Electronic Colloquium on Computational Complexity, Report No. 122 (2011)

Competing Provers Protocols for Circuit Evaluation

Gillat Kol[†] Ran Raz[§]

Abstract

Let C be a (fan-in 2) Boolean circuit of size s and depth d, and let x be an input for C. Assume that a verifier that knows C but doesn't know x can access the low degree extension of x at one random point. Two competing provers try to convince the verifier that $C(x) = 0$ and $C(x) = 1$, respectively, and assume that one of the provers is honest.

For any $r \geq 1$, we give an r rounds protocol with communication complexity $d^{\frac{1}{r}} polylog\,(s)$ that convinces the verifier in the correct value of $C(x)$ (with small probability of error). In particular, when we allow only one round, the protocol exchanges $d \cdot polylog\,(s)$ bits, and when we allow $r = O\left(\frac{log(d)}{loglog(s)}\right)$ rounds, the protocol exchanges only $polylog\,(s)$ bits.

Moreover, the complexity of the verifier and honest provers in this protocol is $poly(s)$, and if in addition the circuit is $log(s)$-space uniform, the complexity of the verifier is $d^{\frac{1}{r}} polylog\,(s)$.[1]

The protocol is obtained by combining the *delegation protocol* of Goldwasser, Kalai and Rothblum [5] and the *competing provers protocols* of Feige and Kilian [3] and some new techniques. We suggest two applications of these results:

Delegating computation to competing clouds: The main motivation behind the protocol of [5] was *delegating computation* to a cloud. Using our new protocol, a verifier can delegate computation to two (or more) competing clouds. If at least one of the clouds is reliable the verifier can trust that the computation is correct (with high probability). The advantage over the protocol of [5] is that the communication complexity and the number of rounds in our protocol are significantly lower.

[†] Department of Computer Science and Applied Mathematics, Weizmann Institute of Science, Rehovot, Israel. Research supported by Israel Science Foundation (ISF) grant. Email: gillat.kol@weizmann.ac.il

[§] Department of Computer Science and Applied Mathematics, Weizmann Institute of Science, Rehovot, Israel. Research supported by Israel Science Foundation (ISF) grant. Email: ran.raz@weizmann.ac.il.

[1] We have learnt that a result similar to ours for the case $r = 1$, as well as the motivation of delegating computation to competing clouds, was independently obtained by Canetti, Riva and Rothblum[2].

ISSN 1433-8092

Communication complexity with competing provers, and circuit lower bounds: Aaronson and Wigderson [1] suggested the model of communication complexity with competing provers, where two competing provers try to convince two players that $f(x,y)=0$ and $f(x,y)=1$, respectively, where x is an input held by the first player and y is an input held by the second player. By scaling down the competing provers protocols of [3], they showed that strong enough lower bounds for the communication complexity of f, in this model, imply lower bounds for the computational complexity of f.

Our results strengthen this connection. More precisely, we show that if f can be computed by a Boolean circuit of size s and depth d then for any $r \geq 1$ there is an r rounds protocol for f, in this model, with communication complexity $d^{\frac{1}{r}} \mathit{polylog}\,(s)$. This can be viewed as a possible direction towards proving circuit lower bounds. For instance, in order to prove $f \notin \mathcal{NC}$, it suffices to show that any one round protocol for f, in this model, requires the exchange of $\omega\,(\mathit{polylog}\,(n))$ bits. This gives a relatively simple combinatorial property that implies strong circuit lower bounds.

1 Introduction

The model of *refereed games,* or, *interactive proofs with competing provers,* was first introduced by Feige, Shamir and Tennenholtz [4]. Informally, given a language L and an input x, an efficient verifier interacts (that is, exchanges private messages) with two computationally-unbounded provers in order to decide whether or not $x \in L$. One prover tries to convince the verifier that $x \in L$, while the other prover tries to convince the verifier that $x \notin L$. (The assumption is that the prover who is right is honest and follows the protocol, while the other is untrusted). The seminal paper of Feige and Kilian [3] shows that the class of languages with a *single round* refereed games is exactly PSPACE, and the class of languages with $\mathit{poly}(n)$ rounds refereed games is exactly EXP.

Let C be a (fan-in 2) Boolean circuit of size s and depth d, and let x be an input for C. Assume that a verifier that knows C but doesn't know x can access the low degree extension of x at one random point. The verifier tries to decide whether or not x satisfies C. Two competing provers try to convince the verifier that $C(x)=0$ and $C(x)=1$, respectively, and, as before, assume that one of the provers is honest.

Problems of this type in the model of *interactive proofs with a single prover* were studied in [6, 5]. In particular, the beautiful paper of Goldwasser, Kalai and Rothblum [5] gives a protocol where the prover convinces the verifier in the correct value of $C(x)$ (with small probability of error), and where the number of rounds and the communication complexity are both $d \cdot \mathit{polylog}\,(s)$, and the complexity of the prover is $\mathit{poly}(s)$. Moreover, maybe the most interesting aspect of their protocol is that if in addition one assumes that the circuit C is $log(s)$-space uniform, the complexity of the verifier in the protocol is $d \cdot polylog\,(s)$. That is, the complexity of the verifier may be significantly smaller than the size of the circuit. In other words, the computation was delegated to the prover, while the verifier is still convinced that the computation is correct.

1.1 Our Results

As before, let C be a (fan-in 2) Boolean circuit of size s and depth d, and let x be an input for C. Assume that a verifier that knows C but doesn't know x can access the low degree extension of x at one random point. Two competing provers try to convince the verifier that $C(x) = 0$ and $C(x) = 1$, respectively.

For any $r \geq 1$, we give an r rounds protocol with communication complexity $d^{\frac{1}{r}} polylog\,(s)$ that convinces the verifier in the correct value of $C(x)$ (with small probability of error). In particular, when we allow only one round, the protocol exchanges $d \cdot polylog\,(s)$ bits, and when we allow $r = O\left(\frac{log(d)}{loglog(s)}\right)$ rounds, the protocol exchanges only $polylog\,(s)$ bits.

Moreover, the complexity of the verifier and (honest) provers in this protocol is $poly(s)$, and if in addition the circuit is $log(s)$-space uniform, the complexity of the verifier is $d^{\frac{1}{r}} polylog\,(s)$.

The protocol is obtained by combining the *delegation protocol* of Goldwasser, Kalai and Rothblum [5] and the *competing provers protocols* of Feige and Kilian [3], together with some new techniques.

We have learnt that a result similar to ours for the case $r = 1$, as well as the motivation of delegating computation to competing clouds, was independently obtained by Canetti, Riva and Rothblum[2].

1.2 Delegating Computation to Competing Clouds

As mentioned above, the main motivation behind the protocol of Goldwasser, Kalai and Rothblum [5] was delegating computation to a cloud: Consider the case of log-space uniform circuits. As mentioned above, the complexity of the verifier in this case is relatively small, assuming that she has access to the low degree extension of x at a random point. Moreover, even if the verifier doesn't have access to the low degree extension of x (but rather knows x), she can compute the low degree extension of x at a random point by herself in almost linear time, which can be significantly smaller than the size of the circuit. Thus, the computation was delegated to the prover, while the verifier is convinced (with high probability) that the computation was performed correctly.

Using our new protocol, a verifier can delegate computation to two (or more) competing clouds, viewing each cloud as a separate prover. If at least one of the clouds is reliable, the verifier will be able to identify the correct answer and trust that the computation is correct (with high probability). The advantage over the protocol of [5] is that the communication complexity and the number of rounds in our protocol are significantly lower.

1.3 Communication Complexity with Competing Provers and Circuit Lower Bounds

Aaronson and Wigderson [1] suggested the model of communication complexity with two competing provers.

Recall that the standard setting of communication complexity involves two separated, computationally-unbounded players, Alice and Bob, each holding a private n-bit input string. Assume that Alice holds $x \in \{0,1\}^n$ and Bob holds $y \in \{0,1\}^n$. Let $f : \{0,1\}^{2n} \to \{0,1\}$ be a Boolean function known to both Alice and Bob, and let $x \circ y \in \{0,1\}^{2n}$ denote the concatenation of x and y. The players' mutual goal is to compute $f(x \circ y)$ with the least amount of communication between them.

In the model of communication complexity with competing provers the players may communicate, via private channels, with two provers *that are assumed to know f, as well as both x and y.* The two provers try to convince the players that $f(x \circ y) = 0$ and $f(x \circ y) = 1$, respectively. The *communication complexity* of a protocol is the total number of bits exchanged between all the parties (players and provers). In this paper we are interested in both the number of rounds and the communication complexity of the suggested protocols.

By scaling down the competing provers protocols of [3], Aaronson and Wigderson showed that strong enough lower bounds for the communication complexity of f, in this model, imply lower bounds for the computational complexity of f. More precisely, they show that for any language $L \in \mathsf{P}$ there are protocols in this model with poly-logarithmic rounds and communication complexity, and for any language $L \in \mathsf{LOGSPACE}$ there are one-round protocols with poly-logarithmic communication complexity.

Our results strengthen this connection and relates the communication complexity of f in this model to the circuit complexity of f.

More precisely, our results imply that if f can be computed by a Boolean circuit of size s and depth d then for any $r \geq 1$ there is an r rounds protocol for f, in this model, with communication complexity $d^{\frac{1}{r}} polylog(s)$. In particular, as before, when we allow only one round, the protocol exchanges $d \cdot polylog(s)$ bits, and when we allow $r = O\left(\frac{log(d)}{loglog(s)}\right)$ rounds, the protocol exchanges only $polylog(s)$ bits.

This can be viewed as a possible direction towards proving circuit lower bounds. For instance, in order to prove $f \notin \mathcal{NC}$, it suffices to show that any one-round protocol for f, in this model, requires the exchange of $\omega\left(\mathit{polylog}\left(n\right)\right)$ bits. The one-round case is very appealing as it gives a relatively simple combinatorial condition that implies strong circuit lower bounds.

We also consider the model of communication complexity with one prover, where a single prover tries to convince the players that $f(x, y) = 1$. We observe that the result of [5] (combined with the approach of [1]) implies a connection between the communication complexity of f in this model and the circuit complexity of f. More precisely, their result implies that if f can be computed by a Boolean circuit of size s and depth d, then there are protocols for f, in this model, with $d \cdot \mathit{polylog}\left(s\right)$ rounds and the exchange of $d \cdot \mathit{polylog}\left(s\right)$ bits.

2 Our Settings

We study the query complexity and communication complexity settings in the presence of (untrusted) provers. In this section we present the formal settings used in the paper.

We start by reviewing the standard query and communication complexity settings (Subsection 2.1). We present a generalized framework for interaction between players and provers, called *general communication protocols* (Subsection 2.2). We describe the settings for query and communication complexity with provers (Subsections 2.3 and 2.4). Finally, we discuss the relations between query and communication complexity with provers (Subsection 2.5).

Let $\mathcal{F}_n$ be the set of Boolean functions on n bits, $\mathcal{F}_n = \{f \mid f : \{0,1\}^n \to \{0,1\}\}$.

2.1 Query and Communication Complexity

2.1.1 Query Complexity

In the query complexity setting, we are given a function $f \in \mathcal{F}_n$ and an input $x \in \{0,1\}^n$. We are then asked to compute $f\left(x\right)$ by reading the least number of bits from x.

2.1.2 Communication Complexity

The communication complexity setting involves two separated, computationally-unbounded players, Alice and Bob, that attempt to evaluate a function $f \in \mathcal{F}_{2n}$ known to both. Alice receives a private n-bit string x, and Bob receives a private n-bit string y. Let $x \circ y \in \{0,1\}^{2n}$ denote the concatenation of the strings x and y. The goal is for one of the players (for concreteness, say Alice) to compute and output $f\left(x \circ y\right)$, with the least amount of communication between them.

2.2 General Communication Protocols

We consider the setting in which a set of players and provers, each holding a private input (bit-string), communicate. A general communication protocol (GCP) prescribes a probabilistic strategy to each participant. During the execution of the protocol, any player may exchange messages (bit-strings) with any of the other players or provers through a private channel. We assume that there is no communication among provers.

At the end of the protocol the first player must output a value in $\{0, 1, \perp\}$. Informally speaking, this value should be the result of the evaluation of a Boolean function on the participants' inputs. A 0 or 1 output is interpreted as the actual value of the function, while $\perp$ indicates that one of the provers did not follow his prescribed strategy.

A GCP proceeds in *rounds*. Each round consists of three sets of messages sent in the following order (messages may be empty):

1. Players send questions: Each player sends a message (question) to each of the provers.
2. Provers reply: Each prover sends a message (answer) to each of the players.
3. Players interact: Each player sends a message to each of the other players, according to some pre-specified order.

The *communication complexity* of a GCP is defined as the maximal number of bits exchanged by all the involved parties (provers and players) in any execution. An (r, c)-GCP is a GCP with at most r rounds, and communication complexity at most c.

2.3 Query and Communication Complexity with Provers

Next, we formally define the settings used in the paper. We define two variants of the query complexity model, called QCP and QCCP, as well as the two variants of the communication complexity model, called CCP and CCCP.

In the CCP and CCCP models, the players attempt to compute the value of a function $f \in \mathcal{F}_{2n}$ on an input $x \circ y \in \{0, 1\}^{2n}$. We assume that the prover(s) know both x and y, while Alice only has x, and Bob only has y.

In the QCP and QCCP models, the player attempts to compute the value of a function $f \in \mathcal{F}_n$ on an input $x \in \{0, 1\}^n$. We assume that the prover(s) have the input x, *while the player has an oracle access to a (possibly a more "elaborated") version of x.* That is, the player has access to a string $y = g(x)$, where g is a function that possibly "stretches" x. We mention that in this paper $g(x)$ will always be the low-degree extension of x (see Subsection 2.4).

In the QCP model, we require the player to output one of the values $f(x)$ or $\perp$ with probability at least $1-\epsilon$, regardless of the behavior of the prover (that is, the player is only allowed to make a mistake with probability at most ϵ). If the prover is truthful (i.e., follows the prescribed strategy), we require the player to always output $f(x)$. The requirements for the CCP model are similar.

In the QCCP model, we assume that at least one of the provers is truthful (i.e., follows the prescribed strategy), and require the player to output $f(x)$ with probability at least $1-\epsilon$, regardless of the behavior of the other prover. Note that, without loss of generality, we may assume that one prover tries to convince the player that $f(x) = 1$, while the other tries to convince the player that $f(x) = 0$. The requirements for the CCCP model are similar.

For the rest of the section let $\epsilon \in [0,1]$, $n, r, c, q, \ell \in \mathbb{N}$, $\mathbb{F}$ be a field, and $g : \{0,1\}^n \to \mathbb{F}^\ell$.

Definition 2.1 (QCP). $\mathsf{QCP}_{\epsilon,g}(r,c,q)$ is the set of functions $f \in \mathcal{F}_n$ that can be computed by an (r,c)-GCP between a player and a prover, in the following sense: The prover is given an input $x \in \{0,1\}^n$, while the player is given the input $y = g(x)$.

For any strategy of the prover, a player who follows the protocol reads at most q symbols from y and, with probability at least $1-\epsilon$, outputs either $f(x)$ or $\perp$. If the prover also follows the protocol, then the player always outputs $f(x)$.

Definition 2.2 (CCP). $\mathsf{CCP}_{\epsilon}(r,c)$ is the set of functions $f \in \mathcal{F}_{2n}$, that can be computed by an (r,c)-GCP between two players, Alice and Bob, and a prover, in the following sense: Alice is given an input $x \in \{0,1\}^n$ and Bob is given an input $y \in \{0,1\}^n$, while the prover is given the input $x \circ y$.

If both players follow the protocol, then for any strategy of the prover, with probability at least $1-\epsilon$, Alice outputs either $f(x \circ y)$ or $\perp$. If the prover also follows the protocol, then Alice always outputs $f(x \circ y)$.

Definition 2.3 (QCCP). $\mathsf{QCCP}_{\epsilon,g}(r,c,q)$ is the set of functions $f \in \mathcal{F}_n$, that can be computed by an (r,c)-GCP between a player and a pair of provers, in the following sense: Both provers are given (the same) input $x \in \{0,1\}^n$, while the player is given the input $y = g(x)$.

Assume that for any fixed f and x, one of the provers, whose identity maybe unknown to the player, always follows the protocol. That is, this prover runs the probabilistic strategy prescribed to him by the protocol. Then, for any strategy of the other prover, a player who follows the protocol reads at most q symbols from y and, with probability at least $1-\epsilon$, outputs $f(x)$.

Definition 2.4 (CCCP). $\mathsf{CCCP}_{\epsilon}(r,c)$ is the set of functions $f \in \mathcal{F}_{2n}$, that can be computed by an (r,c)-GCP between two players, Alice and Bob, and a pair of provers, in the following sense: Alice is given the input $x \in \{0,1\}^n$ and Bob is given the input $y \in \{0,1\}^n$, while both provers are given $x \circ y$.

Assume that for any fixed f, x and y, one of the provers, whose identity maybe unknown to the players, always follows the protocol. That is, this prover runs the probabilistic strategy prescribed to him by the protocol. For any strategy of the other prover, if both players follow the protocol, Alice outputs $f\left(x \circ y\right)$ with probability at least $1-\epsilon$.

2.4 Low Degree Extension (LDE)

Recall that in the QCP and QCCP models, the prover(s) have an input x, while the player has access to $y = g\left(x\right)$. As mentioned before, in this paper, $g\left(x\right)$ is always the low-degree extension of x. In this subsection we define the function $g\left(x\right) = \mathsf{LDE}\left(x\right)$, that maps x to its low-degree extension.

Let $\mathbb{H} = \{0,...,h-1\}$ be a set and $\mathbb{F} = \{0,...,p-1\}$ be a field satisfying $h \le p \in \mathbb{N}$. Recall that the Low-Degree Extension (LDE) of a function $x : \mathbb{H}^m \to \mathbb{F}$ is the unique m variate polynomial $\tilde{x} : \mathbb{F}^m \to \mathbb{F}$ that agrees with x on $\mathbb{H}^m$ (i.e., $\tilde{x}\mid_{\mathbb{H}^m} = x$), and has degree at most $h-1$ in each of its m variables.

We sometimes refer to x and $\tilde{x}$ as truth table strings instead of functions. More precisely, we identify x with the vector in $\mathbb{F}^{h^m}$ whose j^{th} coordinate is $x\left(h_j\right)$, where h_j is the j^{th} element in the lexicographic ordering of $\mathbb{H}^m$. Similarly, we identify $\tilde{x}$ with the vector in $\mathbb{F}^{p^m}$ whose j^{th} coordinate is $\tilde{x}\left(p_j\right)$, where p_j is the j^{th} element in the lexicographic ordering of $\mathbb{F}^m$.

Let $s \in \mathbb{N}$. We define a set $\mathbb{H}_s = \{0,...,h_s-1\}$ and a field $\mathbb{F}_s = \{0,...,p_s-1\}$ satisfying $h_s \le p_s \in \mathbb{N}$. Let $h_s = \lceil log\left(s\right)\rceil$. Let $m_s \in \mathbb{N}$ be the minimal such that $h_s^{m_s} \ge s$. Let $\mathbb{F}_s$ be the field of smallest size satisfying $p_s \ge 12 m_s h_s^3$. Note that $h_s, p_s \le polylog\left(s\right)$, $m_s = O\left(\frac{log(s)}{loglog(s)}\right)$ and $s \le h_s^{m_s}, p_s^{m_s} \le poly\left(s\right)$.

For $n < s \in \mathbb{N}$, we define the function $\mathsf{LDE}_s : \{0,1\}^n \to \mathbb{F}_s^{p_s^{m_s}}$ as follows: Given $x \in \{0,1\}^n$, we pad x with 1's to get an $h_s^{m_s}$-size string $x' \in \mathbb{F}_s^{h_s^{m_s}}$. That is, x' is defined by: $\forall i \in [n] : x'_i = x_i$, and $\forall i \in \{n+1,...,h_s^{m_s}\} : x'_i = 1$. We remark that we pad x with 1's and not with 0's to make the proof of Lemma 5.1 (below) simpler. Finally, we define $\mathsf{LDE}_s\left(x\right)$ to be the low-degree extension $\tilde{x}'$ of x'.

2.5 Connection Between Query and Communication Complexity with Provers

The following observation made by [1] shows that results for the query complexity model (with $g\left(x\right) = \mathsf{LDE}_s\left(x\right)$) imply analogous results for the communication complexity model.

More precisely, in the communication complexity setting, any point in the low degree extension of the concatenated input $x \circ y$, can be computed by the players by the exchange of a single field element.

Let S be a set, and let $c \in \mathbb{N}$. Consider the simple generalization of the standard communication complexity setting that allows Alice and Bob to exchange symbols from the set S (instead of just bits). We denote by $\mathsf{CC}_S(c)$ the class of all functions $f : \{0,1\}^{2n} \to S$ that can be computed by having Alice and Bob exchange at most c elements of S.

Proposition 2.5 ([1])**.** Fix $n, s \in \mathbb{N}$, $s > 2n$, and $z \in \mathbb{F}_s^{m_s}$. Let $f : \{0,1\}^{2n} \to \mathbb{F}_s$ be given by $f(x \circ y) = (\mathsf{LDE}_s(x \circ y))(z)$. Then, $f \in \mathsf{CC}_{\mathbb{F}_s}(1)$.

The proof of Proposition 2.5 follows easily by the fact that low degree extension is a linear function.

We say that a QCP or a QCCP protocol is *non-adaptive* if the queries to $y = g(x)$ made by the player only depend on the player's random string. We mention that protocols presented in this paper are non-adaptive.

The following is an easy corollary of the above proposition.

Corollary 2.6. Let $f \in \mathcal{F}_{2n}$, $r, c, q, s \in \mathbb{N}$, $2n < s$, and $\epsilon \in [0,1]$.

1. Assume $f \in \mathsf{QCP}_{\epsilon,\mathsf{LDE}_s}(r,c,q)$*, with a non-adaptive protocol.*
   Then $f \in \mathsf{CCP}_\epsilon(r, c + O(q \cdot log(s)))$.

2. Assume $f \in \mathsf{QCCP}_{\epsilon,\mathsf{LDE}_s}(r,c,q)$*, with a non-adaptive protocol.*
   Then $f \in \mathsf{CCCP}_\epsilon(r, c + O(q \cdot log(s)))$.

In the above CCP and CCCP protocols for f, the players exchange c bits with the prover(s), and $O(q \cdot log(s))$ bits among themselves.

3 Our Results

Let $\mathcal{F}_{s,d}$ be the set of all functions $f \in \mathcal{F}_{2n}$ that have an $s(n)$ size, $d(n)$ depth, Boolean circuit of fan-in 2. The main result of this paper is that functions in $\mathcal{F}_{s,d}$ can be computed with low query and communication cost in the QCCP model, and therefore (due to Corollary 2.6) with low communication cost in the CCCP model.

The first theorem claims that in the QCP and CCP models, $d \cdot polylog(s)$ communication bits suffice in order to compute any $f \in \mathcal{F}_{s,d}$ with high probability. The QCP part of the theorem is a restatement of the main theorem of [5], while the CCP part of it follows as in Corollary 2.6.

Remark. In all the protocols below, the players and honest provers run in time at most $poly\left(s\right)$.

Theorem 3.1 ([5])**.** *Let $f \in \mathcal{F}_{s,d}$ and $\epsilon = \frac{1}{log(s)}$. Then, there exists a field $\mathbb{F}$ of size $poly\left(d, log\left(s\right)\right)$ and a function*[2] $g : \{0,1\}^n \rightarrow \mathbb{F}^{poly(s)}$*, such that,*

$$f \in \mathsf{QCP}_{\epsilon,g}\left(d \cdot polylog\left(s\right), d \cdot polylog\left(s\right), 1\right) \text{ and } f \in \mathsf{CCP}_{\epsilon}\left(d \cdot polylog\left(s\right), d \cdot polylog\left(s\right)\right).$$

Our next theorem shows that the computation of $f \in \mathcal{F}_{s,d}$ can be done with even lower communication complexity in the competing provers models. The theorem offers a trade-off between the number of rounds and the communication complexity of the protocol. Namely, for every $r \in \mathbb{N}$, there is a protocol that computes f with high probability, and only requires r rounds and $d^{\frac{1}{r}} polylog\left(s\right)$ communication bits.

Theorem 3.2 (**Main**)**.** *Let $f \in \mathcal{F}_{s,d}$ and $\epsilon = \frac{1}{log(s)}$. $\forall r \in \mathbb{N}$* it holds that

$$f \in \mathsf{QCCP}_{\epsilon,\mathsf{LDE}_s}\left(r, d^{\frac{1}{r}} polylog\left(s\right), 1\right) \text{ and } f \in \mathsf{CCCP}_{\epsilon}\left(r, d^{\frac{1}{r}} polylog\left(s\right)\right).$$

The next two theorems are easy corollaries of Theorem 3.2 that are obtained by taking r to extreme values. Theorem 3.3 takes r to be 1, and claims that f can be computed by the exchange of $d \cdot polylog\left(s\right)$ bits and *with only a single communication round.* Theorem 3.4 shows that by choosing $r = O\left(\frac{log(d)}{loglog(s)}\right)$ we can reduce the communication complexity of the protocol to $polylog\left(s\right)$ at the price of having $O\left(\frac{log(d)}{loglog(s)}\right)$ rounds.

Theorem 3.3. *Let $f \in \mathcal{F}_{s,d}$ and $\epsilon = \frac{1}{log(s)}$.* Then,

$$f \in \mathsf{QCCP}_{\epsilon,\mathsf{LDE}_s}\left(\mathbf{1}, d \cdot polylog\left(s\right), 1\right) \text{ and } f \in \mathsf{CCCP}_{\epsilon}\left(\mathbf{1}, d \cdot polylog\left(s\right)\right).$$

Theorem 3.4. *Let $f \in \mathcal{F}_{s,d}$ and $\epsilon = \frac{1}{log(s)}$.* Then*,*

$$f \in \mathsf{QCCP}_{\epsilon,\mathsf{LDE}_s}\left(O\left({}^{log(d)}/_{loglog(s)}\right), polylog\left(s\right), 1\right),$$

$$f \in \mathsf{CCCP}_{\epsilon}\left(O\left({}^{log(d)}/_{loglog(s)}\right), polylog\left(s\right)\right).$$

The next theorem claims that if f is given by a $log\left(s\right)$-space uniform circuit of size s and depth d, then the player(s) in Theorem 3.2 can be very efficient and run in time $d^{\frac{1}{r}} polylog\left(s\right)$.

Theorem 3.5. *Let $f \in \mathcal{F}_{s,d}$ with a $log\left(s\right)$-**space uniform circuit**, and let $\epsilon = \frac{1}{log(s)}$. $\forall r \in \mathbb{N}$* the claim of Theorem 3.2 holds with players that run in time $d^{\frac{1}{r}} polylog\left(s\right)$ (and honest provers that run in time $poly\left(s\right)$), and error 2ϵ.

[2] The function g is a low degree extension with different parameters than the ones we have here.

4 Preliminaries

Fix a size s depth d circuit C, and an input x for C. Set $\epsilon = \frac{1}{log(s)}$. From now on, we omit the sub-s and write $\mathbb{H}$, $\mathbb{F}$, h, p, m instead of $\mathbb{H}_s$, $\mathbb{F}_s$, h_s, p_s, m_s. We enumerate the layers of C such that layer 0 is the output layer and layer d is the input layer. Note that since s is the number of wires in C, the number of gates in each of the layers cannot exceed s.

We add gates to each of the layers of C, so that each will contain exactly $h^m > s$ gates. We label the gates in every layer by elements of $\mathbb{H}^m$. The gates added to every layer i ($i \in \{0, ..., d\}$), are all "special" gates that always return 1. The fact that every layer contains such a gate makes the proof of Lemma 5.1 (below) simpler.

We associate with every layers $i \in \{0, ..., d\}$ the vector $V_i \in \mathbb{F}^{h^m}$ that consists of the values of the layer's gates when C is evaluated on x. That is, for $j \in [h^m]$, the j^{th} coordinate of V_i is the value, when C is evaluated on x, of the gate in layer i whose label is the j^{th} element in the lexicographic ordering of $\mathbb{H}^m$. We denote by $\tilde{V_i} \in \mathbb{F}^{p^m}$ the low-degree extension of V_i.

We may assume, without loss of generality, that the output layer, layer 0, originally consists of a single gate whose label is now 0^m. The player's task can now be rephrased as computing $\tilde{V_0}(0^m)$.

Let $I = \{0, ..., d\}$ and $L = \{0, ..., 2m\}$. Given a set A and an element $a \in A$, we denote $A_{-a} = A \setminus \{a\}$. Let $\phi \in \mathbb{F}^0$ be the unique zero-coordinates vector. Whenever we say that a curve or a polynomial is of degree k, we mean that it is of degree at most k. A line is a degree 1 curve.

5 Reducing Computation of Layer i to Layer $i + 1$

In this section we follow the lines of [5] and show that the problem of computing $\tilde{V_i}$ on a point in $\mathbb{F}^m$ can be reduced to the task of computing $\tilde{V}_{\mathbf{i+1}}$ on two points in $\mathbb{F}^m$, using a sum-check protocol.

5.1 $\tilde{V_i}$ is a Quadratic Polynomial in the Values of $V_{\mathbf{i+1}}$

Let $i \in I_{-d}$. The next lemma claims that $\tilde{V_i}$ is a degree-2 polynomial in the values of V_{i+1}. The reason is that since $\tilde{V_i}$ is the low-degree extension of V_i, for every $z \in \mathbb{F}^m$, the value $\tilde{V_i}(z)$ is a linear combination of the values of V_i on $\mathbb{H}^m$. That is, it is a linear combination of the values of the gates of layer i. In addition, the value of a gate in layer i is a degree-2 polynomial in the values of its two children in layer $i + 1$.

Lemma 5.1. *For every $i \in I_{-d}$, there exists a function $\beta_i : \mathbb{F}^m \times (\mathbb{H}^m)^2 \to \mathbb{F}$ that does not depend on the input x to the circuit, such that:*

1. $\forall z \in \mathbb{F}^m : \tilde{V}_i(z) = \sum_{w_0, w_1 \in \mathbb{H}^m} \beta_i(z, w_0, w_1) V_{i+1}(w_0) V_{i+1}(w_1)$.
2. *For every fixed $w_0, w_1 \in \mathbb{H}^m$, and every $j \in [m]$, the function $\beta_i(z_1 ... z_m, w_0, w_1)$ is a polynomial of degree at most $h-1$ in z_j.*

Proof Fix $i \in I_{-d}$ and let $z \in \mathbb{F}^m$. The value $\tilde{V}_i(z)$ can be written as

$$\tilde{V}_i(z) = \sum_{p \in \mathbb{H}^m} \beta'(z, p) \cdot V_i(p),$$

where $\beta'(z, p) : \mathbb{F}^m \times \mathbb{H}^m \to \mathbb{F}$ is the following $2m$-variables function

$$\beta'(z, p) = \prod_{j=1}^{m} \prod_{q_j \in \mathbb{H} \setminus \{p_j\}} (z_j - q_j) \cdot (p_j - q_j)^{-1}.$$

Note that for every $j \in [m]$ and every fixed $p_0 \in \mathbb{H}^m$, the function $\beta'(z_1 ... z_m, p_0)$ is a degree $h-1$ polynomial in z_j.

The value of gate $p \in \mathbb{H}^m$ in layer i depends on the values of its two child gates in layer $i+1$. Recall that every layer of C contains at least one "special" gate, that returns 1 on every input. Conclude that there exists a function $c : (\mathbb{H}^m)^3 \to \mathbb{F}$ (c depends on the circuit C) such that

$$V_i(p) = \sum_{w_0, w_1 \in \mathbb{H}^m} c(p, w_0, w_1) V_{i+1}(w_0) V_{i+1}(w_1).$$

For example, if gate p in layer i is an OR gate whose children are gates $w_0, w_1 \in \mathbb{H}^m$ in layer $i+1$, and gate $w^* \in \mathbb{H}^m$ in layer $i+1$ is a gate that always returns 1, then

$$V_i(p) = V_{i+1}(w_0) V_{i+1}(w^*) + V_{i+1}(w_1) V_{i+1}(w^*) - V_{i+1}(w_0) V_{i+1}(w_1).$$

Putting it all together,

$$\begin{aligned} \tilde{V}_i(z) &= \sum_{p \in \mathbb{H}^m} \beta'(z, p) \cdot \left(\sum_{w_0, w_1 \in \mathbb{H}^m} c(p, w_0, w_1) V_{i+1}(w_0) V_{i+1}(w_1) \right) \\ &= \sum_{w_0, w_1 \in \mathbb{H}^m} \left(\sum_{p \in \mathbb{H}^m} \beta'(z, p) \cdot c(p, w_0, w_1) \right) V_{i+1}(w_0) V_{i+1}(w_1). \end{aligned}$$

Finally, we denote

$$\beta_i (z, w_0, w_1) = \sum_{p \in \mathbb{H}^m} \beta' (z, p) \cdot c (p, w_0, w_1),$$

and get

$$\tilde{V}_i (z) = \sum_{w_0, w_1 \in \mathbb{H}^m} \beta_i (z, w_0, w_1) V_{i+1} (w_0) V_{i+1} (w_1).$$

Since $\beta' (z_1 ... z_m, p)$ is of degree at most $h-1$ in z_j for every $j \in [m]$, it is also the case that β_i is of degree at most $h-1$ in z_j. ■

5.2 Computing $\tilde{V}_i$ Given $\tilde{V}_{i+1}$ by Sum-Check

Let $i \in I_{-d}$. The following lemma shows that for every $z \in \mathbb{F}^m$, verifying a claimed value for $\tilde{V}_i (z)$ can be done by a sum-check protocol that only requires the values of $\tilde{V}_{\mathbf{i+1}}$ on two points in $\mathbb{F}^m$.

Lemma 5.2. *For every $i \in I_{-d}$, there exists a function $\tilde{\beta}_i : (\mathbb{F}^m)^3 \to \mathbb{F}$ that does not depend on the input x to the circuit, such that the function $\Phi_i : (\mathbb{F}^m)^3 \to \mathbb{F}$ given by*

$$\Phi_i (z, w_0, w_1) = \tilde{\beta}_i (z, w_0, w_1) \tilde{V}_{i+1} (w_0) \tilde{V}_{i+1} (w_1)$$

satisfies:

1. $\forall z \in \mathbb{F}^m : \tilde{V}_i (z) = \sum_{w_0, w_1 \in \mathbb{H}^m} \Phi_i (z, w_0, w_1)$.

2. *Φ_i is of degree at most $2(h-1)$ in each of its $3m$ variables.*

Proof Fix $i \in I_{-d}$, let $z, w_0, w_1 \in \mathbb{F}^m$, and let $\beta_i : \mathbb{F}^m \times (\mathbb{H}^m)^2 \to \mathbb{F}$ be the function promised by Lemma 5.1. We denote by $\tilde{\beta}_i : (\mathbb{F}^m)^3 \to \mathbb{F}$ the low-degree extension of $\beta_i (z, w_0, w_1)$ with respect to w_0 and w_1. That is, fix $z \in \mathbb{F}^m$, and let $\beta_{i,z} : \mathbb{H}^{2m} \to \mathbb{F}$ be given by $\beta_{i,z} (w_0, w_1) = \beta_i (z, w_0, w_1)$. Let $\tilde{\beta}_{i,z} : \mathbb{F}^{2m} \to \mathbb{F}$ be the low-degree extension of $\beta_{i,z}$. Finally, let $\tilde{\beta}_i (z, w_0, w_1) = \tilde{\beta}_{i,z} (w_0, w_1)$.

Due to Lemma 5.1, it holds that

$$\forall z \in \mathbb{F}^m : \tilde{V}_i (z) = \sum_{w_0, w_1 \in \mathbb{H}^m} \Phi_i (z, w_0, w_1).$$

Furthermore, $\Phi_i (z, w_0, w_1)$ is of low-degree in each of its $3m$ variables: Let $z = z_1 ... z_m$, $w_0 = w_{0,1} ... w_{0,m}$, $w_1 = w_{1,1} ... w_{1,m}$, and fix $j \in [m]$. Since $\tilde{\beta}_i$ is the low-degree extension of β_i (with respect to w_0 and w_1), it is of degree at most $h-1$ in $w_{0,j}$ and $w_{1,j}$. Since $\tilde{V}_{i+1}$ is a low-degree extension of V_{i+1}, it is of degree at most $h-1$ in each of its m variables. Conclude that Φ_i is of degree at most $2(h-1)$ in $w_{0,j}$ and $w_{1,j}$.

Fix $w_0, w_1 \in \mathbb{H}^m$. Using Lemma 5.1, the function $\beta_i(z_1...z_m, w_0, w_1)$ is a polynomial of degree $h-1$ in z_j. Again, using the fact that $\tilde{\beta}_i$ is the low-degree extension of β_i, it holds that $\tilde{\beta}_i(z, w_0, w_1)$ is a linear combination of the values $\beta_i(z, w'_0, w'_1)$ ($w'_0, w'_1 \in \mathbb{H}^m$) with coefficients that depend only on w_0, w_1, w'_0, w'_1 (and are independent of z).

For every $w'_0, w'_1 \in \mathbb{H}^m$, the value $\beta_i(z, w'_0, w'_1)$ is a polynomial of degree at most $h-1$ in z_j. Thus, any linear combination of these values, and in particular $\tilde{\beta}_i(z, w_0, w_1)$, is a polynomial of degree at most $h-1$ in z_j. Conclude that Φ_i is of degree at most $h-1$ in z_j. ■

For $i \in I_{-d}$, let Φ_i be the function defined in Lemma 5.2. For every $i \in I_{-d}, \ell \in L$, we define the partial-sum function $\Phi_{i,\ell}(z, x) : \mathbb{F}^m \times \mathbb{F}^\ell \to \mathbb{F}$ as follows:

$$\Phi_{i,\ell}(z, x_1...x_\ell) = \sum_{x_{\ell+1},...,x_{2m} \in \mathbb{H}} \Phi_i(z, x_1...x_{2m}).$$

Note that $\Phi_{i,\ell}(z_1...z_m, x_1...x_\ell)$ is of degree at most $2(h-1)$ in each of its $m+\ell$ variables, and that

$$\tilde{V}_i(z) = \Phi_{i,0}(z, \phi) \tag{1}$$

$$\Phi_i(z, w_0, w_1) = \Phi_{i,2m}(z, w_0 \circ w_1). \tag{2}$$

Also observe that a sum-check protocol can be used to reduce the computation of $\tilde{V}_i(z) = \Phi_{i,0}(z, \phi)$ to the computation of $\Phi_i(z, w_0, w_1) = \Phi_{i,2m}(z, w_0 \circ w_1)$. Since $\tilde{\beta}_i$ does not depend on the input to the circuit x, the player can compute $\Phi_{i,2m}(z, w_0 \circ w_1)$ by herself given $\tilde{V}_{i+1}(w_0)$ and $\tilde{V}_{i+1}(w_1)$. In other words, the evaluation of $\tilde{V}_i$ on z can be reduced, by a sum-check protocol, to the computation of $\tilde{V}_{i+1}$ on the pair of points $w_0, w_1 \in \mathbb{F}^m$.

6 One Round Protocol (Proof of Theorem 3.3)

In this section we prove Theorem 3.3. Namely, we show that for any $f \in \mathcal{F}_{s,d}$ it holds that $f \in \mathsf{QCCP}_{\epsilon,\mathsf{LDE}_s}(1, d \cdot polylog(s), 1)$ where $\epsilon = \frac{1}{log(s)}$. We describe a protocol that given a size s depth d circuit C on n bits, and an input $x \in \{0,1\}^n$, outputs $C(x)$ with probability at least $1-\epsilon$. The protocol consists of a *single round*, requires the exchange of $d \cdot polylog(s)$ bits, and makes a single query to $\tilde{x} = \mathsf{LDE}_s(x)$.

The protocol consists of three steps: (*i*) The player sends questions to the provers (Subsection 6.1). (*ii*) The provers reply (Subsection 6.2). (*iii*) Using the provers' answers, the player decides on the value of $C(x)$ (Subsection 6.4). Subsection 6.3 contains observations that are used in Subsection 6.4 in order to show the correctness of the player's decision.

We assume that one of the provers claims that $C(x) = 1$, while the other claims $C(x) = 0$. Otherwise, if both claim that $C(x) = b$ ($b \in \{0, 1\}$), then since at least one of the provers is truthful, it must be the case that indeed $C(x) = b$. In the protocol below we assume that the player knows the identity of the prover claiming that $C(x) = 1$ (we call him P_1), and the identity of the prover claiming that $C(x) = 0$ (we call him P_0). This assumption is justified as the player may require each prover to add his claimed value for $C(x)$ to his reply message.

For the rest of the section let $\beta \in \mathbb{F} \backslash \mathbb{H}$ be an arbitrary element.

Remark (functions representation). During the execution of the protocol, the participants exchange a set of curves and polynomials. Each of the exchanged curves is of degree (at most) h. Each exchanged polynomial is either of the form $F(r) = \tilde{V}_i(Z(r))$ ($i \in I$) or of the form $F(r, t) = \Phi_{i,\ell}(Z(r), X(t))$ ($i \in I_{-d}, \ell \in L$), where $Z(r) : \mathbb{F} \to \mathbb{F}^m$ and $X(t) : \mathbb{F} \to \mathbb{F}^\ell$ are two previously exchanged degree-h curves. In other words, the exchanged polynomials are all restrictions of $\tilde{V}_i$ and $\Phi_{i,\ell}$ to degree h curves.

The polynomial $\tilde{V}_i(z)$ is of degree at most $h - 1$ in each of its m variables, as it is the low-degree extension of V_i. The polynomial $\Phi_{i,\ell}(z, x_1 ... x_\ell)$ is of degree at most $2(h - 1)$ in each of its $m + \ell$ variables (see Subsection 5.2). Therefore, each exchanged polynomial is of total degree at most $h \cdot 2(h - 1) \cdot (m + \ell) < 6mh^2$. We require each such polynomial to be sent in a *canonical representation*, as a sequence of at most $6mh^2$ coefficients. Curves will be sent as sequences of polynomials.

6.1 The Player's Questions

For every $i \in I_{-d}, \ell \in L$, the player constructs a degree-h curve $C_{i,\ell} : \mathbb{F} \to \mathbb{F}^\ell$, and a line $D_{i,\ell} : \mathbb{F} \to \mathbb{F}^\ell$. For every $i \in I$, the player constructs a degree-2 curve $Z_i : \mathbb{F} \to \mathbb{F}^m$, and a point $z_i \in \mathbb{F}^m$. The curves Z_i ($i \in I$) and $C_{i,\ell}$ ($i \in I_{-d}, \ell \in L$) are given to P_1, while the points z_i ($i \in I$) and the lines $D_{i,\ell}$ ($i \in I_{-d}, \ell \in L$) are given to P_0.

Let $Z_0 : \mathbb{F} \to \mathbb{F}^m$ be the constant 0^m curve. That is, $\forall r \in \mathbb{F} : Z_0(r) = 0^m$. Randomly select $\zeta_0 \in_R \mathbb{F} \backslash \{0, 1\}$. Set $z_0 = Z_0(\zeta_0) = 0^m$.

Recall that $\beta \in \mathbb{F} \backslash \mathbb{H}$ is an arbitrary element. For $i = 0, ..., d - 1$ we construct the additional needed curves and points in the following order:

The curves $(C_{i,\ell})_{\ell \in L}$ and $(D_{i,\ell})_{\ell \in L}$ are constructed recursively:

- **Construct $C_{i,0}, D_{i,0}$:** Define $C_{i,0}, D_{i,0} : \mathbb{F} \to \mathbb{F}^0$ by $\forall t \in \mathbb{F} : C_{i,0}(t) = D_{i,0}(t) = \phi$.

- **Construct $C_{i,\ell}$ given $D_{i,\ell-1}$ ($\ell \in L_{-0}$):**
  Randomly select $\delta_{i,\ell-1} \in_R \mathbb{F} \backslash \{0\}$ and set $a_{i,\ell-1} = D_{i,\ell-1}(\delta_{i,\ell-1})$.
  Let $C_{i,\ell}$ be a *random* degree-h curve satisfying $\forall \alpha \in \mathbb{H} : C_{i,\ell}(\alpha) = a_{i,\ell-1} \circ \alpha$, chosen as follows: Randomly select $c_{i,\ell} \in_R \mathbb{F}^\ell$, and let $C_{i,\ell}$ be the degree-h curve satisfying $\forall \alpha \in \mathbb{H} : C_{i,\ell}(\alpha) = a_{i,\ell-1} \circ \alpha$ and $C_{i,\ell}(\beta) = c_{i,\ell}$.

- **Construct $D_{i,\ell}$ given $C_{i,\ell}$ ($\ell \in L_{-0}$):**
  Randomly select $\gamma_{i,\ell} \in_R \mathbb{F} \backslash \mathbb{H}$.
  Let $D_{i,\ell}$ be a *random* line satisfying $D_{i,\ell}(0) = C_{i,\ell}(\gamma_{i,\ell})$, chosen as follows: Randomly select $d_{i,\ell} \in_R \mathbb{F}^\ell$, and let $D_{i,\ell}$ be the line satisfying $D_{i,\ell}(0) = C_{i,\ell}(\gamma_{i,\ell})$ and $D_{i,\ell}(1) = d_{i,\ell}$.

The curve Z_{i+1} is constructed as follows:
Randomly select $\delta_{i,2m} \in_R \mathbb{F} \backslash \{0\}$ and set $a_{i,2m} = D_{i,2m}(\delta_{i,2m})$.
Denote $a_{i,2m} = w_{i,0} \circ w_{i,1}$ where $w_{i,0}, w_{i,1} \in \mathbb{F}^m$.
Let Z_{i+1} be a *random* degree-2 curve satisfying $Z_{i+1}(0) = w_{i,0}$ and $Z_{i+1}(1) = w_{i,1}$, chosen as follows: Randomly select $w_{i+1} \in_R \mathbb{F}^m$, and let Z_{i+1} be the degree-2 curve satisfying $Z_{i+1}(0) = w_{i,0}$, $Z_{i+1}(1) = w_{i,1}$ and $Z_{i+1}(2) = w_{i+1}$.

The point z_{i+1} is selected as follows:
Randomly select $\zeta_{i+1} \in_R \mathbb{F} \backslash \{0, 1\}$. Set $z_{i+1} = Z_{i+1}(\zeta_{i+1})$.

6.2 The Provers' Answers

P_1's reply

For $i \in I_{-d}, \ell \in L_{-0}$, denote by $T^*_{i,\ell} : \mathbb{F} \times \mathbb{F} \to \mathbb{F}$ the polynomial $T^*_{i,\ell}(r,t) = \Phi_{i,\ell}(Z_i(r), C_{i,\ell}(t))$.
For $i \in I_{-0}$, denote by $U^*_i : \mathbb{F} \to \mathbb{F}$ the polynomial $U^*_i(r) = \tilde{V}_i(Z_i(r))$.

P_1 replies with a set of polynomials $T_{i,\ell} : \mathbb{F} \times \mathbb{F} \to \mathbb{F}$ ($i \in I_{-d}, \ell \in L_{-0}$), and a set of polynomials $U_i : \mathbb{F} \to \mathbb{F}$ ($i \in I_{-0}$). The honest P_1 should reply with $T_{i,\ell} = T^*_{i,\ell}$ and $U_i = U^*_i$.

P_0's reply

For $i \in I_{-d}, \ell \in L$, denote by $Q^*_{i,\ell} : \mathbb{F} \to \mathbb{F}$ the polynomial $Q^*_{i,\ell}(t) = \Phi_{i,\ell}(z_i, D_{i,\ell}(t))$.

P_0 replies with a set of polynomials $Q_{i,\ell} : \mathbb{F} \to \mathbb{F}$ ($i \in I_{-d}, \ell \in L$). The honest P_0 should reply with $Q_{i,\ell} = Q^*_{i,\ell}$.

6.3 Propositions

This subsection contains propositions that will be useful in the analysis of the player's decision in Subsection 6.4. Propositions 1 and 2 claim that the information given to P_1 by the player is independent of the pairs of values $(\zeta_i, \gamma_{i,\ell})$. Propositions 3 and 4 claim that the information given to P_0 by the player is independent of the values $\delta_{i,\ell}$.

Denote by I_1 and I_0 the random variables describing the information given to P_1 and P_0 (respectively): $I_1 = \left((Z_i)_{i\in I}, (C_{i,\ell})_{i\in I_{-d}, \ell\in L}\right)$, and $I_0 = \left((z_i)_{i\in I}, (D_{i,\ell})_{i\in I_{-d}, \ell\in L}\right)$. In the proofs below, whenever we say "object" we mean a field element, point or curve selected by the player while constructing the questions to the provers.

Proposition 1. *Let $i \in I_{-d}, \ell \in L_{-0}$. $\gamma_{i,\ell}$ is independent of I_1. That is, $\gamma_{i,\ell}$ is uniformly distributed in $\mathbb{F}\backslash\mathbb{H}$, even given I_1.*

Proof Let X be a random variable consisting of all the curves $Z_{i'}$ and $C_{i',\ell'}$ selected by the player *before* $\gamma_{i,\ell}$ is selected. Let Y be a random variable consisting of all the curves $Z_{i'}$ and $C_{i',\ell'}$ selected *after* $\gamma_{i,\ell}$. Clearly, $\gamma_{i,\ell}$ is selected independently of X. We will show that Y is independent of the pair $(X, \gamma_{i,\ell})$. This implies that I_1 is independent of $\gamma_{i,\ell}$.

The curve $D_{i,\ell}$ was selected to be a *random* line satisfying $D_{i,\ell}(0) = C_{i,\ell}(\gamma_{i,\ell})$. Therefore, for every fixed $\delta \in \mathbb{F}\backslash\{0\}$, the value $D_{i,\ell}(\delta)$ is uniformly distributed in $\mathbb{F}^\ell$, even given all the objects selected before $D_{i,\ell}$. In particular, $a_{i,\ell} = D_{i,\ell}(\delta_{i,\ell})$ is uniformly distributed in $\mathbb{F}^\ell$, even given all objects selected before $D_{i,\ell}$.

The player can select all the objects selected *after* $D_{i,\ell}$ by only knowing $a_{i,\ell}$. Since $a_{i,\ell}$ is uniformly distributed in $\mathbb{F}^\ell$, even given all objects selected before $D_{i,\ell}$, it holds that the objects selected after $D_{i,\ell}$ are independent of all objects selected before $D_{i,\ell}$. In particular, Y is independent of the pair $(X, \gamma_{i,\ell})$. ■

Proposition 2. *Let $i \in I_{-d}, \ell \in L_{-0}$. ζ_i is independent of the pair $(I_1, \gamma_{i,\ell})$. That is, ζ_i is uniformly distributed in $\mathbb{F}\backslash\{0,1\}$, even given $(I_1, \gamma_{i,\ell})$. Furthermore, ζ_d is independent of I_1.*

Proof Clearly, ζ_i is uniformly distributed, even given all the previously selected objects. In addition, all the objects selected after ζ_i (and after z_i was set) are independent of all the objects selected up-to ζ_i (including ζ_i). ■

Proposition 3. *Let $i \in I_{-d}, \ell \in L_{-2m}$. $\delta_{i,\ell}$ is independent of I_0. That is, $\delta_{i,\ell}$ is uniformly distributed in $\mathbb{F}\backslash\{0\}$, even given I_0.*

Proof Let X be a random variable consisting of all the points $z_{i'}$ and curves $D_{i',\ell'}$ selected by the player *before* $\delta_{i,\ell}$ is selected. Let Y be a random variable consisting of all the points $z_{i'}$

and curves $D_{i',\ell'}$ selected *after* $\delta_{i,\ell}$. Clearly, $\delta_{i,\ell}$ is selected independently of X. We will show that Y is independent of the pair $(X, \delta_{i,\ell})$. This implies that I_0 is independent of $\delta_{i,\ell}$.

The curve $C_{i,\ell+1}$ was selected to be a *random* degree-h curve satisfying $\forall \alpha \in \mathbb{H} : C_{i,\ell+1}(\alpha) = a_{i,\ell} \circ \alpha$. Therefore, for every fixed $\gamma \in \mathbb{F} \backslash \mathbb{H}$, the value $C_{i,\ell+1}(\gamma)$ is uniformly distributed in $\mathbb{F}^{\ell+1}$, even given all objects selected before $C_{i,\ell+1}$. In particular, $C_{i,\ell+1}(\gamma_{i,\ell+1})$ is uniformly distributed in $\mathbb{F}^{\ell+1}$, even given all objects selected before $C_{i,\ell+1}$.

The player can select all the objects selected after $C_{i,\ell+1}$ by only knowing $C_{i,\ell+1}(\gamma_{i,\ell+1})$. Since $C_{i,\ell+1}(\gamma_{i,\ell+1})$ is uniformly distributed in $\mathbb{F}^{\ell+1}$, even given all objects selected before $C_{i,\ell+1}$, it holds that the objects selected after $C_{i,\ell+1}$ are independent of all objects selected before $C_{i,\ell+1}$. In particular, Y is independent of the pair $(X, \delta_{i,\ell})$. ■

Proposition 4. *Let $i \in I_{-d}$. $\delta_{i,2m}$ is independent of I_0. That is, $\delta_{i,2m}$ is uniformly distributed in $\mathbb{F} \backslash \{0\}$, even given I_0.*

Proof Clearly, $\delta_{i,2m}$ is uniformly distributed, even given all the previously selected objects. In addition, all the objects selected after z_{i+1} are independent of all the objects selected up-to z_{i+1} (including z_{i+1}). Therefore, it suffices to show that z_{i+1} itself is independent of all the objects selected up-to $\delta_{i,2m}$ (including $\delta_{i,2m}$).

The curve Z_{i+1} was selected to be a *random* degree-2 curve satisfying $Z_{i+1}(0) = w_{i,0}$ and $Z_{i+1}(1) = w_{i,1}$. Therefore, for every fixed $\zeta \in \mathbb{F} \backslash \{0, 1\}$, the value $Z_{i+1}(\zeta)$ is uniformly distributed in $\mathbb{F}^m$, even given all the objects selected before Z_{i+1}. In particular, $z_{i+1} = Z_{i+1}(\zeta_{i+1})$ is uniformly distributed in $\mathbb{F}^m$, even given all the objects selected up-to $\delta_{i,2m}$ (including $\delta_{i,2m}$). ■

Let $i \in I_{-d}, \ell \in L_{-2m}$. Proposition 5 shows that if P_1 responds with the correct polynomial $T_{i,\boldsymbol{\ell+1}}$, then the player can compute by herself the correct value of $Q_{i,\ell}(\delta_{i,\ell})$.

Proposition 5. *Let $i \in I_{-d}, \ell \in L_{-2m}$. It hold that*

$$\sum_{\alpha \in \mathbb{H}} T^{*}_{i,\boldsymbol{\ell+1}}(\zeta_i, \alpha) = Q^{*}_{i,\ell}(\delta_{i,\ell}).$$

Proof

$$\begin{aligned}\sum_{\alpha\in\mathbb{H}} T^*_{i,\ell+1}\left(\zeta_i,\alpha\right) &= \sum_{\alpha\in\mathbb{H}} \Phi_{i,\ell+1}\left(Z_i\left(\zeta_i\right),C_{i,\ell+1}\left(\alpha\right)\right)\\ &= \sum_{\alpha\in\mathbb{H}} \Phi_{i,\ell+1}\left(z_i,a_{i,\ell}\circ\alpha\right)\\ &= \Phi_{i,\ell}\left(z_i,a_{i,\ell}\right)\\ &= \Phi_{i,\ell}\left(z_i,D_{i,\ell}\left(\delta_{i,\ell}\right)\right)\\ &= Q^*_{i,\ell}\left(\delta_{i,\ell}\right).\end{aligned}$$

■

Fix strategies for P_1 and P_0. Formally, a strategy for P_1 is a probabilistic algorithm that gets the sets of curves Z_i and $C_{i,\ell}$ as inputs, and outputs the sets of polynomials $T_{i,\ell}$ and U_i. A strategy for P_0 is a probabilistic algorithm that gets the set of points z_i and set of curves $D_{i,\ell}$ as inputs, and outputs the set of polynomials $Q_{i,\ell}$. For the rest of the section probabilities will be taken over P_1's random string R_1, P_0's random string R_0, and the player's random string.

We call each of the polynomials sent by the provers a "reply polynomial". The player's decision protocol checks the correctness of the reply polynomials in a pre-determined order. Let F be a reply polynomial. Denote by E_F the event that F is the first incorrect reply polynomial checked by the protocol. That is, E_F is the event that all the reply polynomials checked prior to F are correct, but F is incorrect.

Proposition 6. *Let F be a reply polynomial sent by P_1. Let X be a random variable that is independent of the pair (I_1, R_1). Then, X is independent of (I_1, R_1, I_{E_F}), where $I_{E_F}$is the random variable that gets the value 1 if E_F occurs, and 0 otherwise.*

The same holds when the parts of P_1 and P_0 are switched.

Proof Assume that E_F occurs with positive probability. P_1 is dishonest as he sent an incorrect polynomial F with positive probability. Since we assume that at least one prover is honest, it must be P_0. Therefore, all the reply polynomials sent by P_0 are correct. This implies that the event E_F depends solely on I_1 and R_1. That is, I_{E_F} is a function of I_1 and R_1. Since X is independent of (I_1, R_1), it is independent of (I_1, R_1, I_{E_F}). ■

6.4 The Player's Decision

First, the player queries $\tilde{x}\left(z_d\right)$ and denote the value by v_d. Note that $\tilde{x} = \mathsf{LDE}_s\left(x\right) = \tilde{V}_d$, and thus $v_d = \tilde{V}_d\left(z_d\right)$. Next, the player iterates over the layers of C, from the layer adjacent to the input layer (layer $i = d - 1$), to the output layer (layer $i = 0$). The iteration that handles layer i (simply referred to as "iteration i") assumes that the variable v_{i+1} is set to the value $\tilde{V}_{i+1}(z_{i+1})$. The goal of iteration i is to calculate the value $v_i = \tilde{V}_i(z_i)$. At the end of the loop the player returns v_0, which is, supposedly, $v_0 = \tilde{V}_0(z_0) = \tilde{V}_0(0^m) = C(x)$.

In order to compute v_i, iteration i proceeds in a series of "check steps" (tests), each designed to check the correctness of one of the reply polynomials for layer i. Let F be a reply polynomial, and let S_F be the check step designed to check F. If S_F succeeds, the protocol proceeds to the next check step. If S_F fails, the player outputs the value of $C(x)$ claimed by the prover that *did not* send F. The reason is that we assume that at least one of the provers is honest. Since F is incorrect, the prover that did not send F must be the honest one.

Recall that E_F denotes the event that F is the first incorrect reply polynomial checked by the protocol. Denote by C_F the event that all reply polynomials checked up-to F (including F) are correct. In order to prove Theorem 3.3 it suffices to show that for every reply polynomial F the following holds:

1. If C_F occurs, then S_F always succeeds.

2. If E_F occurs, then with probability at least $1 - \epsilon$, S_F fails.

3. If all the reply polynomials are correct, then the player returns $C(x)$.

Next, we describe the check steps that compose iteration i, and show that each of them satisfies both (1) and (2). Assume for simplicity that the reply polynomial F depends on only one variable. The check step S_F does the following: It computes $F(\sigma)$, where $\sigma \in \mathbb{F}$ is a field element that cannot be predicted by the prover that sent F. It then compares $F(\sigma)$ against the supposed value of $F^*(\sigma)$ (this latter value is computed by earlier check steps). If F is correct then $F = F^*$, and the test succeeds. If F is incorrect, F and F^* are different low-degree polynomials. Thus, it is likely that $F(\sigma) \neq F^*(\sigma)$, and the test is likely to fail.

We assume by induction (on the layers i) that if all the reply polynomials checked by iterations $d-1, ..., i+1$ are correct, then at the beginning of iteration i it holds that $v_{i+1} = \tilde{V}_{i+1}(z_{i+1})$.

Check U_{i+1}: ***Check*** $U_{i+1}(\zeta_{i+1}) = v_{i+1}$***. If not, return*** 0***.***

If $C_{U_{i+1}}$ ***occurs:*** By assumption, $v_{i+1} = \tilde{V}_{i+1}(z_{i+1}) = \tilde{V}_{i+1}(Z_{i+1}(\zeta_{i+1})) = U^*_{i+1}(\zeta_{i+1})$. Since U_{i+1} is correct, the test passes.

If $E_{U_{i+1}}$ ***occurs:*** Using Propositions 2 and 6, ζ_{i+1} is uniformly distributed in $\mathbb{F}\setminus\{0,1\}$, even given I_1, R_1 and $E_{U_{i+1}}$. Since U_{i+1} only depends on I_1 and R_1, it holds that when $E_{U_{i+1}}$ occurs, ζ_{i+1} is uniformly distributed, even given U_{i+1}.

As explained above, $v_{i+1} = U^*_{i+1}(\zeta_{i+1})$. Since U_{i+1} is incorrect, $U_{i+1} \neq U^*_{i+1}$. Since both U_{i+1} and U^*_{i+1} are of degrees less than $6mh^2$ (see the remark regarding functions representation at the beginning of the section), $U_{i+1}(\zeta)$ can only agree with $U^*_{i+1}(\zeta)$ on less than $6mh^2$ elements $\zeta \in \mathbb{F}$.

Recall that when $E_{U_{i+1}}$ occurs, ζ_{i+1} is uniformly distributed in $\mathbb{F}\setminus\{0,1\}$, even given U_{i+1}, and get that the probability that $U_{i+1}(\zeta_{i+1}) = U^*_{i+1}(\zeta_{i+1}) = v_{i+1}$ is less than $\frac{6mh^2}{|\mathbb{F}\setminus\{0,1\}|}$. Since $|\mathbb{F}| \geq 12mh^3$, it holds that $\frac{6mh^2}{|\mathbb{F}\setminus\{0,1\}|} < \frac{6mh^2}{\frac{1}{2}|\mathbb{F}|} \leq \frac{1}{h} \leq \frac{1}{log(s)} = \epsilon$.

Check $Q_{i,2m}$: *Calculate the supposed value $\phi_{i,2m}$ of $\Phi_{i,2m}(z_i, a_{i,2m}) = \Phi_i(z_i, w_{i,0} \circ w_{i,1})$ (see Equation 2) using $U_{i+1}(0)$ as $\tilde{V}_{i+1}(w_{i,0})$ and $U_{i+1}(1)$ as $\tilde{V}_{i+1}(w_{i,1})$.*
***Check** $Q_{i,2m}(\delta_{i,2m}) = \phi_{i,2m}$. **If not, return** 1.*

If** $C_{Q_{i,2m}}$ **occurs: Since U_{i+1} is correct $U_{i+1}(0) = U^*_{i+1}(0) = \tilde{V}_{i+1}(Z_{i+1}(0)) = \tilde{V}_{i+1}(w_{i,0})$, and similarly $U_{i+1}(1) = \tilde{V}_{i+1}(w_{i,1})$. Thus, $\phi_{i,2m}$ is set to $\Phi_{i,2m}(z_i, a_{i,2m}) = \Phi_{i,2m}(z_i, D_{i,2m}(\delta_{i,2m})) = Q^*_{i,2m}(\delta_{i,2m})$. Since $Q_{i,2m}$ is correct, the test passes.

If** $E_{Q_{i,2m}}$ **occurs: Using Propositions 4 and 6, $\delta_{i,2m}$ is uniformly distributed in $\mathbb{F}\setminus\{0\}$, even given I_0, R_0 and $E_{Q_{i,2m}}$. Since $Q_{i,2m}$ only depends on I_0 and R_0, it holds that when $E_{Q_{i,2m}}$ occurs, $\delta_{i,2m}$ is uniformly distributed, even given $Q_{i,2m}$.

As explained above, $\phi_{i,2m}$ is set to $Q^*_{i,2m}(\delta_{i,2m})$. Since $Q_{i,2m}$ is incorrect, $Q_{i,2m} \neq Q^*_{i,2m}$. An argument similar to the one used in check step $S_{U_{i+1}}$ shows that the probability that $Q_{i,2m}(\delta_{i,2m}) = Q^*_{i,2m}(\delta_{i,2m}) = \phi_{i,2m}$ is less than ϵ.

Check $T_{i,2m}$: ***Check** $T_{i,2m}(\zeta_i, \gamma_{i,2m}) = Q_{i,2m}(0)$. **If not, return** 0.*

If** $C_{T_{i,2m}}$ **occurs: $T^*_{i,2m}(\zeta_i, \gamma_{i,2m}) = \Phi_{i,2m}(Z_i(\zeta_i), C_{i,2m}(\gamma_{i,2m})) = \Phi_{i,2m}(z_i, D_{i,2m}(0)) = Q^*_{i,2m}(0)$. Since $Q_{i,2m}$ and $T_{i,2m}$ are correct, the test passes.

If** $E_{T_{i,2m}}$ **occurs: Using Propositions 1, 2 and 6, the pair $(\zeta_i, \gamma_{i,2m})$ is uniformly distributed in $(\mathbb{F}\setminus\{0,1\}) \times (\mathbb{F}\setminus\mathbb{H})$, even given I_1, R_1 and $E_{T_{i,2m}}$. Since $T_{i,2m}$ only depends on I_1 and R_1, it holds that when $E_{T_{i,2m}}$ occurs, the pair $(\zeta_i, \gamma_{i,2m})$ is uniformly distributed, even given $T_{i,2m}$.

Since $Q_{i,2m}$ is correct but $T_{i,2m}$ is incorrect, it holds that $Q_{i,2m} = Q^*_{i,2m}$ and $T_{i,2m} \neq T^*_{i,2m}$. Using the Schwartz–Zippel Lemma, since both $T_{i,2m}$ and $T^*_{i,2m}$ are of total degree less than $6mh^2$ (see the remark regarding functions representation at the beginning of the section), $T_{i,2m}(\zeta, \gamma)$ can only agree with $T^*_{i,2m}(\zeta, \gamma)$ on less than $\frac{6mh^2}{|\mathbb{F}|}$-fraction of the points $(\zeta, \gamma) \in \mathbb{F}^2$.

Recall that when $E_{T_{i,2m}}$ occurs, the pair $(\zeta_i, \gamma_{i,2m})$ is uniformly distributed in $(\mathbb{F}\setminus\{0,1\}) \times (\mathbb{F}\setminus\mathbb{H})$, even given $T_{i,2m}$, and get that the probability that $T_{i,2m}(\zeta_i, \gamma_{i,2m}) = T^*_{i,2m}(\zeta_i, \gamma_{i,2m}) = Q^*_{i,2m}(0) = Q_{i,2m}(0)$ is less than $\frac{6mh^2 \cdot |\mathbb{F}|}{|(\mathbb{F}\setminus\{0,1\}) \times (\mathbb{F}\setminus\mathbb{H})|}$. Since $|\mathbb{F}| \geq 12mh^3$, it holds that $\frac{6mh^2 \cdot |\mathbb{F}|}{|(\mathbb{F}\setminus\{0,1\}) \times (\mathbb{F}\setminus\mathbb{H})|} <$

$\frac{6mh^2 \cdot |\mathbb{F}|}{\frac{1}{2}|\mathbb{F}|^2} \leq \frac{1}{h} \leq \frac{1}{log(s)} = \epsilon$.

Check $Q_{i,2m-1}$**:** ***Check*** $Q_{i,2m-1}(\delta_{i,2m-1}) = \sum_{\alpha \in \mathbb{H}} T_{i,2m}(\zeta_i, \alpha)$***. If not, return*** 1**.**

If $C_{Q_{i,2m-1}}$ ***occurs:*** Using Proposition 5, since $T_{i,2m}$ and $Q_{i,2m-1}$ are correct, the test passes.

If $E_{Q_{i,2m-1}}$ ***occurs:*** Using Propositions 3 and 6, $\delta_{i,2m-1}$ is uniformly distributed in $\mathbb{F} \setminus \{0\}$, even given I_0, R_0 and $E_{Q_{i,2m-1}}$. Since $Q_{i,2m-1}$ only depends on I_0 and R_0, it holds that when $E_{Q_{i,2m-1}}$ occurs, $\delta_{i,2m-1}$ is uniformly distributed, even given $Q_{i,2m-1}$.

Using Proposition 5, since $T_{i,2m}$ is correct, $\sum_{\alpha \in \mathbb{H}} T_{i,2m}(\zeta_i, \alpha) = Q^*_{i,2m-1}(\delta_{i,2m-1})$. Since $Q_{i,2m-1}$ is incorrect, $Q_{i,2m-1} \neq Q^*_{i,2m-1}$. An argument similar to the one used in check step $S_{U_{i+1}}$ shows that the probability that $Q_{i,2m-1}(\delta_{i,2m-1}) = Q^*_{i,2m-1}(\delta_{i,2m-1}) = \sum_{\alpha \in \mathbb{H}} T_{i,2m}(\zeta_i, \alpha)$ is less than ϵ.

Repeat the last two steps to check $T_{i,2m-1}$, $Q_{i,2m-2}$,...,$T_{i,1}$,$Q_{i,0}$.

Set $v_i = Q_{i,0}(\beta)$.
Note that if $Q_{i,0}$ is correct then using Equation 1, $v_i = Q_{i,0}(\beta) = Q^*_{i,0}(\beta) = \Phi_{i,0}(z_i, D_{i,0}(\beta)) = \tilde{V}_i(z_i)$, and the induction hypothesis holds. In particular, if all reply polynomials are correct then $v_0 = \tilde{V}_0(z_0) = \tilde{V}_0(0^m) = C(x)$.

7 Trade-off Protocol (Proof of Theorem 3.2)

In this section we prove Theorem 3.2. We describe a protocol that given a size s depth d circuit C on n bits, and an input $x \in \{0,1\}^n$, outputs $C(x)$ with probability at least $1-\epsilon$. The new protocol uses the one-round protocol described in Section 6 as a building block. The new protocol consists of r rounds, requires the exchange of $d^{\frac{1}{r}} \cdot polylog(s)$ bits, and makes a single query to $\mathsf{LDE}_s(x)$.

Assume without loss of generality that $r \leq log(d)$ (otherwise, set $r = \lfloor log(d) \rfloor$). Set $k = 2\left\lceil d^{\frac{1}{r}} \right\rceil$. The new protocol consists of two stages. The goal of the first stage (Subsection 7.1) is to find two "close" layers $i' > i'' \in I$, $i' - i'' \leq k$ such that the provers agree on a point on $\tilde{V}_{i'}$, but disagree on a point on $\tilde{V}_{i''}$. Note that such layers exist, as the provers agree on the input x for C (and thus on $\mathsf{LDE}_s(x) = \tilde{V}_d$), but disagree on the output $C(x)$ (and therefore on $\tilde{V}_0$).

Finding layers i' and i'' is done by running a version of a binary search that in every round partitions the search range into k (almost equal) interval (instead of just two). Note that $r-1$ search rounds suffice in order to find such layers i' and i'' satisfying $i' - i'' \leq k$:

Each search round reduces the range of indices searched by a factor of at least $\frac{k}{2}$. Thus, after $r-1$ rounds we are left with a range of at most

$$(d+1)\cdot\left(\frac{k}{2}\right)^{-(r-1)} = (d+1)\left(\left\lceil d^{\frac{1}{r}}\right\rceil\right)^{-(r-1)} \leq 2d\left(d^{\frac{1}{r}}\right)^{-(r-1)} \leq 2d\cdot d^{\frac{1-r}{r}} = 2d^{\frac{1}{r}} \leq k.$$

Let C' a subcircuit composed of layers $i', \ldots, i''$ of C, and let $x' = V_{i'}$. The provers agree on a point on the input layer $\mathsf{LDE}_s(x') = \tilde{V}_{i'}$ of C', but disagree on a point on the output layer $\tilde{V}_{i''}$ of C'.

Stage 2 of the new protocol (Subsection 7.2) runs the one-round protocol described in Section 6 with C' and x' as inputs in order to determine which of the provers is truthful.

7.1 Stage 1: Search

First, the player constructs questions for the provers as in Subsection 6.1. That is, the player constructs the curves $C_{i,\ell}$, $D_{i,\ell}$ $(i \in I_{-d}, \ell \in L)$, and the curves Z_i and points z_i $(i \in I)$. For $i \in I$, let $U_i^* : \mathbb{F} \to \mathbb{F}$ be the function $U_i^*(r) = \tilde{V}_i(Z_i(r))$, and let $u_i^* \in \mathbb{F}$ be the value $u_i^* = \tilde{V}_i(z_i)$.

Next, the protocol proceeds in $r-1$ search rounds. In each round the player constructs a set of k indices $M \subseteq I$ (M contains indices that partition the remaining search range into almost equal parts). The player sends the curves Z_i $(i \in M)$ to P_1, and the points z_i $(i \in M)$ to P_0. P_1 replies with the functions $U_i : \mathbb{F} \to \mathbb{F}$ $(i \in M)$, and P_0 replies with the values $u_i \in \mathbb{F}$ $(i \in M)$. The honest P_1 should reply with $U_i = U_i^*$, and the honest P_0 should reply with $u_i = u_i^*$.

The goal of Stage 1 is to find two layers $i' > i'' \in I$, $i' - i'' \leq k$ such that $U_{i'}(\zeta_{i'}) = u_{i'}$, but $U_{i''}(\zeta_{i''}) \neq u_{i''}$. By comparing the values of $U_i(\zeta_i)$ and u_i $(i \in M)$, the player decides on the smaller search range for the next round, until the desired layers i' and i'' are found.

Note that during the first round the player checks the provers agreement on the input and output layers. If $U_0(\zeta_0) = u_0$ (the provers agree on the output $C(x)$), then since at least one of the provers is truthful, the player outputs u_0. If $U_d(\zeta_d) \neq u_d$ (the provers disagree on a point in the low-degree extension $\tilde{x} = \tilde{V}_d$ of the input), the player queries $\tilde{x}(z_d)$. Since at least one of the provers is truthful, the queried value $\tilde{x}(z_d)$ is either $U_d(\zeta_d)$ or u_d. If $\tilde{x}(z_d) = U_d(\zeta_d)$, the player outputs 1. If $\tilde{x}(z_d) = u_d$, the player outputs 0.

7.2 Stage 2: Run the One-Round Protocol

Let C' be the circuit composed of layers $i', \ldots, i''$ of C. Run the one-round protocol on C' and $x' = V_i$, using the questions that were already constructed in Stage 1, and with the following changes. Return the obtained value.

- Instead of querying $\mathsf{LDE}_s(x')$ at the point $z_{i'}$, use the value $u_{i'}$.
- Instead of returning the computed value v_0 in the last step, return 0 if $v_0 = u_{i''}$ and 1 otherwise.

8 Very Efficient Player (Proof of Theorem 3.5)

In this section we prove Theorem 3.5. That is, we assume in addition to the previous assumptions that the circuit C is $\mathit{log}(s)$-space uniform and show that in this case the player can run in time $d^{\frac{1}{r}}\mathit{polylog}(s)$.

First observe that each of the check steps of the protocol described in Section 6, excluding the check steps $S_{Q_{i,2m}}$ $(i \in I_{-d})$, can be preformed by the player in $\mathit{polylog}(s)$ time and without knowing the circuit C. The only check steps that require knowing the circuit C are the check steps $S_{Q_{i,2m}}$ $(i \in I_{-d})$, as explained next.

The check step $S_{Q_{i,2m}}$ $(i \in I_{-d})$ requires the player to compute the value $\Phi_i(z_i, w_{i,0}, w_{i,1})$ given the (supposed) values of $\tilde{V}_{i+1}(w_{i,0})$ and $\tilde{V}_{i+1}(w_{i,1})$. Recall that Φ_i was defined in Lemma 5.2 as $\Phi_i(z, w_0, w_1) = \tilde{\beta}_i(z, w_0, w_1)\tilde{V}_{i+1}(w_0)\tilde{V}_{i+1}(w_1)$. Therefore, in order to compute $\Phi_i(z_i, w_{i,0}, w_{i,1})$, the player needs to know $\tilde{\beta}_i(z_i, w_{i,0}, w_{i,1})$. Note that $\tilde{\beta}_i$ depends on the circuit C, and that for a general size s depth d circuit C, the evaluation of $\tilde{\beta}_i$ may be pricy and require $\mathit{poly}(s)$ time. Moreover, even in the case where the circuit C is $\mathit{log}(s)$-space uniform, the evaluation of $\tilde{\beta}_i$ by the player may require $\mathit{poly}(s)$ time.

The crucial observation is that in the case of a $\mathit{log}(s)$-space uniform circuit C, the function $\tilde{\beta}_i$ is computable in space $O(\mathit{log}(s))$. Note, that the inputs for the function $\tilde{\beta}_i$ are themselves of length $O(\mathit{log}(s))$. Hence, the function $\tilde{\beta}_i$ can be computed in space polynomial in the length of its inputs, that is $\tilde{\beta}_i \in \mathsf{PSPACE}$.

Recall that for every language in PSPACE, Feige and Kilian give a one-round competing provers protocol that only requires polynomial communication complexity [3]. Therefore, there is a one-round competing provers protocol that computes $\tilde{\beta}_i$, with communication complexity polynomial in the length of the inputs for $\tilde{\beta}_i$, that is, communication complexity of $\mathit{polylog}(s)$. Thus, the player cannot evaluate the function $\tilde{\beta}_i$ in time $\mathit{polylog}(s)$ by herself, but she can do that with the help of the two competing provers.

This naturally leads to an $r+2$ rounds protocol (rather than the r rounds protocol that we want). We start by describing the $r+2$ rounds protocol and then show how to collapse the last two rounds of the protocol to round r of the protocol.

8.1 The $r+2$ Rounds Protocol

The first r rounds of the protocol run the r rounds of the protocol of Section 7, except for the player's decision that is still not made. In order to make her decision, the player needs to evaluate $\tilde{\beta}_i(z_i, w_{i,0}, w_{i,1})$ for each of the at most k layers $i \in \{i'', ..., i'-1\}$ that round r of the protocol considers (see Subsection 7.2).

In order to compute the needed values of $\tilde{\beta}_i(z_i, w_{i,0}, w_{i,1})$, the player sends in round $r+1$ the values $(z_i, w_{i,0}, w_{i,1})$ (for $i \in \{i'', ..., i'-1\}$) to both provers, and asks each prover to reply with the values of $\tilde{\beta}_i(z_i, w_{i,0}, w_{i,1})$ (for $i \in \{i'', ..., i'-1\}$). If the provers agree on all the values $\tilde{\beta}_i(z_i, w_{i,0}, w_{i,1})$, the player knows that these values are correct (because at least one of the provers is honest), and can proceed with her decision. If the provers disagree on one or more of the values $\tilde{\beta}_i(z_i, w_{i,0}, w_{i,1})$, the player picks one such value and applies in round $r+2$ the PSPACE protocol of Feige and Kilian in order to identify the correct value and decide which prover is honest.

Next, we show how to reduce the number of rounds from $r+2$ to r.

8.2 Collapsing Round $r+1$ to Round r

We now describe how to collapse round $r+1$ of the protocol to round r. We cannot just send all the values $(z_i, w_{i,0}, w_{i,1})$ (for $i \in \{i'', ..., i'-1\}$) to both provers at the beginning of round r, because this would give each prover additional information that may enable him to cheat in round r.

Note, however, that at the beginning of round r, the prover P_1 gets all the curves Z_i (for $i \in \{i'', ..., i'\}$). Hence, for $i \in \{i'', ..., i'-1\}$, the prover P_1 already knows the needed values of $w_{i,0}, w_{i,1}$ (since $Z_{i+1}(0) = w_{i,0}$ and $Z_{i+1}(1) = w_{i,1}$), as well as the curve Z_i that contains the point z_i. Thus, we can just ask P_1 for the values of $\tilde{\beta}_i(z'_i, w_{i,0}, w_{i,1})$ for all the points z'_i on the image of Z_i.

In addition, at the beginning of round r, the prover P_0 gets all the points z_i as well as the curves $D_{i,2m}$ (for $i \in \{i'', ..., i'-1\}$). Hence, for $i \in \{i'', ..., i'-1\}$, the prover P_0 already knows the needed value of z_i, as well as the curve $D_{i,2m}$ that contains $w_{i,0} \circ w_{i,1}$ (since $D_{i,2m}(\delta_{i,2m}) = a_{i,2m} = w_{i,0} \circ w_{i,1}$). Thus, we can just ask P_0 for the values of $\tilde{\beta}_i\left(z_i, w'_{i,0} \circ w'_{i,1}\right) = \tilde{\beta}_i\left(z_i, w'_{i,0}, w'_{i,1}\right)$ for all the points $w'_{i,0} \circ w'_{i,1}$ on the image of $D_{i,2m}$.

Denote by $A^*_i, B^*_i : \mathbb{F} \to \mathbb{F}$ the following restrictions of the function $\tilde{\beta}_i$ to curves: $A^*_i(t) = \tilde{\beta}_i(Z_i(t), w_{i,0}, w_{i,1})$ and $B^*_i(t) = \tilde{\beta}_i(z_i, D_{i,2m}(t))$ (for $i \in \{i'', ..., i'-1\}$). The new protocol asks P_1 to add to his reply message in round r the polynomials A^*_i, and asks P_0 to add to his reply message in round r the polynomials B^*_i (for $i \in \{i'', ..., i'-1\}$).

Suppose that P_1 replied with $A_i : \mathbb{F} \to \mathbb{F}$ and that P_0 replied with $B_i : \mathbb{F} \to \mathbb{F}$

(for $i \in \{i'', ..., i'-1\}$). Observe that $A_i^*(\zeta_i) = \tilde{\beta}_i(Z_i(\zeta_i), w_{i,0}, w_{i,1}) = \tilde{\beta}_i(z_i, w_{i,0}, w_{i,1})$ and that $B_i^*(\delta_{i,2m}) = \tilde{\beta}_i(z_i, D_{i,2m}(\delta_{i,2m})) = \tilde{\beta}_i(z_i, a_{i,2m}) = \tilde{\beta}_i(z_i, w_{i,0}, w_{i,1})$. If for every $i \in \{i'', ..., i'-1\}$ it holds that $A_i(\zeta_i) = B_i(\delta_{i,2m})$, the player may use the value $A_i(\zeta_i) = B_i(\delta_{i,2m})$ as $\tilde{\beta}_i(z_i, w_{i,0}, w_{i,1})$ and complete her decision protocol. Otherwise, she proceeds to round $r+2$.

Thus, we showed how to collapse round $r+1$ of the protocol to round r.

8.3 Collapsing Round $r+2$ to Round r

We now show how to collapse round $r+2$ to round r, resulting in an r rounds protocol. Round $r+2$ is only reached if there exists $j \in \{i'', ..., i'-1\}$ such that $A_j(\zeta_j) \neq B_j(\delta_{j,2m})$. In this case, in order to decide which of the provers is truthful, the player runs the one-round protocol of [3].

In round $r+2$, if reached, the player and provers run the one-round protocol of [3] in order to determine the value of $\tilde{\beta}_j(z_j, w_{j,0}, w_{j,1})$, for some $j \in \{i'', ..., i'-1\}$ and some $z_j, w_{j,0}, w_{j,1}$, where P_0 claims that the value is $B = B_j(\delta_{j,2m})$ and P_1 claims that the value is $A = A_j(\zeta_j)$. We would like to collapse round $r+2$ to round r, but the problem is that after the questions part of round r, the values of $j, z_j, w_{j,0}, w_{j,1}, A, B$ are still not all known to the player or the provers. Note, however, that the number of possibilities for these values is small from the point of view of each of the parties.

First note that j can take at most k values and A, B can each take at most $p = |\mathbb{F}|$ values. Also, after the questions part of round r, the player and P_1 already know $w_{j,0}, w_{j,1}$, and from the point of view of P_0 the element $w_{j,0} \circ w_{j,1}$ can take at most p values (as $w_{j,0} \circ w_{j,1}$ is on the image of $D_{j,2m}$, which is of size at most p). As for z_j, after the questions part of round r, the player and P_0 already know z_j, and from the point of view of P_1 the point z_j can take at most p values (as z_j is on the image of Z_j, which is of size at most p).

Let Λ be the set of all $\lambda = \left(j', z'_{j'}, w'_{j',0}, w'_{j',1}, A', B'\right)$, such that, $j' \in \{i'', ..., i'-1\}$, and $z'_{j'}$ is on the image of $Z_{j'}$, and $w'_{j',0} \circ w'_{j',1}$ is on the image of $D_{j',2m}$, and $A', B' \in \mathbb{F}$. For every $\lambda \in \Lambda$, let $\mathcal{Q}(\lambda) = (\mathcal{Q}_0(\lambda), \mathcal{Q}_1(\lambda))$ be independently chosen pair of questions for the Feige-Kilian protocol corresponding to λ (that is, for $\lambda = \left(j', z'_{j'}, w'_{j',0}, w'_{j',1}, A', B'\right)$, the Feige-Kilian protocol for evaluating $\tilde{\beta}_{j'}\left(z'_{j'}, w'_{j',0}, w'_{j',1}\right)$ when P_0 claims that the value is B' and P_1 claims that the value is A').

We can now collapse round $r+2$ to round r as follows. In the questions part of round r, in addition to the questions of the original protocol of Section 7, the player will send to P_0 the pair $(\lambda, \mathcal{Q}_0(\lambda))$, for every $\lambda = \left(j', z'_{j'}, w'_{j',0}, w'_{j',1}, A', B'\right) \in \Lambda$, such that, $z'_{j'} = z_{j'}$. In the same way, the player will send to P_1 the pair $(\lambda, \mathcal{Q}_1(\lambda))$, for every $\lambda = \left(j', z'_{j'}, w'_{j',0}, w'_{j',1}, A', B'\right) \in$

Λ, such that, $\left(w'_{j',0}, w'_{j',1}\right) = (w_{j',0}, w_{j',1})$. Note that this doesn't give the provers additional information about the questions of the original protocol of Section 7, because each prover could have constructed by himself questions that are distributed the same as the additional questions that he got.

In the reply part of round r, the prover P_0 will send his reply for the question $\mathcal{Q}_0(\lambda)$ for the Feige-Kilian protocol corresponding to λ, for every pair $(\lambda, \mathcal{Q}_0(\lambda))$ that he got. This is in addition to the replies for the questions of the original protocol of Section 7, and in addition to the functions B_i (required in order to collapse round $r+1$ to round r, as discussed above). In the same way, the prover P_1 will send his reply for the question $\mathcal{Q}_1(\lambda)$ for the Feige-Kilian protocol corresponding to λ, for every pair $(\lambda, \mathcal{Q}_1(\lambda))$ that he got (this is in addition to the replies for the questions of the original protocol of Section 7, and in addition to the functions A_i). Denote the reply of P_0 for the question $\mathcal{Q}_0(\lambda)$ by $\mathcal{R}_0(\lambda)$, and denote the reply of P_1 for the question $\mathcal{Q}_1(\lambda)$ by $\mathcal{R}_1(\lambda)$.

The decision made by the player is as follows. If for every $i \in \{i'', ..., i'-1\}$ it holds that $A_i(\zeta_i) = B_i(\delta_{i,2m})$, the player may use the value $A_i(\zeta_i) = B_i(\delta_{i,2m})$ as $\tilde{\beta}_i(z_i, w_{i,0}, w_{i,1})$ and make her decision according to the decision part of the original protocol of Section 7. If for some $j \in \{i'', ..., i'-1\}$ it holds that $A_j(\zeta_j) \neq B_j(\delta_{j,2m})$, the player decides according to the decision part of the Feige-Kilian protocol corresponding to $\lambda = (j, z_j, w_{j,0}, w_{j,1}, A_j(\zeta_j), B_j(\delta_{j,2m}))$. This can be done since both provers have sent replies $\mathcal{R}_0(\lambda)$ and $\mathcal{R}_1(\lambda)$, respectively.

Thus, we obtained an r rounds protocol. Note that the size of Λ is $k \cdot polylog(s)$. Hence, the total communication of the new protocol is $d^{\frac{1}{r}} polylog(s)$, as required. Since the error of the original protocol of Section 7 is ϵ, and the error of the protocol of Feige and Kilian is negligible and in particular is smaller than $\frac{\epsilon}{|\Lambda|}$, the total error of the new protocol is at most 2ϵ (by the union bound).

8.4 Running Time of the Player

The running time of the player in the protocol described in the previous subsections is $d \cdot polylog(s)$, rather than the desired $d^{\frac{1}{r}} polylog(s)$. This is because in the original protocol of Section 7, the player generates for every layer $i \in I_{-d}$ the corresponding curves and points, at the beginning of the protocol, which takes time $d \cdot polylog(s)$. To reduce the running time of the player to $d^{\frac{1}{r}} polylog(s)$, note that the player only needs to generate the curves and points corresponding to layers $i \in I_{-d}$ that are actually used.

More precisely, define the questions corresponding to layer $i \in I_{-d}$ to be the set of curves $C_{i,\ell}$ and $D_{i,\ell}$ ($\ell \in L$), the curve Z_{i+1}, and the point z_{i+1}. Note that the questions for layer i can be generated independently of the questions for all the other layers.

We change the protocol, so that the player generates the questions corresponding to layer i only if and when layer i is used. That is, the questions corresponding to layer i are generated during round $r' \in [r]$ iff i is one of the k indices queried in round r'. Note that since only a total number of $O(r \cdot k) \leq O\left(log(d) \cdot d^{\frac{1}{r}}\right)$ indices are queried, the player only spends $d^{\frac{1}{r}} polylog(s)$ time constructing questions.

References

[1] Scott Aaronson and Avi Wigderson. Algebrization: A new barrier in complexity theory. *TOCT*, 1(1), 2009.

[2] Ran Canetti, Ben Riva, and Guy Rothblum. Verifiable computation with two or more clouds. *Workshop on Cryptography and Security in Clouds, March 15-16, 2011 in Zurich, Switzerland*, http://www.zurich.ibm.com/ cca/csc2011/submissions/riva.pdf.

[3] Uriel Feige and Joe Kilian. Making games short (extended abstract). In *STOC*, pages 506–516, 1997.

[4] Uriel Feige, Adi Shamir, and Moshe Tennenholtz. The noisy oracle problem. In *CRYPTO*, pages 284–296, 1988.

[5] Shafi Goldwasser, Yael Tauman Kalai, and Guy N. Rothblum. Delegating computation: interactive proofs for muggles. In *STOC*, pages 113–122, 2008.

[6] Yael Tauman Kalai and Ran Raz. Interactive pcp. In *ICALP (2)*, pages 536–547, 2008.

ECCC ISSN 1433-8092

http://eccc.hpi-web.de