \middle| \begin{array}{l} g \leftarrow G \\ \left([\mathbf{s}|\mathbf{t}]^d_h, (\mathsf{pk}, \mathsf{vk})\right) \leftarrow \mathsf{SC.S}\left(\kappa, \ell, [\mathbf{s}]^d_g\right) \\ (f, (A, B), \Pi) \leftarrow \mathsf{P}^*\left(g, [\mathbf{s}|\mathbf{t}]^d_h, (\mathsf{pk}, \mathsf{vk})\right) \end{array}\right] = \mathrm{negl}(\kappa) \ .$$

where f above is describing an $(m + \ell)$-variate polynomial of individual degree at most d.

Efficiency. In the above honest experiment, the running time of $\mathsf{SC.S}$ is $(d + |\mathbb{H}|)^{\ell+m} \cdot \mathsf{poly}(\kappa)$, and the length of the verification key vk and the length of the proof Π are bounded by $\ell \cdot \mathsf{poly}(\kappa)$.

3.2 The Protocol

In what follows we construct our publicly verifiable non-interactive sum-check protocol $(\mathsf{SC.S}, \mathsf{SC.P}, \mathsf{SC.V})$.

Protocol 3.4 (Publicly Verifiable Non-interactive Sum-Check)**.**

The setup algorithm $\mathsf{SC.S}$. The setup algorithm is given as input the security parameter κ, $\ell \in \mathbb{N}$, and an encoding $[\mathbf{s}]^d_g$. The setup algorithm proceeds as follows.

1. Set $g_0 = g$.
2. Sample random $x_1, \ldots, x_\ell \leftarrow \mathbb{F}$ and set $g_i = \langle x_i\rangle_g$.
3. Sample random $\mathbf{t} = (t_1, \ldots, t_\ell) \leftarrow \mathbb{F}^\ell$.
4. For $i \in [0, \ell]$ let $\mathbf{r}_i = (s_1, \ldots, s_m, t_1, \ldots, t_i) \in \mathbb{F}^{m+i}$.
5. Set
$$\mathsf{pk} = \left(1^{|\mathbb{H}|^\ell}, \left\{[\mathbf{r}_i]^d_{g_i}\right\}_{i\in[0,\ell]}\right) \quad , \quad \mathsf{vk} = \left\{[t_i]^d_{g_i}\right\}_{i\in[0,\ell]} \ ,$$
   where the encoding $[\mathbf{r}_i]^d_{g_i}$ is computed from $[\mathbf{s}]^d_g$, $\mathbf{t}$ and x_i, using Fact 3.2.

6. Output: $\left([\mathbf{r}]^d_{g_\ell}, (\mathsf{pk}, \mathsf{vk})\right)$.

The prover algorithm $\mathsf{SC.P}$**.** The prover is given as input the key

$$\mathsf{pk} = \left(1^{|\mathbb{H}|^\ell}, \left\{[\mathbf{r}_i]^d_{g_i}\right\}_{i\in[0,\ell]}\right) \quad ,$$

and a multivariate polynomial $f : \mathbb{F}^{m+\ell} \to \mathbb{F}$ of individual degree at most d represented by a list of its coefficients. The prover proceeds as follows: Let $f_\ell \equiv f$ and for every $i \in [0, \ell - 1]$, let $f_i : \mathbb{F}^{m+i} \to \mathbb{F}$ be the polynomial such that

$$f_i(x_1, \dots, x_{m+i}) \equiv \sum_{\mathbf{x}\in\mathbb{F}^{\ell-i}} f(x_1, \dots, x_{m+i}, \mathbf{x}) \ . \tag{7}$$

For $i \in [\ell]$ let $\left\{f_{i,\delta} : \mathbb{F}^{m+i-1} \to \mathbb{F}\right\}_{\delta\in[0,d]}$ be the polynomials such that

$$f_i(x_1, \dots, x_{m+i}) \equiv \sum_{\delta\in[0,d]} x^\delta_{m+i} \cdot f_{i,\delta}(x_1, \dots, x_{m+i-1}) \ . \tag{8}$$

Given the encodings in pk the prover computes the following elements, using Fact 3.2:

$$\begin{aligned}(A, B) &= \left(\langle f_0(\mathbf{r}_0)\rangle_{g_0}, \langle f_\ell(\mathbf{r}_\ell)\rangle_{g_\ell}\right)\\ \Pi &= \left\{C_{i,\delta} = \langle f_{i,\delta}(\mathbf{r}_{i-1})\rangle_{g_{i-1}}, C'_{i,\delta} = \langle f_{i,\delta}(\mathbf{r}_{i-1})\rangle_{g_i}\right\}_{i\in[\ell],\ \delta\in[0,d]} \quad .\end{aligned} \tag{9}$$

The prover outputs $((A, B), \Pi)$.

The verifier algorithm $\mathsf{SC.V}$**.** The verifier is given as input the following elements

$$\mathsf{vk} = \left\{[t_i]^d_{g_i}\right\}_{i\in[0,\ell]} \quad , \quad (A, B) \quad , \quad \Pi = \left\{C_{i,\delta}, C'_{i,\delta}\right\}_{i\in[\ell],\ \delta\in[0,d]} \quad .$$

From the input elements the verifier computes the following elements for every $i \in [\ell]$:

$$A_i = \sum_{\delta\in[0,d]} e\left(C'_{i,\delta}, \langle t^\delta_i\rangle_{g_i}\right) \quad , \quad B_{i-1} = \sum_{x\in\mathbb{H}} \sum_{\delta\in[0,d]} e\left(C_{i,\delta}, \langle x^\delta\rangle_{g_{i-1}}\right) \ , \tag{10}$$

and sets $A_0 = e(g_0, A)$ and $B_\ell = e(g_\ell, B)$. The verifier accepts if and only if:

$$\forall i \in [\ell], \delta \in [0, d] : \quad e(g_i, C_{i,\delta}) = e\left(g_{i-1}, C'_{i,\delta}\right) \tag{11}$$

$$\forall i \in [0, \ell] : \quad A_i = B_i \ . \tag{12}$$

3.3 Analysis of Protocol 3.4

In this section we prove the security of Protocol 3.4 based on the following assumption.

Assumption 3.5. *For any poly-size adversary* Adv *and every* $\kappa \in \mathbb{N}$*:*

$$\Pr\left[\begin{array}{l|l} \mathbf{c} = \mathbf{c}' & g, h \leftarrow G \\ \mathbf{c} \neq 0^d & t \leftarrow \mathbb{F} \\ \mathbf{c} \cdot (t^0, \dots, t^d) = 0 & \langle\mathbf{c}\rangle_g, \langle\mathbf{c}'\rangle_h \leftarrow \mathsf{Adv}\left(g, [t]^d_h\right)\end{array}\right] = \mathrm{negl}(\kappa) \ .$$

Theorem 3.6. *Under Assumption 3.5, Protocol 3.4 is a publicly verifiable non-interactive sum-check (Definition 3.3).*

In the rest of this section, we prove Theorem 3.6.

Completeness. Fix κ, a polynomial $f : \mathbb{F}^{m+\ell} \to \mathbb{F}$ of individual degree at most d, a vector $\mathbf{s} \in \mathbb{F}^m$ and an element $g \in G$. Consider the honest experiment

$$([\mathbf{s}|\mathbf{t}]_h^d, (\mathsf{pk}, \mathsf{vk})) \leftarrow \mathsf{SC.S}(\kappa, \ell, [\mathbf{s}]_g^d)$$
$$((A, B), \Pi) \leftarrow \mathsf{SC.P}(\mathsf{pk}, f)$$

We need to show that

$$1 = \mathsf{SC.V}(\mathsf{vk}, (A, B), \Pi)$$

and

$$(A, B) = \left(\left\langle \sum_{\mathbf{x} \in \mathbb{H}^\ell} f(\mathbf{s}, \mathbf{x}) \right\rangle_g, \langle f(\mathbf{s}, \mathbf{t}) \rangle_h \right) .$$

The latter condition follows directly by construction. We focus on showing that the verifier accepts. The verifier's test in Equation (11) passes since by Equation (9), for all $i \in [\ell]$ and $\delta \in [0, d]$:

$$e\left(g_i, C_{i,\delta}\right) = \langle f_{i,\delta}(\mathbf{r}_{i-1}) \rangle_{e(g_{i-1}, g_i)} = e\left(g_{i-1}, C'_{i,\delta}\right) \quad .$$

It remains to show that the verifier's test in Equation (12) passes. We have that for every $i \in [\ell]$:

$$\begin{aligned}
A_i &= \sum_{\delta \in [0,d]} e\left(C'_{i,\delta}, \langle t_i^\delta \rangle_{g_i}\right) \\
&= \sum_{\delta \in [0,d]} e\left(\langle f_{i,\delta}(\mathbf{r}_{i-1}) \rangle_{g_i}, \langle t_i^\delta \rangle_{g_i}\right) && \text{(By Equation (9))} \\
&= \left\langle \sum_{\delta \in [0,d]} f_{i,\delta}(\mathbf{r}_{i-1}) \cdot t_i^\delta \right\rangle_{e(g_i, g_i)} \\
&= \langle f_i(\mathbf{r}_{i-1}, t_i) \rangle_{e(g_i, g_i)} && \text{(By Equation (8))} \\
&= \langle f_i(\mathbf{r}_i) \rangle_{e(g_i, g_i)} \quad .
\end{aligned}$$

Similarly, for every $i \in [\ell]$:

$$\begin{aligned}
B_{i-1} &= \sum_{x \in \mathbb{H}} \sum_{\delta \in [0,d]} e\left(C_{i,\delta}, \langle x^\delta \rangle_{g_{i-1}}\right) \\
&= \sum_{x \in \mathbb{H}} \sum_{\delta \in [0,d]} e\left(\langle f_{i,\delta}(\mathbf{r}_{i-1}) \rangle_{g_{i-1}}, \langle x^\delta \rangle_{g_{i-1}}\right) && \text{(By Equation (9))} \\
&= \left\langle \sum_{x \in \mathbb{H}} \sum_{\delta \in [0,d]} f_{i,\delta}(\mathbf{r}_{i-1}) \cdot x^\delta \right\rangle_{e(g_{i-1}, g_{i-1})} \\
&= \left\langle \sum_{x \in \mathbb{H}} f_i(\mathbf{r}_{i-1}, x) \right\rangle_{e(g_{i-1}, g_{i-1})} && \text{(By Equation (8))} \\
&= \langle f_{i-1}(\mathbf{r}_{i-1}) \rangle_{e(g_{i-1}, g_{i-1})} && \text{(By Equation (7))}
\end{aligned}$$

By construction we also have that:

$$A_0 = e(g_0, A) = e\left(g_0, \langle f_0(\mathbf{r}_0)\rangle_{g_0}\right) = \langle f_0(\mathbf{r}_0)\rangle_{e(g_0,g_0)}$$
$$B_\ell = e(g_\ell, B) = e\left(g_\ell, \langle f_\ell(\mathbf{r}_\ell)\rangle_{g_\ell}\right) = \langle f_\ell(\mathbf{r}_\ell)\rangle_{e(g_\ell,g_\ell)} \quad .$$

Put together we get, as required, that for all $i \in [0, \ell]$, $A_i = B_i = \langle f_i(\mathbf{r}_i)\rangle_{e(g_i,g_i)}$.

Soundness. Let P^* be a poly-size cheating prover and let $m = m(\kappa)$ and $\ell = \ell(\kappa)$ be polynomials. Assume towards contradiction that there exists a polynomial p such that for infinitely many $\kappa \in \mathbb{N}$, there exist $\mathbf{s} \in \mathbb{F}^m$ such that:

$$\Pr\left[\begin{array}{l|l} 1 = \mathsf{SC.V}(\mathsf{vk}, (A, B), \Pi) & g \leftarrow G \\ B = \langle f(\mathbf{s}, \mathbf{t})\rangle_h & \left([\mathbf{s}|\mathbf{t}]_h^d, (\mathsf{pk}, \mathsf{vk})\right) \leftarrow \mathsf{SC.S}\left(\kappa, \ell, [\mathbf{s}]_g^d\right) \\ A \neq \left\langle \sum_{\mathbf{x}\in\mathbb{H}^\ell} f(\mathbf{s}, \mathbf{x})\right\rangle_g & (f, (A, B), \Pi) \leftarrow \mathsf{P}^*\left(g, [\mathbf{t}]_h^d, (\mathsf{pk}, \mathsf{vk})\right) \end{array}\right] \geq \frac{1}{p(\kappa)} \quad . \tag{13}$$

Next we describe an adversary Adv that breaks Assumption 3.5. That is, fix κ and $\mathbf{s} = (s_1, \ldots, s_m)$ such that Equation (13) holds. We show that:

$$\Pr\left[\begin{array}{l|l} \mathbf{c} = \mathbf{c}' & g, h \leftarrow G \\ \mathbf{c} \neq 0^d & t \leftarrow \mathbb{F} \\ \mathbf{c} \cdot (t^0, \ldots, t^d) = 0 & \langle \mathbf{c}\rangle_g, \langle \mathbf{c}'\rangle_h \leftarrow \mathsf{Adv}\left(g, [t]_h^d\right) \end{array}\right] \geq \frac{1}{\ell \cdot p(\kappa)} \quad . \tag{14}$$

We first describe Adv and then prove Equation (14). Given as input g and $[t]_h^d$, Adv proceeds as follows:

1. Sample $i^* \leftarrow [\ell]$.
2. For every $i \in [0, i^* - 2]$, sample $g_i \leftarrow G$.
3. Set $(g_{i^*-1}, g_{i^*}) = (g, h)$.
4. For every $i \in [i^* + 1, \ell]$, sample $x_i \leftarrow \mathbb{F}$ and set $g_i = \langle x_i\rangle_h$.
5. For every $i \in [0, \ell] \setminus \{i^*\}$, sample $t_i \leftarrow \mathbb{F}$.
6. Let $t_{i^*} = t$ and let $\mathbf{t} = (t_1, \ldots, t_\ell)$. Note that Adv is given $[t]_h^d$ but not t.
7. For $i \in [0, \ell]$ let $\mathbf{r}_i = (s_1, \ldots, s_m, t_1, \ldots, t_i) \in \mathbb{F}^{m+i}$.
8. Set

$$\mathsf{pk} = \left(1^{|\mathbb{H}|^\ell}, \left\{[\mathbf{r}_i]_{g_i}^d\right\}_{i\in[0,\ell]}\right) \quad , \quad \mathsf{vk} = \left\{[t_i]_{g_i}^d\right\}_{i\in[0,\ell]} \quad , \tag{15}$$

   where the encoding $[\mathbf{r}_i]_{g_i}^d$ is computed as follows:

   - For $i \in [0, i^* - 1]$, Adv knows $\mathbf{r}_i$ and therefore, it can directly compute $[\mathbf{r}_i]_{g_i}^d$.
   - For $i \in [i^*, \ell]$, Adv knows $\mathbf{s}$ and t_j for $j \neq i^*$ and therefore, it can compute $[\mathbf{r}_i]_h^d$ from $[t]_h^d$, following Fact 3.2.
   - For $i \in [i^* + 1, \ell]$, Adv knows x_i such that $g_i = \langle x_i\rangle_h$ and therefore, it can compute $[\mathbf{r}_i]_{g_i}^d$ from $[\mathbf{r}_i]_h^d$ following Fact 3.2.

9. Obtain the output of the cheating prover:

$$\left(f, (A,B), \Pi = \left\{C_{i,\delta}, C'_{i,\delta}\right\}_{i\in[\ell],\ \delta\in[0,d]}\right) \leftarrow \mathsf{P}^* \left(g_0, [\mathbf{r}]^d_{g_\ell}, (\mathsf{pk},\mathsf{vk})\right) \ . \tag{16}$$

10. Evaluate the honest prover strategy with the polynomial f and obtain the output:

$$\left((\bar{A},\bar{B}), \overline{\Pi} = \left\{\bar{C}_{i,\delta}, \bar{C}'_{i,\delta}\right\}_{i\in[\ell],\ \delta\in[0,d]}\right) \leftarrow \mathsf{SC.P}\left(\mathsf{pk}, f\right) \ . \tag{17}$$

11. Let $\mathbf{c}, \mathbf{c}', \bar{\mathbf{c}}, \bar{\mathbf{c}}' \in \mathbb{F}^{d+1}$ be vectors such that:

$$\begin{aligned} \langle \mathbf{c} \rangle_g &= \left(C_{i^*,\delta}\right)_{\delta\in[0,d]} \quad , \quad \langle \mathbf{c}' \rangle_h = \left(C'_{i^*,\delta}\right)_{\delta\in[0,d]} \ , \\ \langle \bar{\mathbf{c}} \rangle_g &= \left(\bar{C}_{i^*,\delta}\right)_{\delta\in[0,d]} \quad , \quad \langle \bar{\mathbf{c}}' \rangle_h = \left(\bar{C}'_{i^*,\delta}\right)_{\delta\in[0,d]} \ . \end{aligned} \tag{18}$$

12. Output $\langle \mathbf{c} - \bar{\mathbf{c}} \rangle_g, \langle \mathbf{c}' - \bar{\mathbf{c}}' \rangle_g$.

We observe that encodings computed by Adv in Equation (15) are distributed identically to those generated by $\mathsf{SC.S}$. That is:

$$\left\{ \left([\mathbf{r}]^d_{g_\ell}, (\mathsf{pk},\mathsf{vk})\right) \;\middle|\; \begin{array}{l} g,h \leftarrow G \\ t \leftarrow \mathbb{F} \\ \mathsf{Adv}\left(g, [t]^d_h\right) \end{array} \right\} \equiv \left\{ \mathsf{SC.S}\left(\kappa, \ell, [\mathbf{s}]^d_g\right) \;\middle|\; g \leftarrow G \right\} \tag{19}$$

By our assumption (Equation (13)) and Equation (19) it follows that the output of the cheating prover in Equation (16) satisfies:

$$\Pr\left[\begin{array}{l} 1 = \mathsf{SC.V}(\mathsf{vk}, (A,B), \Pi) \\ B = \langle f(\mathbf{s},\mathbf{t}) \rangle_h \\ A \neq \left\langle \sum_{\mathbf{x}\in\mathbb{H}^\ell} f(\mathbf{s},\mathbf{x}) \right\rangle_g \end{array} \;\middle|\; \begin{array}{l} g,h \leftarrow G \\ t \leftarrow \mathbb{F} \\ \mathsf{Adv}\left(g, [t]^d_h\right) \end{array} \right] \geq \frac{1}{p(\kappa)} \ .$$

By the completeness property of the sum-check protocol and Equation (19) it follows that the output of the honest prover in Equation (17) satisfies:

$$\Pr\left[\begin{array}{l} 1 = \mathsf{SC.V}(\mathsf{vk}, (\bar{A},\bar{B}), \overline{\Pi}) \\ \bar{B} = \langle f(\mathbf{s},\mathbf{t}) \rangle_h \\ \bar{A} = \left\langle \sum_{\mathbf{x}\in\mathbb{H}^\ell} f(\mathbf{s},\mathbf{x}) \right\rangle_g \end{array} \;\middle|\; \begin{array}{l} g,h \leftarrow G \\ t \leftarrow \mathbb{F} \\ \mathsf{Adv}\left(g, [t]^d_h\right) \end{array} \right] = 1 \ .$$

Put together we have that:

$$\Pr\left[\begin{array}{l} 1 = \mathsf{SC.V}(\mathsf{vk}, (A,B), \Pi) \\ 1 = \mathsf{SC.V}(\mathsf{vk}, (\bar{A},\bar{B}), \overline{\Pi}) \\ B = \bar{B} \\ A \neq \bar{A} \end{array} \;\middle|\; \begin{array}{l} g,h \leftarrow G \\ t \leftarrow \mathbb{F} \\ \mathsf{Adv}\left(g, [t]^d_h\right) \end{array} \right] \geq \frac{1}{p(\kappa)} \ .$$

Let $\{A_i, B_i\}_{i\in[0,\ell]}$ be the values computed by $\mathsf{SC.V}(\mathsf{vk}, (A,B), \Pi)$ in Equation (10) and let $\left\{\bar{A}_i, \bar{B}_i\right\}_{i\in[0,\ell]}$ be the values computed by $\mathsf{SC.V}(\mathsf{vk}, (\bar{A},\bar{B}), \overline{\Pi})$. Whenever the verifier's accept, by the test in Equation (12) we have that for all $i \in [0,\ell]$, $A_i = B_i$ and $\bar{A}_i = \bar{B}_i$ Therefore:

$$\Pr\left[\begin{array}{l} 1 = \mathsf{SC.V}(\mathsf{vk}, (A,B), \Pi) \\ 1 = \mathsf{SC.V}(\mathsf{vk}, (\bar{A},\bar{B}), \overline{\Pi}) \\ \exists i \in [\ell] : \begin{array}{l} B_{i-1} \neq \bar{B}_{i-1} \\ A_i = \bar{A}_i \end{array} \end{array} \;\middle|\; \begin{array}{l} g,h \leftarrow G \\ t \leftarrow \mathbb{F} \\ \mathsf{Adv}\left(g, [t]^d_h\right) \end{array} \right] \geq \frac{1}{p(\kappa)} \ .$$

Note that by Equation (19) the views of the cheating and honest provers in Equations (16) and (17) are independent of i^*. Therefore:

$$\Pr\left[\begin{array}{l|l} 1 = \mathsf{SC.V}(\mathsf{vk}, (A, B), \Pi) & g, h \leftarrow G \\ 1 = \mathsf{SC.V}(\mathsf{vk}, (\bar{A}, \bar{B}), \overline{\Pi}) & t \leftarrow \mathbb{F} \\ B_{i^*-1} \neq \bar{B}_{i^*-1} & \mathsf{Adv}\left(g, [t]_h^d\right) \\ A_{i^*} = \bar{A}_{i^*} & \end{array}\right] \geq \frac{1}{\ell \cdot p(\kappa)} \ . \tag{20}$$

Recall that:

$$\Pi = \left\{C_{i,\delta}, C'_{i,\delta}\right\}_{i\in[\ell],\ \delta\in[0,d]} \quad , \quad \overline{\Pi} = \left\{\bar{C}_{i,\delta}, \bar{C}'_{i,\delta}\right\}_{i\in[\ell],\ \delta\in[0,d]} \ .$$

By Equation (18), $\mathbf{c}, \mathbf{c}', \bar{\mathbf{c}}, \bar{\mathbf{c}}' \in \mathbb{F}^{d+1}$ are vectors such that:

$$\langle \mathbf{c} \rangle_g = \left(C_{i^*,\delta}\right)_{\delta\in[0,d]} \quad , \quad \langle \mathbf{c}' \rangle_h = \left(C'_{i^*,\delta}\right)_{\delta\in[0,d]} \ ,$$
$$\langle \bar{\mathbf{c}} \rangle_g = \left(\bar{C}_{i^*,\delta}\right)_{\delta\in[0,d]} \quad , \quad \langle \bar{\mathbf{c}}' \rangle_h = \left(\bar{C}'_{i^*,\delta}\right)_{\delta\in[0,d]} \ .$$

Whenever the verifier's accept, by the test in Equation (11) it holds that $\mathbf{c} = \mathbf{c}'$ and $\bar{\mathbf{c}} = \bar{\mathbf{c}}'$. In this case we can rewrite Equation (10) as:

$$A_{i^*} = \sum_{\delta\in[0,d]} e\left(C'_{i^*,\delta}, \langle t^\delta \rangle_h\right) = \left\langle \mathbf{c} \cdot (t^0, \ldots, t^d) \right\rangle_{e(h,h)} \ ,$$
$$\bar{A}_{i^*} = \sum_{\delta\in[0,d]} e\left(\bar{C}'_{i^*,\delta}, \langle t^\delta \rangle_h\right) = \left\langle \bar{\mathbf{c}} \cdot (t^0, \ldots, t^d) \right\rangle_{e(h,h)} \ ,$$
$$B_{i^*-1} = \sum_{x\in\mathbb{H}} \sum_{\delta\in[0,d]} e\left(C_{i^*,\delta}, \langle x^\delta \rangle_g\right) = \sum_{x\in\mathbb{H}} \left\langle \mathbf{c} \cdot (x^0, \ldots, x^d) \right\rangle_{e(h,h)} \ ,$$
$$\bar{B}_{i^*-1} = \sum_{x\in\mathbb{H}} \sum_{\delta\in[0,d]} e\left(\bar{C}_{i^*,\delta}, \langle x^\delta \rangle_g\right) = \sum_{x\in\mathbb{H}} \left\langle \bar{\mathbf{c}} \cdot (x^0, \ldots, x^d) \right\rangle_{e(h,h)} \ .$$

Combined with Equation (20) we get that:

$$\Pr\left[\begin{array}{l|l} (\mathbf{c} - \bar{\mathbf{c}}) = (\mathbf{c}' - \bar{\mathbf{c}}') & g, h \leftarrow G \\ (\mathbf{c} - \bar{\mathbf{c}}) \neq 0^d & t \leftarrow \mathbb{F} \\ (\mathbf{c} - \bar{\mathbf{c}}) \cdot (t^0, \ldots, t^d) = 0 & \mathsf{Adv}\left(g, [t]_h^d\right) \end{array}\right] \geq \frac{1}{\ell \cdot p(\kappa)} \ .$$

Since Adv outputs $\langle \mathbf{c} - \bar{\mathbf{c}} \rangle_g, \langle \mathbf{c}' - \bar{\mathbf{c}}' \rangle_h$ we reach a contradiction to Assumption 3.5.

4 Modified Interactive GKR Protocol

Notations. Let $C : \{0,1\}^n \to \{0,1\}$ be a boolean circuit with fan-in 2 of size S and depth D. For the sake of simplicity, add dummy gates so that each layer has the exact same size S. This increases the size of the circuit to at most S^2. Let $\mathbb{H}$ be a set of size at most $\mathsf{polylog}(S)$,[6] and let $m \in \mathbb{N}$ such that $|\mathbb{H}|^m = S$. Let $\ell = 3m$. We identify the indices $\{0, \ldots, S-1\}$ with the elements of $\mathbb{H}^m$. Let $\mathbb{F}$ be a field such that $\mathbb{H} \subset \mathbb{F}$ and such that $|\mathbb{F}| \geq m \cdot D \cdot \kappa$.[7]

Fix a circuit C as above, and an input $x \in \{0,1\}^n$. For every $i < D$, denote by $V_i \in \{0,1\}^S$ the value of the i'th layer of the circuit C on input x, where V_0 corresponds to the output layer. And denote by $V_D \in \{0,1\}^n$ the value of the input layer of the circuit C. For every $i \in [0, D-1]$, let $\tilde{V}_i : \mathbb{F}^m \to \mathbb{F}$ denote the low-degree extension of V_i, and let $\tilde{V}_D : \mathbb{F}^{m'} \to \mathbb{F}$ denote the low-degree extension of V_D, where m' is such that $\mathbb{H}^{m'} = n$.

[6]In the next section, we will use the protocol presented in this section with $|\mathbb{H}| = 2$.

[7]In the next section, we will use the protocol presented in this section with a field $\mathbb{F}$ such that $|\mathbb{F}| = 2^{\Theta(\kappa)}$.

Overview. The (interactive) GKR protocol, for delegating the computation $C(x) = 0$, consists of D phases where each phase runs a sum-check protocol. For $i \geq 1$, the i'th sum-check reduces the task of verifying the value of a single point in the low-degree extension of the $i-1$'st layer, $\tilde{V}_{i-1}(z_{i-1})$, to verifying the values of *two* points in the low-degree extension of the i'th layer, $\tilde{V}_i(x_i)$ and $\tilde{V}_i(y_i)$. Thus, there appears to be an exponential blowup in the number of points that the verifier will need to verify.

In [GKR15], they avoid this exponential blowup, by using a "2-to-1" reduction, where the verifier asks for the values of all the points on the line between x_i and y_i, and then chooses a random point z_i on the line, and only verifies the value of this single point $\tilde{V}_i(z_i)$.

We present a modification of the GKR protocol without this "2-to-1" reduction. Instead of continuing with a single sum-check to verify the value of $\tilde{V}_i(z_i)$, we continue with two sum-checks in parallel, to verify the values of both $\tilde{V}_i(x_i)$ and $\tilde{V}_i(y_i)$. The important observation is that we can reduce verifying these two points in the $i-1$'st layer to verifying only *two* points in the i'th layer. This is achieved by having the verifier use the same randomness in both sum-checks, and we argue that this does not compromise the security. This prevents the exponential blowup discussed above, and we will only have $O(D)$ sum-checks to verify the values of $\tilde{V}_0(0,\ldots,0)$ and $\{\tilde{V}_i(x_i), \tilde{V}_i(y_i)\}_{i\in[D-1]}$.

Before we describe our modified protocol in detail, we note some basic facts (from [GKR15]).

Facts. For any point $w \in \mathbb{H}^m$, it holds that

$$\tilde{V}_{i-1}(w) = \sum_{(x,y)\in\mathbb{H}^{2m}} \tilde{\mathrm{add}}_i(w,x,y)\cdot\left(\tilde{V}_i(x)+\tilde{V}_i(y)\right) + \tilde{\mathrm{mult}}_i(w,x,y)\cdot\tilde{V}_i(x)\cdot\tilde{V}_i(y),$$

where $\mathrm{add}_i : \mathbb{H}^{3m} \to \{0,1\}$ is the function that on input $(w,x,y) \in \mathbb{H}^{3m}$ outputs 1 if and only if the w gate in layer $i-1$ is an ADD gate whose children are gates x and y in layer i, and $\tilde{\mathrm{add}}_i : \mathbb{F}^{3m} \to \mathbb{F}$ is the low-degree extension of add_i. Similarly, $\mathrm{mult}_i : \mathbb{H}^{3m} \to \{0,1\}$ is the function that on input $(w,x,y) \in \mathbb{H}^{3m}$ outputs 1 if and only if the w gate in layer $i-1$ is a MULT gate whose children are gates x and y in layer i, and $\tilde{\mathrm{mult}}_i : \mathbb{F}^{3m} \to \mathbb{F}$ is the low-degree extension of mult_i. The functions add_D and mult_D are defined analogously except they take inputs $(w,x,y) \in \mathbb{H}^{m+2m'}$.

Therefore, for every point $z \in \mathbb{F}^m$ in the extension of the $i-1$'st layer,

$$\tilde{V}_{i-1}(z) = \sum_{(w,x,y)\in\mathbb{H}^{3m}} \tilde{\beta}(z,w)\cdot\left(\tilde{\mathrm{add}}_i(w,x,y)\cdot\left(\tilde{V}_i(x)+\tilde{V}_i(y)\right) + \tilde{\mathrm{mult}}_i(w,x,y)\cdot\tilde{V}_i(x)\cdot\tilde{V}_i(y)\right) \tag{21}$$

where $\tilde{\beta}$ is the polynomial defined in Proposition 2.1.

Denote the summand by

$$f_{i,z}(w,x,y) := \tilde{\beta}(z,w)\cdot\left(\tilde{\mathrm{add}}_i(w,x,y)\cdot\left(\tilde{V}_i(x)+\tilde{V}_i(y)\right) + \tilde{\mathrm{mult}}_i(w,x,y)\cdot\tilde{V}_i(x)\cdot\tilde{V}_i(y)\right). \tag{22}$$

Then proving the value of $\tilde{V}_{i-1}(z)$ is equivalent to proving the value of $\sum_{(w,x,y)\in\mathbb{H}^{3m}} f_{i,z}(w,x,y)$.

Note that $f_{i,z}(w,x,y)$ is of degree $\leq 2(|\mathbb{H}|-1)$ in each variable. In particular, if $\mathbb{H} = \{0,1\}$, then $f_{i,z}(w,x,y)$ is of degree ≤ 2 in each coordinate of (z,w,x,y). Jumping ahead, when we convert our modified GKR protocol to a non-interactive publicly verifiable variant, we indeed set $\mathbb{H} = \{0,1\}$.

4.1 The Protocol

We next present the protocol in detail. We assume (for simplicity) that the verifier has oracle access to the functions $\tilde{\mathrm{add}}_i$ and $\tilde{\mathrm{mult}}_i$.

Remark 4.1. We note that in [GKR15], it was similarly assumed that verifier has oracle access to $\tilde{\mathrm{add}}$ and $\tilde{\mathrm{mult}}$. This was refered to as the "barebones protocol." Then it was argued that for non-deterministic log-space (NL) computations, the verifier can indeed efficiently compute $\tilde{\mathrm{add}}$ and $\tilde{\mathrm{mult}}$ on his own, for $\tilde{\mathrm{add}}$

and $\tilde{\text{mult}}$ which are not *the* low-degree extensions of add and mult, but rather are *some* low degree extension (of slightly higher degree) of add and mult. Then, for log-space uniform bounded-depth circuits, $\tilde{\text{add}}$ and $\tilde{\text{mult}}$ are in NL and hence can be delegated.

In our setting, it suffcies to assume that the verifier has oracle access to $\tilde{\text{add}}$ and $\tilde{\text{mult}}$, since when we later convert this protocol into a non-interactive publicly verifiable one, we will include the relevant values of $\tilde{\text{add}}$ and $\tilde{\text{mult}}$ in the verification key vk.

Protocol 4.2 (Modified Interactive GKR Protocol)**.**

Generating the queries: For each $i \in [D]$, the verifier randomly samples a sum-check query $(w_{i-1}, x_i, y_i) \leftarrow \mathbb{F}^{3m}$. Let $x_0 = y_0 = 0^m$. Let $v_{0,x_0} = v_{0,y_0} = 0$.

The i'th phase ($1 \leq i \leq D-1$): The prover and verifier run two sum-check protocols, as described in Protocol 2.3, to prove that

$$v_{i-1,x_{i-1}} = \sum_{(w,x,y)\in\mathbb{H}^{3m}} f_{i,x_{i-1}}(w,x,y) \quad \text{and} \quad v_{i-1,y_{i-1}} = \sum_{(w,x,y)\in\mathbb{H}^{3m}} f_{i,y_{i-1}}(w,x,y),$$

where $x_{i-1}, y_{i-1} \in \mathbb{F}^m$ are determined by the random messages sent by the verifier in the previous phase. These two sum-check protocols are executed in parallel, where the verifier uses the *same randomness* in both. Namely, the messages of the verifier correspond to messages of a single sum-check, whereas the messages of the prover correspond to two (parallel) sum-checks.

Recall that the sum-check protocol reduces the task of verifying that $v_{i-1,x_{i-1}} = \sum_{(w,x,y)\in\mathbb{H}^{3m}} f_{i,x_{i-1}}(w,x,y)$ and $v_{i-1,y_{i-1}} = \sum_{(w,x,y)\in\mathbb{H}^{3m}} f_{i,y_{i-1}}(w,x,y)$, to the task of verifying that two elements given by the prover are equal to $f_{i,x_{i-1}}(w_{i-1},x_i,y_i)$ and $f_{i,y_{i-1}}(w_{i-1},x_i,y_i)$, respectively.

The verifier does not have oracle access to $f_{i,x_{i-1}}$ and $f_{i,y_{i-1}}$, and cannot efficiently compute these functions on his own. However, since $\tilde{\beta}$ is efficiently computable, verifying the values claimed for $f_{i,x_{i-1}}(w_{i-1},x_i,y_i)$ and $f_{i,y_{i-1}}(w_{i-1},x_i,y_i)$ reduces to verifying the (two) values claimed for

$$\tilde{\text{add}}_i(w_{i-1},x_i,y_i) \cdot \left(\tilde{V}_i(x_i) + \tilde{V}_i(y_i)\right) + \tilde{\text{mult}}_i(w_{i-1},x_i,y_i) \cdot \tilde{V}_i(x_i) \cdot \tilde{V}_i(y_i). \tag{23}$$

If these two values are not the same (as they should be) the verifier rejects.

The prover sends v_{i,x_i} as the value for $\tilde{V}_i(x_i)$ and v_{i,y_i} as the value for $\tilde{V}_i(y_i)$. Next, the verifier checks that v_{i,x_i} and v_{i,y_i} give the claimed value for Equation (23), and continues to the next phase with the values v_{i,x_i} and v_{i,y_i}. Recall that we assume that the verifier has oracle access to $\tilde{\text{add}}_i(w_{i-1},x_i,y_i)$ and $\tilde{\text{mult}}_i(w_{i-1},x_i,y_i)$, and thus can indeed efficiently heck that v_{i,x_i} and v_{i,y_i} give the claimed values for Equation (23).

The D'th phase: This phase proceeds as in the first $D-1$ phases except the verifier computes the values of $\tilde{V}_D(x_D)$ and $\tilde{V}_D(y_D)$ on his own and checks that these give the claimed values for Equation (23).

Theorem 4.3. *The protocol described above, denoted by $(P_{\text{GKR}}, V_{\text{GKR}})$, has the following properties.*

- **Completeness.** *For every circuit $C : \{0,1\}^n \to \{0,1\}$ of size S and depth D, characterized by $\{\tilde{\text{add}}_i, \tilde{\text{mult}}_i\}_{i\in[D]}$ and every $x \in \{0,1\}^n$ such that $C(x) = 0$,*

$$\Pr\left[\left(P_{\text{GKR}}(C,x), V_{\text{GKR}}^{\{\tilde{\text{add}}_i,\tilde{\text{mult}}_i\}_{i\in[D]}}(x)\right) = 1\right] = 1$$

- **Soundness.** *For every circuit $C : \{0,1\}^n \to \{0,1\}$ of size S and depth D, characterized by $\{\tilde{\text{add}}_i, \tilde{\text{mult}}_i\}_{i\in[D]}$, every $x \in \{0,1\}^n$ such that $C(x) \neq 0$, and every P^*,*

$$\Pr\left[\left(P^*(C,x), V_{\text{GKR}}^{\{\tilde{\text{add}}_i,\tilde{\text{mult}}_i\}_{i\in[D]}}(x)\right) = 1\right] \leq \frac{12(|\mathbb{H}|-1)mD}{\mathbb{F}}$$

- **Efficiency.** *The prover's run time is $\mathsf{poly}(S)$, the verifier's runtime is $(D+n)\cdot\mathsf{polylog}(S)$, and the communication complexity is $D\cdot\mathsf{polylog}(S)$.*

We do not use Theorem 4.3 to obtain our non-interactive publicly verifiable delegation scheme. Nevertheless we provide a proof sketch of Theorem 4.3 for the sake of completeness.

Proof Sketch. The completeness and efficiency guarantees follow immediately from the construction. Thus, in what follows we focus on proving soundness.

Suppose for contradiction that there exists a cheating prover P^* and there exists a circuit $C : \{0,1\}^n \to \{0,1\}$ and an input $x \in \{0,1\}^n$ such that $C(x) \neq 0$ and yet

$$\eta := \Pr\left[\left(P^*(C,x), V_{\text{GKR}}^{\{\tilde{\text{add}}_i, \tilde{\text{mult}}_i\}_{i\in[D]}}(x)\right) = 1\right] > \frac{12(|\mathbb{H}|-1)mD}{\mathbb{F}}.$$

Denote by CHEAT the event that the verifier accepts the interactive proof given by P^*. Note that if CHEAT holds then it must be the case that

$$(v_{0,x_0}, v_{0,y_0}) \neq (\tilde{V}_{0,x_0}, \tilde{V}_{0,y_0}) \text{ and } (v_{D,x_D}, v_{D,y_D}) = (\tilde{V}_{D,x_D}, \tilde{V}_{D,y_D}).$$

Therefore, it CHEAT holds it must be the case that there exists $i \in [D]$ such that

$$(v_{i-1,x_i-1}, v_{i-1,y_i-1}) \neq (\tilde{V}_{i-1,x_i-1}, \tilde{V}_{i-1,y_i-1}) \text{ and } (v_{i,x_i}, v_{i,y_i}) = (\tilde{V}_{i,x_i}, \tilde{V}_{i,y_i}).$$

We next argue that this contradicts the soundness of the sum-check protocol, as guaranteed in Lemma 2.4.

The above equation implies (without loss of generality) that if CHEAT holds then with probability at least 1/2 there exists $i \in [D]$ such that

$$v_{i-1,x_i-1} \neq \tilde{V}_{i-1,x_i-1} \text{ and } v_{i,x_i} = \tilde{V}_{i,x_i}.$$

This implies that there exists $i \in [D]$ such that

$$\Pr\left[\left(v_{i-1,x_i-1} \neq \tilde{V}_{i-1,x_i-1} \text{ and } v_{i,x_i} = \tilde{V}_{i,x_i}\right) \wedge V \text{ accepts}\right] \geq \frac{\eta}{2D}$$

where the randomness is over the random coin tosses of the verifier V.

By our assumption $\eta > \frac{12(|\mathbb{H}|-1)mD}{\mathbb{F}}$ and thus

$$\frac{\eta}{2D} > \frac{6(|\mathbb{H}|-1)m}{|\mathbb{F}|}.$$

Therefore, P^* convinces the sum-check verifier to accept an incorrect statement

$$v_{i-1,x_{i-1}} \neq \sum_{(w,x,y)\in\mathbb{H}^{3m}} f_{i,x_{i-1}}(w,x,y)$$

where $f_{i,x_{i-1}}(w,x,y)$ is a polynomial of degree $\leq 2(|\mathbb{H}|-1)$ in each variable, with probability greater than $\frac{2(|\mathbb{H}|-1)(3m)}{|\mathbb{F}|}$, contradicting the soundness of the (interactive) sum-check protocol (see Lemma 2.4).

□

5 Publicly Verifiable Non-interactive GKR

Notations. We use the same notations as those used in Section 4. Namely, we let $C : \{0,1\}^n \to \{0,1\}$ be a boolean circuit with fan-in 2 of size S and depth D. For the sake of simplicity, we add dummy gates so that each layer has the exact same size S. Let $\mathbb{H} = \{0,1\}$ and let $m \in \mathbb{N}$ such that $|\mathbb{H}|^m = S$. We identify the indices $\{0,\ldots,S-1\}$ with the elements of $\mathbb{H}^m$. Let $\mathbb{F}$ be a field such that $\mathbb{H} \subset \mathbb{F}$ and such that $|\mathbb{F}| = 2^{\Theta(\kappa)}$.

Overview. Recall that the modified GKR protocol presented in Section 4 consists of a series of sum-checks. The basic idea is to convert this interactive protocol into a non-interactive publicly verifiable one by converting the sum-checks into non-interactive publicly verifiable ones, as was done in Section 3.

At first it may seem that the soundness of our non-interactive protocol simply follows from the soundness of the non-interactive sum-check protocol, since our protocol simply consists of a sequence of sum-checks. However, a closer look reveals several subtleties. In particular, the security of our protocol relies on the fact that the underlying GKR protocol does not use a "2-to-1" reduction and that the underlying sum-check protocol is secure even when the sum-check value is only given in the exponent. This allows us to ensure consistency between the sum-checks.

5.1 Definitions

We now present the notion of a publicly verifiable non-interactive delegation scheme. We define it for log-space uniform circuits though it can be defined more generally for any class of circuits $\mathcal{C}$ where each $C \in \mathcal{C}$ can be represented by a (possibly non-uniform) Turing machine. Without loss of generality, it suffices to construct delegation schemes for proving computations of the form $C_n(x) = 0$ since to prove $C'_n(x) = y$, the prover can convert C' to C where $C_n(x, y)$ computes $C'_n(x)$ and checks equality with y.[8]
We proceed to define the notion of a publicly verifiable non-interactive delegation scheme for log-space uniform circuits.

Syntax A publicly verifiable non-interactive delegation scheme for log-space uniform circuits consists of a randomized algorithm S and a pair of deterministic polynomial-time algorithms (P, V) with the following syntax.

- The setup algorithm S takes as input:
  - The security parameter κ.
  - A parameter n.

  The setup algorithm outputs a pair of proving and verification keys $(\mathsf{pk}, \mathsf{vk})$.
- The prover algorithm P takes as input:
  - The proving key pk.
  - A log-space Turing machine M.
  - A string $x \in \{0,1\}^n$.

  The prover algorithm outputs a proof Π.
- The verifier algorithm V takes as input:
  - The verification key vk.
  - The log-space Turing machine M.
  - The string $x \in \{0,1\}^n$.
  - The proof Π.

  The verifier algorithm outputs an acceptance bit.

Definition 5.1 (Publicly Verifiable Non-interactive Delegation Scheme)**.** A publicly verifiable non-interactive delegation scheme $(\mathsf{S}, \mathsf{P}, \mathsf{V})$ for log-space uniform circuits satisfies the following properties:

Completeness. For every polynomial $n = n(\kappa)$, $\kappa \in \mathbb{N}$, log-space Turing machine M describing a family of circuits $C = \{C_n : \{0,1\}^n \to \{0,1\}\}_{n \in \mathbb{N}}$, and string $x \in \{0,1\}^n$ such that $C_n(x) = 0$:

$$\Pr\left[\ 1 = \mathsf{V}(\mathsf{vk}, (M, x), \Pi)\ \middle|\ \begin{array}{l} (\mathsf{pk}, \mathsf{vk}) \leftarrow \mathsf{S}\left(1^\kappa, n\right) \\ \Pi \leftarrow \mathsf{P}(\mathsf{pk}, (M, x)) \end{array}\right] = 1 \ .$$

Soundness. For every polynomial $n = n(\kappa)$ and poly-size cheating prover P^*, there exists a negligible function negl such that for every $\kappa \in \mathbb{N}$:

$$\Pr\left[\begin{array}{l} 1 = \mathsf{V}(\mathsf{vk}, (M, x), \Pi) \\ C_n(x) \neq 0 \end{array}\ \middle|\ \begin{array}{l} (\mathsf{pk}, \mathsf{vk}) \leftarrow \mathsf{S}\left(1^\kappa, n\right) \\ ((M, x), \Pi) \leftarrow \mathsf{P}^*\left(\mathsf{pk}, \mathsf{vk}\right) \end{array}\right] = \mathrm{negl}(\kappa) \ .$$

where M is a log-space Turing machine that on input 1^n outputs the description of a circuit $C_n : \{0,1\}^n \to \{0,1\}$ of size $S(n)$ such that $S(n(\kappa)) = \mathsf{poly}(\kappa)$.

[8] If the delegation scheme is for a class $\mathcal{C}$ and $C' \in \mathcal{C}$, then $C \in \mathcal{C}$.

Efficiency. For any log-space uniform family of circuits $C = \{C_n : \{0,1\}^n \to \{0,1\}\}_{n\in\mathbb{N}}$ described by a Turing machine M where C_n has size $S = S(n)$ and depth $D = D(n)$ and M has description length c, the prover's runtime is $\mathsf{poly}(S)$, the verifier's runtime is $(D+c+n)\cdot\mathsf{polylog}(S)$, and the communication complexity is $D\cdot\mathsf{polylog}(S)$.

Our main result is the following:

Theorem 5.2. *Under Assumption 3.5 for $d = 2$, there is a publicly verifiable non-interactive delegation scheme for log-space uniform circuits (Definition 5.1).*

First we construct a delegation scheme for any fixed family of circuits (not necessarily log-space uniform).

Lemma 5.3. *Under Assumption 3.5 for $d = 2$, Protocol 5.7 is a publicly verifiable non-interactive delegation scheme for any fixed family of circuits $U = \{U_n : \{0,1\}^n \to \{0,1\}\}_{n\in\mathbb{N}}$.*

The definition of a delegation scheme for a fixed family of circuits $U = \{U_n : \{0,1\}^n \to \{0,1\}\}_{n\in\mathbb{N}}$ is analogous to Definition 5.1, except the instance is an input string $x \in \{0,1\}^n$ for U_n and the verifier's runtime is $(D + n)\cdot\mathsf{polylog}(S)$.

To prove Theorem 5.2, we apply this delegation scheme to the following family of *universal circuits*.

Proposition 5.4. *For every $c \in \mathbb{N}$, every $D = D(n)$, and every $T = T(n)$, there exists a family of universal circuits $U = \{U_{n'} : \{0,1\}^{c+n} \to \{0,1\}\}_{n'\in\mathbb{N}}$ of size $\mathsf{poly}(T)$ and depth $O(D + \log T\cdot\log n)$, such that for any $n \in \mathbb{N}$, any $x \in \{0,1\}^n$, and any log-space Turing machine M with description length c that on input 1^n outputs the description of a depth $D(n)$ and size $T(n)$ circuit $C_n : \{0,1\}^n \to \{0,1\}$, it holds that*

$$U_{n'}(M, x) = C_n(x).$$

Proposition 5.4 follows immediately from the following two Lemmas.

Lemma 5.5. *[GKR15] For every $c \in \mathbb{N}$, every $S = S(n)$, and every $T = T(n)$, there exists a family of universal circuits $U = \{U_{n'} : \{0,1\}^{c+n} \to \{0,1\}\}_{n'\in\mathbb{N}}$ of size $\mathsf{poly}(2^S)$ and depth $O(S\cdot\log T)$ such that for any space S and time T Turing machine M with description length c that on input 1^n outputs the description of a circuit $C_n : \{0,1\}^n \to \{0,1\}$, it holds that*

$$U_{n'}(M, 1^n) = C_n.$$

Lemma 5.6. *[Val76] For a fixed constant $c \in \mathbb{N}$, every $D = D(n)$, and every $T = T(n)$, there exists a family of universal circuits $U = \{U_{n'} : \{0,1\}^{c\cdot T+n} \to \{0,1\}\}_{n'\in\mathbb{N}}$ of size $\mathsf{poly}(T)$ and depth $O(D + \log T\cdot\log n)$, such that for any $n \in \mathbb{N}$, any $x \in \{0,1\}^n$, and any circuit $C : \{0,1\}^n \to \{0,1\}$ of size T and depth D, it holds that*

$$U_{n'}(C, x) = C(x).$$

In Sections 5.2 and 5.3, we present our delegation scheme for any fixed family of circuits (Protocol 5.7) and prove Lemma 5.3.

5.2 The Protocol for a Fixed Family of Circuits

In this subsection, we present our publicly verifiable non-interactive protocol for delegating the computation of any fixed family of circuits $U = \{U_n\}_{n\in\mathbb{N}}$ with size $S = S(n)$, depth $D = D(n)$, and input length $n = n(\kappa)$ such that $S(n(\kappa)) = \mathsf{poly}(\kappa)$. Let $m = \log S$ and let $m' = \log n$.

Recall from Section 3.1 that for $g \in G$ and $t \in \mathbb{F}$, we denote the element $g^t \in G$ by $\langle t\rangle_g$. We also denote the degree d monomial encoding of a vector $\mathbf{s} \in \mathbb{F}^m$ by $[\mathbf{s}]_g^d := \left\{\left\langle \mathbf{s}^{\boldsymbol{\delta}}\right\rangle_g\right\}_{\boldsymbol{\delta}\in[0,d]^m}$.

We convert the modified interactive GKR protocol from Section 4 to a publicly verifiable protocol where $\mathbb{H} = \{0,1\}$ and $\mathbb{F}$ is a field of size $p = 2^{\theta(\kappa)}$.

Protocol 5.7 (Publicly Verifiable Non-interactive Delegation Scheme for a Fixed Family of Circuits)**.**

The setup algorithm S. The setup algorithm is given as input the security parameter 1^κ and an index n and generates public keys for proving and verifying the computation of the circuit U_n.

1. Let $\mathbf{s}_0 = (x_0, y_0)$ where $x_0 = y_0 = \vec{0} \in \mathbb{F}^m$. Sample a random element $h_0 \leftarrow G$. Compute $[\mathbf{s}_0]^2_{h_0}$.
2. For each layer $i \in [D]$, do the following:

   Sample a random element $r_{i-1} \in \mathbb{F}$ and let $g_{i-1} = \langle r_{i-1} \rangle_{h_{i-1}}$. Using the encoding $[\mathbf{s}_{i-1}]^2_{h_{i-1}}$, compute the encoding $[\mathbf{s}_{i-1}]^2_{g_{i-1}}$ (this rerandomization can be done efficiently by Fact 3.2).

   - If $i \in [D-1]$, compute
   $$([\mathbf{s}_{i-1}|\mathbf{t}_i]^2_{h_i}, (\mathsf{SC.pk}_i, \mathsf{SC.vk}_i)) \leftarrow \mathsf{SC.S}(1^\kappa, 1^{3m}, [\mathbf{s}_{i-1}]^2_{g_{i-1}}). \tag{24}$$
     Let $(w_{i-1}, x_i, y_i) = \mathbf{t}_i$ where $w_{i-1}, x_i, y_i \in \mathbb{F}^m$. Let $\mathbf{s}_i = (x_i, y_i)$.
   - If $i = D$, compute
   $$([\mathbf{s}_{D-1}|\mathbf{t}_D]^2_{h_D}, (\mathsf{SC.pk}_D, \mathsf{SC.vk}_D)) \leftarrow \mathsf{SC.S}(1^\kappa, 1^{m+2m'}, [\mathbf{s}_{D-1}]^2_{g_{D-1}}).$$
     Let $(w_{D-1}, x_D, y_D) = \mathbf{t}_D$ where $w_{D-1} \in \mathbb{F}^m$ and $x_D, y_D \in \mathbb{F}^{m'}$. Let $\mathbf{s}_i = (x_i, y_i)$.
3. For each layer $i \in [D]$, use $[\mathbf{s}_{i-1}|\mathbf{t}_i]^2_{h_i} = [x_{i-1}, y_{i-1}, w_{i-1}, x_i, y_i]^2_{h_i}$ to compute
$$\mathsf{ADD}_{i,x} = \left\langle \tilde{\beta}(x_{i-1}, w_{i-1}) \cdot \tilde{\mathrm{add}}_i(w_{i-1}, x_i, y_i) \right\rangle_{h_i} \quad \text{and} \quad \mathsf{MULT}_{i,x} = \left\langle \tilde{\beta}(x_{i-1}, w_{i-1}) \cdot \tilde{\mathrm{mult}}_i(w_{i-1}, x_i, y_i) \right\rangle_{h_i},$$
$$\mathsf{ADD}_{i,y} = \left\langle \tilde{\beta}(y_{i-1}, w_{i-1}) \cdot \tilde{\mathrm{add}}_i(w_{i-1}, x_i, y_i) \right\rangle_{h_i} \quad \text{and} \quad \mathsf{MULT}_{i,y} = \left\langle \tilde{\beta}(y_{i-1}, w_{i-1}) \cdot \tilde{\mathrm{mult}}_i(w_{i-1}, x_i, y_i) \right\rangle_{h_i} \tag{25}$$
   where $\tilde{\beta}$ is the polynomial defined in Section 2.1 and $\tilde{\mathrm{add}}_i$ and $\tilde{\mathrm{mult}}_i$ are the low-degree extensions of the add_i and mult_i functions defined in Section 4.[9]
4. Let
$$\mathsf{pk} = \left\{ \mathsf{SC.pk}_i, [\mathbf{s}_i]^1_{g_i} \right\}_{i \in [D]}$$
   and
$$\mathsf{vk} = \left(\{\mathsf{SC.vk}_i, \mathsf{ADD}_{i,x}, \mathsf{MULT}_{i,x}, \mathsf{ADD}_{i,y}, \mathsf{MULT}_{i,y}, g_i\}_{i \in [D]}, [\mathbf{s}_D]^1_{g_D} \right).$$
   Output
$$(\mathsf{pk}, \mathsf{vk}).$$

The prover algorithm P. The prover is given as input the key
$$\mathsf{pk} = \left\{ \mathsf{SC.pk}_i, [\mathbf{s}_i]^1_{g_i} \right\}_{i \in [D]}$$
and a string $x \in \{0,1\}^n$. The prover generates a proof for the statement
$$U_n(x) = 0$$
which is equivalent to the statement $\tilde{V}_0(\vec{0}) = 0$, where $\tilde{V}_0$ is the low-degree extension of the output layer of $U_n(x)$, as follows.

[9]The verifier checks consistency between layers by verifying Equation 22 in the exponent. The verifier does not have enough resources to compute $\tilde{\mathrm{add}}_i$ and $\tilde{\mathrm{mult}}_i$ on his own. Although by Proposition 2.1, $\tilde{\beta}$ is efficiently computable given coordinates in the clear, the verifier is only given coordinates in the exponent and thus cannot compute the products $\tilde{\beta} \cdot \tilde{add}_i$ and $\tilde{\beta} \cdot \tilde{mult}_i$ in the exponent. Therefore these values are provided in vk.

1. Perform the computation $U_n(x)$. Denote the i'th layer of the computation tableau by $V_i \in \{0,1\}^{2^m}$.
2. Recall that in Equation (22), we defined the polynomial

$$f_{i,z}(w,x,y) = \tilde{\beta}(z,w) \cdot \left(\tilde{\mathrm{add}}_i(w,x,y) \cdot \left(\tilde{V}_i(x) + \tilde{V}_i(y) \right) + \tilde{\mathrm{mult}}_i(w,x,y) \cdot \tilde{V}_i(x) \cdot \tilde{V}_i(y) \right).$$

   Let z, z' be variables in $\mathbb{F}^m$. For each layer $i \in [D]$, the prover can compute the coefficients of the polynomials

$$F_i(z,z',w,x,y) := f_{i,z}(w,x,y)$$
$$F'_i(z,z',w,x,y) := f_{i,z'}(w,x,y)$$

   by computing the coefficients of the polynomials $\tilde{\beta}$ and $\tilde{\mathrm{add}}_i$, $\tilde{\mathrm{mult}}_i$, and $\tilde{V}_i$.
3. For each layer $i \in [D]$, compute

$$((A_{i,x}, B_{i,x}), \mathsf{SC}.\Pi_{i,x}) \leftarrow \mathsf{SC.P}(\mathsf{SC.pk}_i, F_i)$$
$$((A_{i,y}, B_{i,y}), \mathsf{SC}.\Pi_{i,y}) \leftarrow \mathsf{SC.P}(\mathsf{SC.pk}_i, F'_i).$$

   Recall that by Equation (24), $\mathsf{SC.pk}_i$ allows a prover to prove the value of the sum of a function with the first $|\mathbf{s}_{i-1}| = 2m$ coordinates fixed to $\mathbf{s}_{i-1} = (x_{i-1}, y_{i-1})$ and the remaining $|\mathbf{t}_i| = 3m$ (or $m+2m'$ when $i = D$) coordinates ranging over $\mathbb{H}^{|\mathbf{t}_i|}$. Thus when $\mathsf{SC.P}$ outputs a sum-check proof for the polynomial $F_i(z,z',w,x,y)$, the coordinates of z and z' are fixed to $\mathbf{s}_{i-1} = (x_{i-1}, y_{i-1})$ so the proof is for the value of $\sum_{(w,x,y)\in\mathbb{H}^{3m}} f_{i,x_{i-1}}(w,x,y)$. Similarly, the sum-check proof for the polynomial $F'_i(z,z',w,x,y)$ is for the value of $\sum_{(w,x,y)\in\mathbb{H}^{3m}} f_{i,y_{i-1}}(w,x,y)$.
4. For each layer $i \in [D]$, let $(x_i, y_i) = \mathbf{s}_i$ and use $[\mathbf{s}_i]^1_{g_i}$ to compute

$$A_{i+1,x\cdot y} = \left\langle \tilde{V}_i(x_i) \cdot \tilde{V}_i(y_i) \right\rangle_{g_i}.$$

5. Output

$$\Pi = \{(A_{i,x}, B_{i,x}), \mathsf{SC}.\Pi_{i,x}, (A_{i,y}, B_{i,y}), \mathsf{SC}.\Pi_{i,y}, A_{i+1,x\cdot y}\}_{i\in[D]}.$$

The verifier algorithm V. The verifier is given as input the key

$$\mathsf{vk} = \left(\{\mathsf{SC.vk}_i, \mathsf{ADD}_{i,x}, \mathsf{MULT}_{i,x}, \mathsf{ADD}_{i,y}, \mathsf{MULT}_{i,y}, g_i\}_{i\in[D]}, [\mathbf{s}_D]^1_{g_D} \right),$$

a string $x \in \{0,1\}^n$ and a proof

$$\Pi = \{(A_{i,x}, B_{i,x}), \mathsf{SC}.\Pi_{i,x}, (A_{i,y}, B_{i,y}), \mathsf{SC}.\Pi_{i,y}, A_{i+1,x\cdot y}\}_{i\in[D]}.$$

The verifier performs the following checks and accepts if and only if all checks pass:

1. Check that

$$A_{1,x} = 1 \quad \text{and} \quad A_{1,y} = 1.$$

2. Let $(x_D, y_D) = \mathbf{s}_D$ and use $[\mathbf{s}_D]^1_{g_D}$ and x to compute

$$A_{D+1,x} = \left\langle \tilde{V}_D(x_D) \right\rangle_{g_D} \quad \text{and} \quad A_{D+1,y} = \left\langle \tilde{V}_D(y_D) \right\rangle_{g_D}$$

   where $\tilde{V}_D$ is the low-degree extension of the input layer of $U_n(x)$.

   For each $i \in [D]$, check that

$$e(g_i, A_{i+1,x\cdot y}) = e(A_{i+1,x}, A_{i+1,y}).$$

3. For each $i \in [D]$, check that

$$1 = \mathsf{SC.V}(\mathsf{SC.vk}_i, (A_{i,x}, B_{i,x}), \mathsf{SC.}\Pi_{i,x}) \quad \text{and} \quad 1 = \mathsf{SC.V}(\mathsf{SC.vk}_i, (A_{i,y}, B_{i,y}), \mathsf{SC.}\Pi_{i,y}).$$

4. For each $i \in [D]$, check that

$$e\left(B_{i,x}, g_i\right) = e\left(\mathsf{ADD}_{i,x}, A_{i+1,x} \cdot A_{i+1,y}\right) \cdot e\left(\mathsf{MULT}_{i,x}, A_{i+1,x\cdot y}\right),$$
$$e\left(B_{i,y}, g_i\right) = e\left(\mathsf{ADD}_{i,y}, A_{i+1,x} \cdot A_{i+1,y}\right) \cdot e\left(\mathsf{MULT}_{i,y}, A_{i+1,x\cdot y}\right).$$

5.3 Analysis of Protocol 5.7

Completeness. Fix a family of circuits $U = \{U_n\}_{n\in\mathbb{N}}$, a polynomial $n = n(\kappa)$, a security parameter $\kappa \in \mathbb{N}$, and a string $x \in \{0,1\}^n$ such that $U_n(x) = 0$. Consider the honest experiment

$$(\mathsf{pk}, \mathsf{vk}) \leftarrow \mathsf{S}\left(1^\kappa, n\right)$$
$$\Pi \leftarrow \mathsf{P}(\mathsf{pk}, x)$$

We show that

$$1 = \mathsf{V}(\mathsf{vk}, x, \Pi).$$

First we prove the following claims using the completeness of the sum-check protocol (Protocol 3.4).

Claim 5.8. *For every $i \in [D+1]$,*

$$A_{i,x} = \left\langle \tilde{V}_{i-1}(x_{i-1}) \right\rangle_{g_{i-1}} \quad \textit{and} \quad A_{i,y} = \left\langle \tilde{V}_{i-1}(y_{i-1}) \right\rangle_{g_{i-1}}.$$

Proof. Consider $i \in [D]$. By the generation of $\mathsf{SC.pk}_i$ in Equation (24) and by the completeness of Protocol 3.4,

$$A_{i,x} = \left\langle \sum_{(w,x,y)\in\mathbb{H}^{3m}} F_i(x_{i-1}, y_{i-1}, w, x, y) \right\rangle_{g_{i-1}} = \left\langle \sum_{(w,x,y)\in\mathbb{H}^{3m}} f_{i,x_{i-1}}(w, x, y) \right\rangle_{g_{i-1}} = \left\langle \tilde{V}_{i-1}(x_{i-1}) \right\rangle_{g_{i-1}}$$

where the last equality follows from Equations (21) and (22). The argument for $A_{i,y}$ is analogous.

For $i = D+1$, the verifier computes $A_{i,x}$ and $A_{i,y}$. □

Claim 5.9. *For every $i \in [D]$,*

$$B_{i,x} = \left\langle f_{i,x_{i-1}}(w_{i-1}, x_i, y_i) \right\rangle_{h_i} \quad \textit{and} \quad B_{i,y} = \left\langle f_{i,y_{i-1}}(w_{i-1}, x_i, y_i) \right\rangle_{h_i}.$$

Proof. By the completeness of Protocol 3.4,

$$B_{i,x} = \left\langle F_i(x_{i-1}, y_{i-1}, w_{i-1}, x_i, y_i) \right\rangle_{h_i} = \left\langle f_{i,x_{i-1}}(w_{i-1}, x_i, y_i) \right\rangle_{h_i}.$$

The argument for $B_{i,y}$ is analogous. □

Now we show that each of the verifier's checks in Protocol 5.7 passes.

1. Recall that $x_0 = y_0 = \vec{0} \in \mathbb{F}^m$. By Claim 5.8, $A_{1,x} = \left\langle \tilde{V}_0(\vec{0}) \right\rangle_{g_0} = \left\langle U_n(x) \right\rangle_{g_0} = 1$.

2. The honest prover computes $A_{i,x\cdot y} = \left\langle \tilde{V}_{i-1}(x_{i-1}) \cdot \tilde{V}_{i-1}(y_{i-1}) \right\rangle_{g_{i-1}}$ so by Claim 5.8,

$$e(g_{i-1}, A_{i,x\cdot y}) = \left\langle \tilde{V}_{i-1}(x_{i-1}) \cdot \tilde{V}_{i-1}(y_{i-1}) \right\rangle_{e(g_{i-1}, g_{i-1})} = e(A_{i,x}, A_{i,y}).$$

3. The sum-check verifier accepts by the completeness of Protocol 3.4.
4. By the definition of $\mathsf{ADD}_{i,x}$ and $\mathsf{MULT}_{i,x}$ in Equation (25), by Claim 5.8, and by the honest prover's computation of $A_{i,x\cdot y} = \left\langle \tilde{V}_{i-1}(x_{i-1}) \cdot \tilde{V}_{i-1}(y_{i-1}) \right\rangle_{g_{i-1}}$,
$$e\left(\mathsf{ADD}_{i,x}, A_{i+1,x} \cdot A_{i+1,y}\right) = \left\langle \tilde{\beta}(x_{i-1}, w_{i-1}) \cdot \tilde{\mathrm{add}}_i(w_{i-1}, x_i, y_i) \cdot \left(\tilde{V}_i(x_i) + \tilde{V}_i(y_i)\right) \right\rangle_{e(h_i,g_i)}$$
$$e\left(\mathsf{MULT}_{i,x}, A_{i+1,x\cdot y}\right) = \left\langle \tilde{\beta}(x_{i-1}, w_{i-1}) \cdot \tilde{\mathrm{mult}}_i(w_{i-1}, x_i, y_i) \cdot \tilde{V}_i(x_i) \cdot \tilde{V}_i(y_i) \right\rangle_{e(h_i,g_i)} .$$
By Claim 5.9,
$$e(B_{i,x}, g_i) = \left\langle f_{i,x_{i-1}}(w_{i-1}, x_i, y_i)\right\rangle_{e(h_i,g_i)} .$$
So by the definition of $f_{i,z}$ in Equation (22),
$$e\left(B_{i,x}, g_i\right) = e\left(\mathsf{ADD}_{i,x}, A_{i+1,x} \cdot A_{i+1,y}\right) \cdot e\left(\mathsf{MULT}_{i,x}, A_{i+1,x\cdot y}\right).$$
Similarly,
$$e\left(B_{i,y}, g_i\right) = e\left(\mathsf{ADD}_{i,y}, A_{i+1,x} \cdot A_{i+1,y}\right) \cdot e\left(\mathsf{MULT}_{i,y}, A_{i+1,x\cdot y}\right).$$

Soundness. Assume for the sake of contradiction that for the family of circuits $U = \{U_n\}_{n\in\mathbb{N}}$, there exists a poly-size algorithm P^*, a polynomial $n = n(\kappa)$ such that $S(n(\kappa)) = \mathsf{poly}(\kappa)$, and a polynomial p such that for infinitely many $\kappa \in \mathbb{N}$:
$$\Pr\left[\begin{array}{c} 1 = \mathsf{V}(\mathsf{vk}, x, \Pi) \\ U_n(x) \neq 0 \end{array} \middle| \begin{array}{c} (\mathsf{pk}, \mathsf{vk}) \leftarrow \mathsf{S}\left(1^\kappa, n\right) \\ (x, \Pi) \leftarrow \mathsf{P}^*\left(\mathsf{pk}, \mathsf{vk}\right) \end{array}\right] \geq \frac{1}{p(\kappa)} .$$

Using P^*, we construct a poly-size algorithm $\mathsf{Adv}_{\mathsf{SC}}$ that cheats with non-negligible probability in the sum-check protocol (Protocol 3.4), contradicting Theorem 3.6. First we prove some statements about the behavior of P^*.

Define the set of cheating answers, with respect to vk, as
$$\mathrm{CHEAT}_{\mathsf{vk}} = \left\{(x, \Pi) : (\mathsf{V}(\mathsf{vk}, x, \Pi) = 1) \bigwedge (U_n(x) \neq 0)\right\}.$$

Recall from the construction of Protocol 5.7 that vk defines the elements x_0, y_0, and $\{(w_{i-1}, x_i, y_i)\}_{i\in[D]}$.

For every $j \in [D]$, define the set of answers which cheat in the jth sum-check, with respect to vk, as
$$\mathrm{CHEAT}_{\mathsf{vk}}(j) = \left\{ (x, \Pi) \in \mathrm{CHEAT}_{\mathsf{vk}} : \begin{array}{l} \left(\forall i \in [j]\ A_{i,x} \neq \left\langle \tilde{V}_{i-1}(x_{i-1}) \right\rangle_{g_{i-1}} \bigvee A_{i,y} \neq \left\langle \tilde{V}_{i-1}(y_{i-1}) \right\rangle_{g_{i-1}} \right) \\ \bigwedge \left(B_{j,x} = \left\langle f_j^{x_{j-1}}(w_{j-1}, x_j, y_j)\right\rangle_{h_j}\right) \bigwedge \left(B_{j,y} = \left\langle f_j^{y_{j-1}}(w_{j-1}, x_j, y_j)\right\rangle_{h_j}\right) \end{array} \right\}.$$

Claim 5.10. *Consider* vk *and* (x, Π) *where* $\Pi = \{(A_{i,x}, B_{i,x}), \mathsf{SC}.\Pi_{i,x}, (A_{i,y}, B_{i,y}), \mathsf{SC}.\Pi_{i,y}, A_{i+1,x\cdot y},\}_{i\in[D]}$ *such that* $\mathsf{V}(\mathsf{vk}, x, \Pi) = 1$.

For any $j \in [D]$*, if for all* $i \in [j]$ *either*
$$A_{i,x} \neq \left\langle \tilde{V}_{i-1}(x_{i-1}) \right\rangle_{g_{i-1}} \quad \textit{or} \quad A_{i,y} \neq \left\langle \tilde{V}_{i-1}(y_{i-1}) \right\rangle_{g_{i-1}} ,$$
then either
$$A_{j+1,x} \neq \left\langle \tilde{V}_j(x_j) \right\rangle_{g_j} \quad \textit{or} \quad A_{j+1,y} \neq \left\langle \tilde{V}_j(y_j) \right\rangle_{g_j} \quad \textit{or} \quad (x, \Pi) \in \mathrm{CHEAT}_{\mathsf{vk}}(j).$$

Note that for $j = D$*, it must be the case that* $(x, \Pi) \in \mathrm{CHEAT}_{\mathsf{vk}}(j)$ *since* V *computes* $A_{D+1,x}$ *and* $A_{D+1,y}$*.*

Proof. Assume that both

$$A_{j+1,x} = \left\langle \tilde{V}_j(x_j) \right\rangle_{g_j} \quad \text{and} \quad A_{j+1,y} = \left\langle \tilde{V}_j(y_j) \right\rangle_{g_j}.$$

Since V checks consistency between the values given for $f_j^{x_{j-1}}(w_{j-1}, x_j, y_j)$ and $f_j^{y_{j-1}}(w_{j-1}, x_j, y_j)$ in the exponent of h_j (these are $B_{j,x}$ and $B_{j,y}$) with the values given for $\tilde{V}_j(x_j)$ and $\tilde{V}_j(y_j)$ in the exponent of g_j (these are $A_{j+1,x}$ and $A_{j+1,y}$), it must hold that

$$B_{j,x} = \left\langle f_j^{x_{j-1}}(w_{j-1}, x_j, y_j) \right\rangle_{h_j} \quad \text{and} \quad B_{j,y} = \left\langle f_j^{y_{j-1}}(w_{j-1}, x_j, y_j) \right\rangle_{h_j}.$$

Therefore, if for all $i \in [j]$ either

$$A_{i,x} \neq \left\langle \tilde{V}_{i-1}(x_{i-1}) \right\rangle_{g_{i-1}} \quad \text{or} \quad A_{i,y} \neq \left\langle \tilde{V}_{i-1}(y_{i-1}) \right\rangle_{g_{i-1}},$$

then by definition,

$$(x, \Pi) \in \mathrm{CHEAT}_{\mathsf{vk}}(j).$$

□

The following claim follows from inductively applying Claim 5.10.

Claim 5.11. *If $(x, \Pi) \in \mathrm{CHEAT}_{\mathsf{vk}}$, then $(x, \Pi) \in \mathrm{CHEAT}_{\mathsf{vk}}(j)$ for some $j \in [D]$.*

Recall that according our contradiction assumption,

$$\Pr\left[(x, \Pi) \in \mathrm{CHEAT}_{\mathsf{vk}} \;\middle|\; \begin{array}{l} (\mathsf{pk}, \mathsf{vk}) \leftarrow \mathsf{S}\left(1^\kappa, n\right) \\ (x, \Pi) \leftarrow \mathsf{P}^*(\mathsf{pk}, \mathsf{vk}) \end{array} \right] \geq \frac{1}{p(\kappa)}.$$

For every $j \in [D]$, define the set of prefixes

$$\mathrm{PREFIX}(j) = \left\{ \left\{ [\mathbf{s}_{i-1}]^2_{g_{i-1}}, [\mathbf{s}_{i-1}|\mathbf{t}_i]^2_{h_i}, (\mathsf{SC.pk}_i, \mathsf{SC.vk}_i) \right\}_{i \in [j-1]} : \Pr\left[(x, \Pi) \in \mathrm{CHEAT}_{\mathsf{vk}}(j)\right] \geq \frac{1}{2D \cdot p(\kappa)} \right\}.$$

where the probability is taken over the randomness used to generate $(\mathsf{pk}, \mathsf{vk}) \leftarrow \mathsf{S}(1^\kappa, n)$ and $(x, \Pi) \leftarrow \mathsf{P}^*(\mathsf{pk}, \mathsf{vk})$, conditioned on the first $j-1$ invocations of $\mathsf{SC.S}$ in Equation (24) taking as input $[\mathbf{s}_{i-1}]^2_{g_{i-1}}$ and outputting $\left([\mathbf{s}_{i-1}|\mathbf{t}_i]^2_{h_i}, (\mathsf{SC.pk}_i, \mathsf{SC.vk}_i)\right)$.

For any $(\mathsf{pk}, \mathsf{vk})$ and $j \in [0, \ldots, D]$, denote the jth prefix of $(\mathsf{pk}, \mathsf{vk})$ by

$$(\mathsf{pk}, \mathsf{vk})[j] := \left\{ [\mathbf{s}_{i-1}]^2_{g_{i-1}}, [\mathbf{s}_{i-1}|\mathbf{t}_i]^2_{h_i}, (\mathsf{SC.pk}_i, \mathsf{SC.vk}_i) \right\}_{i \in [j]}.$$

Claim 5.12. *For some $j \in [D]$, it holds that*

$$\Pr\left[(\mathsf{pk}, \mathsf{vk})[j-1] \in \mathrm{PREFIX}(j) \;\middle|\; \begin{array}{l} (\mathsf{pk}, \mathsf{vk}) \leftarrow \mathsf{S}(1^\kappa, n) \\ (x, \Pi) \leftarrow \mathsf{P}^*(\mathsf{pk}, \mathsf{vk}) \end{array} \right] \geq \frac{1}{2D \cdot p(\kappa)}.$$

Proof. Assume for the sake of contradiction that this inequality does not hold for any $j \in [D]$.

By Claim 5.11, there exists some $j^* \in [D]$ such that

$$\Pr\left[(x, \Pi) \in \mathrm{CHEAT}_{\mathsf{vk}}(j^*) \;\middle|\; \begin{array}{l} (\mathsf{pk}, \mathsf{vk}) \leftarrow \mathsf{S}\left(1^\kappa, n\right) \\ (x, \Pi) \leftarrow \mathsf{P}^*(\mathsf{pk}, \mathsf{vk}) \end{array} \right] \geq \frac{1}{D \cdot p(\kappa)}. \tag{26}$$

Let E be the event that $(\mathsf{pk}, \mathsf{vk})[j^* - 1] \in \mathrm{PREFIX}(j^*)$.

By conditioning on E,

$$\begin{aligned}\Pr[(x,\Pi)\in \mathrm{CHEAT}_{\mathsf{vk}}(j^*)] &= \Pr[E]\cdot\Pr[(x,\Pi)\in \mathrm{CHEAT}_{\mathsf{vk}}(j^*)|E] + \Pr[\neg E]\cdot \Pr[(x,\Pi)\in \mathrm{CHEAT}_{\mathsf{vk}}(j^*)|\neg E]\\ &\le \Pr[E] + \Pr[(x,\Pi)\in \mathrm{CHEAT}_{\mathsf{vk}}(j^*)|\neg E]\\ &< \frac{1}{2D\cdot p(\kappa)} + \frac{1}{2D\cdot p(\kappa)}\\ &= \frac{1}{D\cdot p(\kappa)}\end{aligned}$$

which contradicts Equation (26). □

By Claim 5.12, there exists some $j^*\in[D]$ and prefix

$$\left\{\left[\mathbf{s}^*_{i-1}\right]^2_{g^*_{i-1}}, \left[\mathbf{s}^*_{i-1}|\mathbf{t}^*_i\right]^2_{h^*_i}, (\mathsf{SC.pk}^*_i, \mathsf{SC.vk}^*_i)\right\}_{i\in[j^*-1]} \in \mathrm{PREFIX}(j^*).$$

If $j^*=1$, then let $x_0=y_0=\vec{0}\in\mathbb{F}^m$.
Otherwise, for every $i\in[j^*-1]$, let $(x_{i-1},y_{i-1},w_{i-1},x_i,y_i)=\mathbf{s}^*_{i-1}|\mathbf{t}^*_i$ where $x_{i-1},y_{i-1},w_{i-1},x_i,y_i\in\mathbb{F}^m$.
Let $\mathbf{s}=(x_{j^*-1},y_{j^*-1})$.
Consider the following experiment:

$$g\leftarrow G$$
$$\left([\mathbf{s}|\mathbf{t}]^2_h, (\mathsf{SC.pk},\mathsf{SC.vk})\right)\leftarrow \mathsf{SC.S}\left(\kappa,\ell,[\mathbf{s}]^2_g\right)$$

$\mathsf{Adv}_{\mathsf{SC}}(g,[\mathbf{s}|\mathbf{t}]^2_h,(\mathsf{SC.pk},\mathsf{SC.vk}))$, breaks the soundness of the sum-check protocol, as follows:

1. Generate a set of keys $(\mathsf{pk},\mathsf{vk})$ for Protocol 5.7, where

$$\mathsf{pk}=\{\mathsf{SC.pk}_i,[\mathbf{s}_i]^1_{g_i}\}_{i\in[D]}$$

$$\mathsf{vk}=\left(\{\mathsf{SC.vk}_i,\mathsf{ADD}_{i,x},\mathsf{MULT}_{i,x},\mathsf{ADD}_{i,y},\mathsf{MULT}_{i,y},g_i\}_{i\in[D]},[\mathbf{s}_D]^1_{g_D}\right),$$

as follows:

(a) For every $i\in[j^*-1]$, let $\mathbf{s}_{i-1}=\mathbf{s}^*_{i-1}$ and $g_{i-1}=g^*_{i-1}$. The prefix contains $[\mathbf{s}_{i-1}]^1_{g_{i-1}}$.
Let $h_i=h^*_i$, $\mathsf{SC.pk}_i=\mathsf{SC.pk}^*_i$, and $\mathsf{SC.vk}_i=\mathsf{SC.vk}^*_i$.

(b) Let $\mathbf{s}_{j^*-1}=\mathbf{s}$ and $g_{j^*-1}=g$. Compute $[\mathbf{s}_{j^*-1}]^1_{g_{j^*-1}}$.
Let $\mathsf{SC.pk}_{j^*}=\mathsf{SC.pk}$, $\mathsf{SC.vk}_{j^*}=\mathsf{SC.vk}$, $h_{j^*}=h$, and $(w_{j^*-1},x_{j^*},y_{j^*})=\mathbf{t}$, where if $j^*\in[D-1]$, then $w_{j^*-1},x_{j^*},y_{j^*}\in\mathbb{F}^m$, and if $j^*=D$, then $w_{j^*-1}\in\mathbb{F}^m$ and $x_{j^*},y_{j^*}\in\mathbb{F}^{m'}$.
Let $\mathbf{s}_{j^*}=(x_{j^*},y_{j^*})$.

(c) For each layer $i\in[j^*+1,\ldots,D]$, do the following:
Sample a random element $r_{i-1}\in\mathbb{F}$ and let $g_{i-1}=\langle r_{i-1}\rangle_{h_{i-1}}$. Using the encoding $[\mathbf{s}_{i-1}]^2_{h_{i-1}}$, compute the encoding $[\mathbf{s}_{i-1}]^2_{g_{i-1}}$.

- If $i\in[D-1]$, compute

$$([\mathbf{s}_{i-1}|\mathbf{t}_i]^2_{h_i},(\mathsf{SC.pk}_i,\mathsf{SC.vk}_i))\leftarrow\mathsf{SC.S}(1^\kappa,1^{3m},[\mathbf{s}_{i-1}]^2_{g_{i-1}}).$$

Let $(w_{i-1},x_i,y_i)=\mathbf{t}_i$ where $w_{i-1},x_i,y_i\in\mathbb{F}^m$. Let $\mathbf{s}_i=(x_i,y_i)$.

- If $i = D$, compute

$$([\mathbf{s}_{D-1}|\mathbf{t}_D]^2_{h_D}, (\mathsf{SC.pk}_D, \mathsf{SC.vk}_D)) \leftarrow \mathsf{SC.S}(1^\kappa, 1^{m+2m'}, [\mathbf{s}_{D-1}]^2_{g_{D-1}}).$$

Let $(w_{D-1}, x_D, y_D) = \mathbf{t}_D$ where $w_{D-1} \in \mathbb{F}^m$ and $x_D, y_D \in \mathbb{F}^{m'}$. Let $\mathbf{s}_i = (x_i, y_i)$.

(d) For each layer $i \in [D]$, use $[\mathbf{s}_{i-1}|\mathbf{t}_i]^2_{h_i} = [x_{i-1}, y_{i-1}, w_{i-1}, x_i, y_i]^2_{h_i}$ to compute

$$\mathsf{ADD}_{i,x}, \mathsf{MULT}_{i,x}, \mathsf{ADD}_{i,y}, \mathsf{MULT}_{i,y}$$

as defined in Equation (25).

2. Sample $(x, \Pi) \leftarrow \mathsf{P}^*(\mathsf{pk}, \mathsf{vk})$ where

$$\Pi = \{(A_{i,x}, B_{i,x}), \mathsf{SC}.\Pi_{i,x}, (A_{i,y}, B_{i,y}), \mathsf{SC}.\Pi_{i,y}, A_{i+1,x\cdot y}\}_{i\in[D]}\,.$$

3. Randomly sample $z^* \in \{x, y\}$. Let z, z' be variables in $\mathbb{F}^m$. If $z^* = x$, then let $F^*(z, z', w, x, y) = f_{j^*,z}(w, x, y)$ and otherwise let $F^*(z, z', w, x, y) = f_{j^*,z'}(w, x, y)$.

4. Output

$$(F^*, (A_{j^*,z^*}, B_{j^*,z^*}), \mathsf{SC}.\Pi_{j^*,z^*})\,.$$

The distribution of $(\mathsf{pk}, \mathsf{vk})$ is identical to the output distribution of $\mathsf{S}(1^\kappa, n)$ conditioned on the first $j^* - 1$ sum-check keys equaling $\left\{[\mathbf{s}^*_{i-1}|\mathbf{t}^*_i]^2_{h_i}, (\mathsf{SC.pk}^*_i, \mathsf{SC.vk}^*_i)\right\}_{i\in[j^*-1]}$.

By the definition of PREFIX(j^*),

$$\Pr[(x, \Pi) \in \mathrm{CHEAT}_{\mathsf{vk}}(j^*)] \geq \frac{1}{2D \cdot p(\kappa)}.$$

For any $(x, \Pi) \in \mathrm{CHEAT}_{\mathsf{vk}}(j^*)$,

$$\left(A_{j^*,z} \neq \left\langle \tilde{V}_{j^*-1}(z_{j^*-1}) \right\rangle_{g_{j^*-1}}\right) \bigwedge \left(B_{j^*,z} = \left\langle f^{z_{j^*-1}}_{j^*}(w_{j^*-1}, x_{j^*}, y_{j^*})\right\rangle_{h_{j^*}}\right)$$

for some $z \in \{x, y\}$ by definition.

With probability at least $1/2$, $\mathsf{Adv}_{\mathsf{SC}}$ samples $z^* = z$. Thus, we conclude that with probability at least $1/(4D \cdot p(\kappa))$, $\mathsf{Adv}_{\mathsf{SC}}$ outputs a cheating proof for the sum-check protocol. Since $S(n(\kappa)) = \mathsf{poly}(\kappa)$, this contradicts Theorem 3.6.

Efficiency. Since Protocol 5.7 consists of $O(D)$ sum-checks, the prover's runtime is $\mathsf{poly}(S)$ and the communication complexity is $D \cdot \mathsf{polylog}(S)$. This follows from the efficiency guarantees of the sum-check protocol (Protocol 3.4) and the efficiency of rerandomization and evaluation with respect to a monomial encoding (Fact 3.2).

For the verifier, it takes $D \cdot \mathsf{polylog}(S)$ time to verify the sum-check proofs and perform the additional verifications to check consistency between sum-checks. It takes $n \cdot \mathsf{polylog}(S)$ time to compute encodings of points in the low-degree extension $\tilde{V}_D$ of the input layer since $\tilde{V}_D$ is a polynomial in $m' = \log n$ variables so it consists of n monomials. □

Acknowledgements We would like to thank Dan Boneh for helpful discussions regarding bilinear groups.

References

[ACC$^+$16] Prabhanjan Ananth, Yu-Chi Chen, Kai-Min Chung, Huijia Lin, and Wei-Kai Lin. Delegating RAM computations with adaptive soundness and privacy. In *Theory of Cryptography - 14th International Conference, TCC 2016-B, Beijing, China, October 31 - November 3, 2016, Proceedings, Part II*, pages 3–30, 2016.

[ALM$^+$98] Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy. Proof verification and the hardness of approximation problems. *J. ACM*, 45(3):501–555, 1998.

[AS92] Sanjeev Arora and Shmuel Safra. Probabilistic checking of proofs; A new characterization of NP. In *33rd Annual Symposium on Foundations of Computer Science, Pittsburgh, Pennsylvania, USA, 24-27 October 1992*, pages 2–13, 1992.

[BCC$^+$14] Nir Bitansky, Ran Canetti, Alessandro Chiesa, Shafi Goldwasser, Huijia Lin, Aviad Rubinstein, and Eran Tromer. The hunting of the SNARK. *IACR Cryptology ePrint Archive*, 2014:580, 2014.

[BCCT13] Nir Bitansky, Ran Canetti, Alessandro Chiesa, and Eran Tromer. Recursive composition and bootstrapping for SNARKS and proof-carrying data. In Boneh et al. [BRF13], pages 111–120.

[BCG$^+$14] Eli Ben-Sasson, Alessandro Chiesa, Christina Garman, Matthew Green, Ian Miers, Eran Tromer, and Madars Virza. Zerocash: Decentralized anonymous payments from bitcoin. In *2014 IEEE Symposium on Security and Privacy, SP 2014, Berkeley, CA, USA, May 18-21, 2014*, pages 459–474, 2014.

[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*, pages 21–31, 1991.

[BGKW88] Michael Ben-Or, Shafi Goldwasser, Joe Kilian, and Avi Wigderson. Multi-prover interactive proofs: How to remove intractability assumptions. In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing*, pages 113–131, 1988.

[BGL$^+$15] Nir Bitansky, Sanjam Garg, Huijia Lin, Rafael Pass, and Sidharth Telang. Succinct randomized encodings and their applications. *IACR Cryptology ePrint Archive*, 2015:356, 2015.

[BHK17] Zvika Brakerski, Justin Holmgren, and Yael Tauman Kalai. Non-interactive delegation and batch NP verification from standard computational assumptions. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017*, pages 474–482, 2017.

[BKK$^+$17] Saikrishna Badrinarayanan, Yael Tauman Kalai, Dakshita Khurana, Amit Sahai, and Daniel Wichs. Non-interactive delegation for low-space non-deterministic computation. Cryptology ePrint Archive, Report 2017/1250, 2017. `https://eprint.iacr.org/2017/1250`.

[BR93] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In Dorothy E. Denning, Raymond Pyle, Ravi Ganesan, Ravi S. Sandhu, and Victoria Ashby, editors, *ACM Conference on Computer and Communications Security*, pages 62–73. ACM, 1993.

[BRF13] Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors. *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013*. ACM, 2013.

[CCC$^+$16] Yu-Chi Chen, Sherman S. M. Chow, Kai-Min Chung, Russell W. F. Lai, Wei-Kai Lin, and Hong-Sheng Zhou. Cryptography for parallel RAM from indistinguishability obfuscation. In *ITCS*, pages 179–190. ACM, 2016.

[CCRR18] Ran Canetti, Yilei Chen, Leonid Reyzin, and Ron D. Rothblum. Fiat-shamir and correlation intractability from strong kdm-secure encryption. In *Advances in Cryptology - EUROCRYPT 2018 - 37th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tel Aviv, Israel, April 29 - May 3, 2018 Proceedings, Part I*, pages 91–122, 2018.

[CH16] Ran Canetti and Justin Holmgren. Fully succinct garbled RAM. In *ITCS*, pages 169–178. ACM, 2016.

[CHJV15] Ran Canetti, Justin Holmgren, Abhishek Jain, and Vinod Vaikuntanathan. Succinct garbling and indistinguishability obfuscation for RAM programs. In *STOC*, pages 429–437. ACM, 2015.

[Coo71] Stephen A. Cook. The complexity of theorem-proving procedures. In *Proceedings of the 3rd Annual ACM Symposium on Theory of Computing, May 3-5, 1971, Shaker Heights, Ohio, USA*, pages 151–158, 1971.

[Dam92] Ivan Damgård. Towards practical public key systems secure against chosen ciphertext attacks. In *Proceedings of CRYPTO91*, pages 445–456, 1992.

[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.

[FGL$^+$91] Uriel Feige, Shafi Goldwasser, László Lovász, Shmuel Safra, and Mario Szegedy. Approximating clique is almost np-complete (preliminary version). In *FOCS*, pages 2–12. IEEE Computer Society, 1991.

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

[GGPR13] Rosario Gennaro, Craig Gentry, Bryan Parno, and Mariana Raykova. Quadratic span programs and succinct nizks without pcps. In *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.

[GKR15] Shafi Goldwasser, Yael Tauman Kalai, and Guy N. Rothblum. Delegating computation: Interactive proofs for muggles. *J. ACM*, 62(4):27, 2015.

[GMR88] Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM J. Comput.*, 17(2):281–308, 1988.

[Gro10] Jens Groth. Short pairing-based non-interactive zero-knowledge arguments. In *ASIACRYPT*, volume 6477 of *Lecture Notes in Computer Science*, pages 321–340. Springer, 2010.

[GW11] Craig Gentry and Daniel Wichs. Separating succinct non-interactive arguments from all falsifiable assumptions. In *Proceedings of the Forty-third Annual ACM Symposium on Theory of Computing*, STOC '11, pages 99–108, New York, NY, USA, 2011. ACM.

[HR18] J. Holmgren and R. Rothblum. Delegating computations with (almost) minimal time and space overhead. 2018.

[Kar72] Richard M. Karp. Reducibility among combinatorial problems. In *Proceedings of a symposium on the Complexity of Computer Computations, held March 20-22, 1972, at the IBM Thomas J. Watson Research Center, Yorktown Heights, New York, USA*, pages 85–103, 1972.

[Kil92] Joe Kilian. A note on efficient zero-knowledge proofs and arguments (extended abstract). In *Proceedings of the 24th Annual ACM Symposium on Theory of Computing*, pages 723–732. ACM, 1992.

[KLW15] Venkata Koppula, Allison Bishop Lewko, and Brent Waters. Indistinguishability obfuscation for turing machines with unbounded memory. In *STOC*, pages 419–428. ACM, 2015.

[KP16] Yael Tauman Kalai and Omer Paneth. Delegating RAM computations. In *Theory of Cryptography - 14th International Conference, TCC 2016-B, Beijing, China, October 31 - November 3, 2016, Proceedings, Part II*, pages 91–118, 2016.

[KR09] Yael Tauman Kalai and Ran Raz. Probabilistically checkable arguments. In *CRYPTO*, 2009.

[KRR13] Yael Tauman Kalai, Ran Raz, and Ron D. Rothblum. Delegation for bounded space. In Boneh et al. [BRF13], pages 565–574.

[KRR14] Yael Tauman Kalai, Ran Raz, and Ron D. Rothblum. How to delegate computations: the power of no-signaling proofs. In *STOC*, pages 485–494. ACM, 2014.

[KRR16] Yael Tauman Kalai, Ran Raz, and Oded Regev. On the space complexity of linear programming with preprocessing. In *Proceedings of the 2016 ACM Conference on Innovations in Theoretical Computer Science, Cambridge, MA, USA, January 14-16, 2016*, pages 293–300, 2016.

[LFKN92] Carsten Lund, Lance Fortnow, Howard J. Karloff, and Noam Nisan. Algebraic methods for interactive proof systems. *J. ACM*, 39(4):859–868, 1992.

[Lip12] Helger Lipmaa. Progression-free sets and sublinear pairing-based non-interactive zero-knowledge arguments. In *TCC*, pages 169–189, 2012.

[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. IEEE Computer Society, 1994. Full version in [Mic00].

[Mic00] Silvio Micali. Computationally sound proofs. *SIAM J. Comput.*, 30(4):1253–1298, 2000.

[Nao03] Moni Naor. On cryptographic assumptions and challenges. In *Proceedings of the 23rd Annual International Cryptology Conference*, pages 96–109, 2003.

[PR17] Omer Paneth and Guy N. Rothblum. On zero-testable homomorphic encryption and publicly verifiable non-interactive arguments. In *Theory of Cryptography - 15th International Conference, TCC 2017, Baltimore, MD, USA, November 12-15, 2017, Proceedings, Part II*, pages 283–315, 2017.

[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.

[RRR16] Omer Reingold, Guy N. Rothblum, and Ron D. Rothblum. Constant-round interactive proofs for delegating computation. In *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, Cambridge, MA, USA, June 18-21, 2016*, pages 49–62, 2016.

[Sha92] Adi Shamir. IP = PSPACE. *Journal of the ACM*, 39(4):869–877, 1992.

[Val76] Leslie G. Valiant. Universal circuits (preliminary report). In *Proceedings of the Eighth Annual ACM Symposium on Theory of Computing*, STOC '76, pages 196–203, New York, NY, USA, 1976. ACM.