Two 1-Round Protocols for Delegation of Computation

Ran Canetti* Ben Riva† Guy N. Rothblum‡

September 21, 2011

Abstract

Consider a weak client that wishes to delegate computation to an untrusted server and be able to succinctly verify the correctness of the result, all within one round of interaction. We provide solutions for two relaxed variants of this problem. Specifically:

- We consider a model where the client delegates the computation to *two or more* servers, and is guaranteed to output the correct answer as long as even a *single* server is honest. We call this model *Refereed Delegation of Computation (RDoC)*. In this model, we show a 1-round unconditionally statistically sound protocol for any log-space uniform $\mathcal{NC}$ circuit. In contrast, all known one-round delegation protocols with a single server are only computationally sound.

- We consider a model with a non-succinct offline stage and pubic verifiability. (Previously, this model was considered only with private verifiability, namely the client has to maintain some secret local information pertaining to the offline stage [Gennaro et al., CRYPTO 2010]). Public verifiability does away with the secret state, and so allows delegating the offline stage to a "semi-trusted" external third party that is potentially used by many clients, even mutually suspicious ones. It also allows for a stronger, more adaptive notion of soundness.

  In this model we show a 1-round computationally-sound protocol for any circuit C, *even a non-uniform one*. The client runs in time $poly(log(size(C)), depth(C))$, and soundness is guaranteed assuming the existence of collisions resistant hashing and poly-logarithmic PIR. Previously, publicly verifiable one round delegation protocols were known only for functions in log-space uniform $\mathcal{NC}$.

*Boston University and Tel Aviv University, canetti@tau.ac.il.

†Tel Aviv University, benriva@tau.ac.il.

‡Microsoft Research, Silicon Valley Campus, rothblum@alum.mit.edu.

Contents

1 Introduction

An emerging paradigm in modern computing is *pay-per-use* Cloud Computing. As companies and users reduce their computing assets and turn to weaker computing devices, an increasing number and variety of computations are being performed remotely by untrusted parties that may be error-prone or even malicious.

This shift motivates exploring methods for delegating computations reliably: a weak client delegates his computation to a powerful server. After the server returns the result of the computation, the client should be able to verify the correctness of that result using considerably *less* resources than required to actually *perform* the computation from scratch.

Prior work has considered using an interactive protocol for delegating computation from a client to an untrusted server. These works ask for protocols where the client does not work *too much* (ideally, the client will only have to do work that is quasilinear in the input length). There has been a rich body of work on this question. Kilian [Kil92] proposes a protocol that uses two rounds of interaction for delegating any polynomial-time computation. Micali [Mic00] "squashes" the Kilian protocol to one round in the Random Oracle model. These protocols enjoy public verifiability, namely the client keeps no private state. A non publicly verifiable, one round version of these protocols is shown to be sound based on non-standard extractability assumptions [CL08, BCCT11, GLR11]. The soundness guarantee of these protocols is only computational.

Goldwasser, Kalai and Rothblum [GKR08] propose a delegation protocol with statistical soundness. However, the protocol works only for log-space uniform $\mathcal{NC}$ circuits, and the number of rounds is quasilinear in the depth of the circuit.

A relaxation of the model, suggested by [GGP10], considers two stages of the protocol, offline and online. In the offline stage we fix a function (or a circuit) f and allow the client to work *harder* (e.g., proportional to the size of f). In the 1-round online stage, the client can delegate the computation of $f(x)$ for any input x, and can verify the result in time much smaller than the size of f. However, in all the recent works in this model (i.e. [GGP10, CKV10, AIK10, Chu11, Ben11]), the result of the offline stage must be kept secret from the server — or else, a malicious server can cheat. This forces the client to perform the expensive offline stage by herself (or by a completely trusted entity).

1.1 Our Contributions

We present two natural extensions of the existing models. In each one of these extended models we show a protocol with properties that are not known to be achievable by previous constructions. More specifically:

Refereed Delegation of Computation (RDoC) Model. Instead of delegating the computation to a single server, we let the client interact with *two or more* servers, and require that the client outputs the right value as long as there exists *one* server that follows the protocol. That is, the client asks for the value of $f(x)$ from several servers. In case they make contradictory claims about $f(x)$, they "play" against each other in a protocol where the weak client can efficiently determine the true claim as long as there is at least *one* honest server. As for the efficiency, we require that the computational requirements from an honest server are not much more than those required to compute the function in the first place, and that the client's running time would be much smaller than required to compute the function. (If the client interacts with three servers, at least a majority of them follow their protocols, then it suffices to simply ask each server for the answer and take the majority answer. Even if we assume there is only one honest server, the client can still detect inconsistency and compute the problem on its own. However, these solutions are not satisfactory for us.)

A closely related model to ours is the *Refereed Games (RG)* model of Feige and Kilian [FK97] where they focus on two unbounded competing servers and polynomial time referee/client. However, here we are faced with the additional challenges of building protocols with efficient honest servers, with super-efficient client and for any number of servers. Indeed, our model can be considered also as *refereed games with efficient servers* and super-efficient clients.

Another related model is the multi-prover model (MIP). Here a verifier/client interacts with two or more provers. However, in the MIP model, soundness is guaranteed only if no two malicious provers/servers can communicate or coordinate their strategies during the protocol. This assumption is arguably less realistic for cloud computing. Moreover, the definition of MIP allows cases where even a single malicious prover/server can cause the verifier/client to reject the proof of valid statements.

A 1-round unconditionally-sound RDoC for any circuit computable in $\mathcal{L}$-uniform $\mathcal{NC}$. We show a one-round delegation of computation (RDoC) protocol in this model with statistical soundness. The protocol adapts techniques from the work of Feige and Kilian [FK97], who construct a RDoC protocol where the servers are inefficient even for log-space computations, together with ideas and techniques from the work of Goldwasser, Kalai and Rothblum [GKR08], and some new techniques.

We provide a brief overview of the protocol. For the description here we restrict attention to the case when there are exactly two servers, one honest and one malicious (but the referee/client does not know which is honest). We later show how to extend our protocols to more than two servers.

In high level, our protocol follows the structure of the [GKR08] interactive proof. We view the computation as a circuit. The servers make claims about the output layer of the circuit, and we use a (very efficient) sum-check protocol to reduce a claim about a high layer in the circuit, which we call an input claim, into a claim about a lower layer (closer to the circuit's input layer), we call this an output claim. The guarantee is that if the input claim is false, then w.h.p. over the referee's coins the output claim will also be false. They use this sub-protocol to reduce the claim about the circuit's output layer into a claim about the circuit's input layer, and complete the protocol by noting that claims about the input layer can be verified by the referee in quasi-linear time.

However, the [GKR08] protocol is highly interactive: First, each sum-check sub-protocol requires a logarithmic number of rounds. Second, the claim for each layer in the circuit depends on the coins chosen by the referee in the sum-check for the layer above it, so all of these sum check protocols must be run sequentially from the top circuit output layer to the bottom circuit input layer. To eliminate the first source of interaction, we use a variant of the one-round refereed game for the sum check test from [FK97]. This still leaves us with a significant technical obstacle: How can we collapse all of sum-check protocols from the different layers into just one round of interaction? The difficulty comes from the fact that, in order to run the [FK97] protocol, both servers need to know the claim being debated. This claim, however, depends on the referee's (non-public) coins in the sum check for the layer above. Revealing all of those coins to both servers ahead of time would compromise soundness.

We overcome this obstacle (and additional lower-level ones) using techniques tailored to our setting. In a nutshell, the claim for each layer is the value of a low-degree multi-variate polynomial (say p) on a certain secret point (say z) that is known only to the referee. The referee sends to each server a different low-degree parametric curve passing through the point z (but also through many others), and asks for the (low-degree) polynomial q describing p restricted to that server's curve. Essentially, soundness follows because each server (on its own) cannot tell which of the points on its curve is the one that the referee will be checking. If the server cheats and sends $q' \neq q$, then (since q and q' are low degree polynomials over a larger field) with high probability the server must be cheating on the point z that the referee is checking on.

In September 2011 Kol and Raz have posted an alternative exposition of this protocol [KR11]. That paper contains also an extension of this protocol that somewhat reduces the workload of the client at the price of a comparable increase in the number of rounds. We note that our protocol was posted at [CRR11b] and described at [CRR11c].

Another protocol in this model is described in [CRR11a]. That protocol has computational soundness and takes a logarithmic number of rounds.

Publicly Verifiable Delegation of Computation (PCDoC). The recently used model for offline/online delegation of computation [GGP10, CKV10, AIK10, Chu11, Ben11] assumes that the client can work "hard"

during the offline stage, or alternatively rely on the assistance of a trusted party that will perform the offline stage on the client's behalf. However, all the recent protocols in this model require the client to keep some secret information that pertains to the offline stage. This secret information is then used in the verification process. Having such secret information is a serious impediment. First, it requires the client to put complete trust in the entity that participates in the offline stage. Furthermore, in some of the protocols soundness is only guaranteed as long as the server does not know which past interactions convinced the client.

We define a similar model in which avoids these impediments. Instead of an offline stage that ends with a secret key, the offline stage ends with a public data that: (1) can be verified by anyone, and, (2) can be used by anyone to verify the computation of any input (in the online stage).

Imagine that some well-known company (e.g., Microsoft, Google) publishes short public keys for a set of different circuits. (Or, Microsoft ships these keys as part of its products, as is currently done with Certificate Authorities.) Any interested party can also check these values and verify that they are correct. Later on, a (weak) client can take these values and delegate its computation to *any* server, without running the offline stage by itself.

A 1-round computationally-sound PVDoC for non-uniform circuits. We give a brief overview of the protocol. Our starting point is again the [GKR08] protocol, squashed by [KR09] to a single round trip. The difficulty that prevents this protocol from working for larger circuits is that the client cannot verify claims about the circuit structure because the explicit circuit was too large (larger than the client running time) and (for general non-uniform circuits) there was no shorter implicit representation. We solve this difficulty by adding the following offline stage: First, all possible queries on the structure of the circuit to be evaluated are computed in advance. Next, the answers to all these queries are Merkle tree hashed using a function that is chosen from a family of collision resistant hash functions. The output of the offline stage is the root of the Merkle tree hashing. Notice that the same root can be reused in an online stage by *any* interested client delegating a computation to *any* server. (These values do not depend on the input, but only on the circuit itself.) Since the circuit is publicly known, anyone can re-compute the root and publicly verify that it is indeed the correct value. In the online stage, the client runs in time $poly(log(size(C)), depth(C))$.

We remark that the same techniques can be also applied directly on the [GKR08] protocol (i.e., without using the [KR09] transformation). In addition to supporting more general circuits, the resulting protocol is (arguably) simpler to understand and potentially easier to implement than the full [GKR08] protocol. Furthermore, although the resulting protocol requires $poly(log(size(C)), depth(C))$ number of rounds, the practicality of each sub-protocol ([GKR08] and Merkle Tree) is fairly reasonable compared to previous constructions (e.g., that use fully homomorphic encryption or PIR with huge databases).

1.2 Organization

Section 2 reviews prior work. Section 3 describes the protocol of [GKR08] which we will use extensively in our constructions. Section 4 defines the model of refereed delegation of computation, shows a "parallel repetition" theorem of RDoC protocols and describes how to extend RDoC with two servers to any number of servers. Furthermore, it reviews the protocol of [FK97] which we use in our construction and presents the construction of one-round RDoC for any $\mathcal{L}$-uniform $\mathcal{NC}$ computation. Section 5 defines the model of publicly verifiable delegation of computation, and presents the construction of one-round computationally sound PVDoC for any circuit.

2 Prior Work

Prior work has studied the question of proving the correctness of general computations. (Most previous works focused on interactive proofs between a verifier and a prover. However, given an interactive protocol for proving the correctness of a computation of f, one can easily get verifiable delegation of computation by asking the server for $y = f(x)$ and a proof that y is the correct result.) Babai *et al.* [BFLS91] consider this question in a setting where the prover is a non-adaptive oracle. Kilian [Kil92] and Micali [Mic00] build

on their techniques and show efficient computationally sound protocols, whose security is based on cryptographic assumptions and where soundness holds only against computationally bounded cheating provers. Micali gets a non-interactive computationally sound proof based on the existence of a Random Oracle whereas Kilian gets a two-round interactive computationally sound proof assuming the existence of collision resistance hash family.

Goldwasser *et. al.* [GKR08] present an information theoretically sound interactive proof protocol for verifiable computation for any language in $\mathcal{L}$-uniform $\mathcal{NC}$. The number of rounds between the prover and the verifier is poly-logarithmic in the size of the computation. Using the technique of Kalai and Raz [KR09] this protocol can be transformed into a one-round protocol, assuming the existence of a computational Private Information Retrieval scheme with poly-log communication.

Gennaro *et al.* [GGP10], Chung *et. al.* [CKV10] and Applebaum *et al.* [AIK10] consider a model with an offline stage. Based on the existence of a fully homomorphic encryption, they construct computationally sound protocols, where in the offline stage the verifier runs in time proportional to the size of the computation. Afterwards, in an online stage, the verifier (using the secret result of the online stage) runs in time proportional to the size of its inputs and the computation results. In these works, as long as the verifier does not encounter cheating provers, the same pre-processing information can be used in multiple rounds, yielding improved amortized complexity. Recently, Benabbas *et al.* [Ben11] and Chung *et al.* [Chu11] show protocols in which security is preserved even if the verifier encounter cheating provers. We stress that in all the above works, the offline stage ends with a secret key that is known only to the verifier.

A related proof model with several provers is the model of Multi-Prover Interactive Proofs, suggested by Ben-Or *et al.* [BOGKW88]. In this model, even if *all* of the provers cheat, the verifier will detect that they are cheating. However, soundness is guaranteed assuming that malicious provers *cannot* communicate or coordinate their strategies during the protocol. This is in contrast to the refereed games of Feige and Kilian [FK97] and to our RDoC model, where soundness is guaranteed as long as one server is honest, even if some group of malicious servers communicate *during* the protocol. In addition, the referee learns who are the cheating provers.

3 The Protocols of [GKR08, KR09]

Given that our protocols rely heavily on the structure of the [GKR08, KR09] protocols, we start with a brief exposition of them.

3.1 Preliminaries: Low Degree Extension (LDE)

Given a field F, a subset $H \subseteq F$ and a function $f : H^m \to F$, we let the low degree extension of f, denoted $\widetilde{f} = LDE(f)$, be the *unique* multi-variate polynomial $\widetilde{f} : F^m \to F$ that satisfies:

- (low-degree) $deg(\widetilde{f}) < |H|$ for each variable.
- (extension) $f(x) = \widetilde{f}(x)$ for all $x \in H^m$.

Such polynomials can be constructed using Lagrange Interpolation.

Similarly we define the low degree extension of a vector. Let $\alpha : H^m \to \{0, \ldots |H|^m - 1\}$ be the lexicographic order of H^m. Given a vector $\vec{w} = (w_0, \ldots w_{k-1}) \in F^k$, where $k \leq |H|^m$, we can view this vector as a function $f_{\vec{w}} : H^m \to F$ such that $f_{\vec{w}}(z) = w_{\alpha(z)}$ when $\alpha(z) \leq k - 1$ and $f_{\vec{w}}(z) = 0$ otherwise. We define the low degree extension of the vector $\vec{w}$ to be $LDE(f_{\vec{w}})$.

3.2 The Bare-Bones Protocol, Given a Circuit Specification Oracle

Notations and Parameters: The protocol is between a server and a client, where both know the input x of length k.

Given an arithmetic circuit $C : \{0,1\}^k \to \{0,1\}$ of fan-in 2 gates, size S and depth d, the players choose the following parameters: 1) An extension field H of $GF[2]$ such that $max(d, log(S)) \leq |H| \leq$

$poly(d, log(S))$, 2) An integer m such that $S \leq |H|^m \leq poly(S)$, 3) An extension field F of H such that $|F| \leq poly(|H|)$. (The size of F influences the soundness of the protocol.).

Using standard techniques, we can transform the arithmetic circuit C to a new arithmetic circuit $C' : F^k \rightarrow F^S$ over the field F with the following properties: 1) C' is of size $poly(S)$ and depth d, with fan-in 2 gates, 2) Each layer, except for the input layer, is of size S (simply by adding dummy gates), 3) For every $(x_1, \ldots, x_k) \in \{0,1\}^k$, $C'(x_1, \ldots, x_k) = (C(x_1, \ldots, x_k), 0, \ldots, 0)$.

Let $spec_c()$ be the predicate describing a circuit C. That is, $spec_c(i, b, w_1, w_2, w_3)$ returns 1 if in the i-th layer of C there is a gate that connects wires w_2 and w_3 to wire w_1, and this gate is a b-gate where $b \in \{0 = add, 1 = mult\}$.[1] Let $\widetilde{spec}_{c(i)}$ be the low degree extension of $spec_c(i, \cdot, \cdot, \cdot, \cdot)$ with respect to H, F and m, of degree δ (that depends on $spec_c()$) in each variable. In this section we assume the client has an oracle access to $\widetilde{spec}_{c'(i)}$.

We denote the output layer as the 0 layer and the other layers according to their distance from the output layer. The input layer is the d-th layer. For $0 \leq i \leq d$ we associate a vector $v_i = (v_{i,0}, \ldots, v_{i,S-1}) \in F^S$ with the values of all gates of the i-th layer in the computation of $C'(x_1, \ldots, x_k)$. v_0 is the circuit result $(C(x_1, \ldots, x_k), 0, \ldots, 0)$ and v_d is the circuit input $(x_1, \ldots, x_k)$. Let $\widetilde{V}_i : F^m \rightarrow F$ be the low degree extension of the vector v_i with respect to H, F and m. This polynomial is of degree $\leq |H| - 1$ in each of its variables, and given v_i can be computed in time $\leq poly(|F|^m) = poly(S)$. Since v_d is of length k, $\widetilde{V}_d$ can be computed in time $\leq k \cdot poly(|H|, m)$.

The Protocol: The server claims that $C'(x_1, \ldots, x_k) = (0, \ldots, 0)$. An interactive protocol is executed between the server and the client. In each step the server reduces the correctness of the computation of layer i to the correctness of the computation of layer $i + 1$. Concretely, for layer i, the server claims that $\widetilde{V}_i(u_i) = r_i$ for some randomly chosen u_i that the client picked and sent to the server. Then, the server reduces the correctness of this claim to the correctness of $\widetilde{V}_{i+1}(u_{i+1}) = r_{i+1}$ for some randomly chosen u_{i+1} that the client picked. This process continues until they reach the input layer and then the client verifies the correctness of this layer by himself (as $\widetilde{V}_d$ is small and known to the client).

We now describe in detail the reduction between the layers. Let $f_u^{(i)} : (F^m)^3 \rightarrow F$ be the a $3m$-variate polynomial of size $\leq poly(S)$ and degree $\leq 2\delta$ defined by

$$f_u^{(i)}(p, w, w') = \widetilde{\beta}(u, p) \cdot [\ \widetilde{spec}_{c'(i+1)}(0, p, w, w')(\widetilde{V}_{i+1}(w) + \widetilde{V}_{i+1}(w')) \ + \ \widetilde{spec}_{c'(i+1)}(1, p, w, w')(\widetilde{V}_{i+1}(w) \cdot \widetilde{V}_{i+1}(w'))\ ]$$

where $\widetilde{\beta}(u, p)$ is a $|H| - 1$ degree polynomial that depends only on F, H and m, and, can be computed in time $poly(|H|, m)$ (see [Rot09]). Note that given an oracle access to $\widetilde{spec}_{c'(i)}$ and the values of $\widetilde{V}_{i+1}(w)$ and $\widetilde{V}_{i+1}(w')$, the polynomial $f_u^{(i)}(p, w, w')$ can be evaluated in time $poly(|H|, m)$.

Now, given a claim for layer i that $\widetilde{V}_i(u_i) = r_i$ for some randomly chosen u_i that the client picked and sent to the server, it can be shown that $\widetilde{V}_i(u_i) = \sum_{p,w,w' \in H^m} f_{u_i}^{(i)}(p, w, w')$. Thus, proving that $\widetilde{V}_i(u_i) = r_i$ is equivalent to proving that $r_i = \sum_{p,w,w' \in H^m} f_{u_i}^{(i)}(p, w, w')$.

This part is done by a standard *sum-check* interactive protocol between the two players. For each layer of the circuit, the client and the server execute a sum-check interactive protocol that consists of $3m$ rounds. The last step of the sum-check requires a computation of $f_{u_i}^{(i)}(p, w, w')$ by the client. In order to do that, the server sends a low degree polynomial $\widetilde{V}_{i+1}(\gamma(t))$ where $\gamma(t)$ is the 1-degree curve that passes through w and w'. Using this polynomial, the client computes $\widetilde{V}_{i+1}(w)$ and $\widetilde{V}_{i+1}(w')$ and uses that to compute and verify $f_{u_i}^{(i)}(p, w, w')$. Then, the client picks a random point t' on the curve $\gamma(t)$ and continues to the correctness proof of the claimed value of $\widetilde{V}_{i+1}(\gamma(t'))$ (where $u_{i+1} = \gamma(t')$ in the next round).

[1] It is possible to use any gate that can be expressed as a polynomial of its inputs.

Complexity: It is shown in [Rot09] that by taking F such that $|F| \geq 700md\delta = poly(|H|)$ we get soundness of $\frac{1}{100}$. In addition, the overall running time of the server is $poly(|F|^m) = poly(S)$, the running time of the client is $poly(|F|, m)+k\cdot poly(|H|, m) = k\cdot poly(d, log(S))$ and the communication complexity is $poly(|F|, m) = poly(d, log(S))$.

3.3 Realizing the Oracle for $\mathcal{L}$-uniform $\mathcal{NC}$ Circuits

The above protocol is presented for any circuit given an oracle access to $\widetilde{spec}_{c'(i)}$. [GKR08] shows how to realize the protocol without an oracle access but for a more restricted class of languages, $\mathcal{L}$-uniform $\mathcal{NC}$, which is the class of languages that can be computed by circuits of poly-size and poly-logarithmic depth where there is a log-space Turing Machine that generates those circuits.

Specifically, [GKR08] shows the following:

1. For a language L in $\mathcal{NL}$ (i.e., L has a non-deterministic log-space Turing Machine), it is possible to compute the $\widetilde{spec}_{c(i)}$ (where c is the circuit that computes L) in time $poly(log(k))$. Thus, the client can compute it by himself and, as a result, realize the bare-bones protocol.

2. For a language L in $\mathcal{L}$-uniform $\mathcal{NC}$, the client delegates also the computation of the oracle answers to the server. More specifically, let C be the circuit that computes L and let $TM_{spec(c)}$ the Turing Machine that computes $\widetilde{spec}_{c(i)}$. Since L is $\mathcal{L}$-uniform, $TM_{spec(c)}$ is also non-deterministic log-space Turing Machine, and therefore the computation of $TM_{spec(c)}$ can be delegated to the server. As a result, the bare-bones protocol is executed once for the delegation of C, and d times for the delegations of $TM_{spec(c)}$. Hence, the running time of the client is still $k \cdot poly(d, log(S))$.

3.4 The Transformation of [KR09]

Assuming the existence of a poly-logarithmic cPIR scheme, [KR09] presents a transformation from any public-coin unconditionally-sound proof system into a one-round computationally-sound proof system. In high level the transformation is as follows. The verifier sends all the random coins together in the same round, hidden inside different cPIR queries. The (honest) prover prepares a database with all the possible answers, and returns the answers to the verifier queries all together. Then, the verifier peels the cPIR answers and feeds the original verifier with the results.

The exact transformation is more subtle, and we refer the reader to [KR09] for more details. We note that the transformation does not change the expressiveness of the underlying protocol, and in particular, transforming the protocol of [GKR08] results in a protocol for $\mathcal{L}$-uniform $\mathcal{NC}$ circuits.

We denote the verifier's message in this protocol by GKR-KR$_v(S, d, \lambda)$ given the circuit size S, depth d and security parameter λ. Similarly, we denote the verifier's response by GKR-KR$_p(C, x, q)$ for a given circuit C, input x and queries $q =$GKR-KR$_v(S, d, \lambda)$.

4 Refereed Delegation of Computation

4.1 The Model

A refereed delegation of computation for a function f is a protocol between a referee/client R and N servers $P_1, P_2, \ldots, P_N$. All parties may use local randomness. The referee and the servers receive an input x. The servers claim different results for the computation of $f(x)$ and the referee should be able to determine the correct answer with high probability. We assume that at least one of the servers is honest.

Definition 1 (Refereed Delegation of Computation)**.** *Let $(P_1, P_2, \ldots, P_N, R)$ be an ε-RDoC with N servers for a function f if the following holds:*

- *For any input x and any i, if server P_i is honest then for any $P_1^*, \ldots, P_{i-1}^*, P_{i+1}^*, \ldots, P_N^*$ the output of R is $f(x)$ w.p. at least $1 - \varepsilon$.*

- *The complexity of the referee is at most quasi-linear in $|x|$ and the complexity of the (honest) servers is polynomial in the complexity of evaluating f.*

For completeness of the description, we briefly review the model of Refereed Games [FK97]. A refereed game (RG) for a language L is a protocol between a referee R and two competing *unbounded* servers P_1 and P_2. All three parties may use local randomness. The referee and the servers receive $x \in \{0,1\}^*$. Without loss of generality we can assume P_1 claims that $x \in L$ and P_2 claims that $x \notin L$, and the referee should be able to determine the correct answer with probability at least 2/3.

4.1.1 Parallel Repetition for RDoC

We have the following "parallel repetition" theorem for RDoC for boolean functions.

Theorem 2 (Parallel Repetition for RDoC)**.** *Let $(P_1, P_2, \ldots, P_N, R)$ be a ε-RDoC for a boolean function f, and let $(P_1^k, P_2^k, \ldots, P_N^k, R^k)$ be a RDoC obtained by running $(P_1, P_2, \ldots, P_N, R)$ k times in parallel and in which R^k accepts if and only if R accepted in the majority of the executions. Then, $(P_1^k, P_2^k, \ldots, P_N^k, R^k)$ is a RDoC with error probability $\varepsilon^{poly(k)}$.*

Proof (sketch). We use the fact that parallel repetition reduces the error probability of any interactive proof system, and we build an interactive proof system (P, V) for the language $L = \{\ x \mid f(x) = 1\ \}$ from our RDoC $(P_1, P_2, \ldots, P_N, R)$. Without loss of generality, we assume $x \in L$ and P_1 is an honest server. We view the referee R and the honest server P_1 as the verifier V, and the other servers as the prover P. Similarly, we view P_1^k and R^k as the verifier V^k in the parallel repetition version of (P, V). Since $(P_1, P_2, \ldots, P_N, R)$ is a RDoC, the soundness of (P, V) is bounded by ε. Now, if we assume there are malicious servers $P_2^k, \ldots, P_N^k$ that convince the referee in $(P_1^k, P_2^k, \ldots, P_N^k, R^k)$ with probability p, it means there is a prover P^k that can convince V^k with probability p. However, since the parallel repetition of interactive proofs reduced the error probability to $\varepsilon^{poly(l)}$, p is negligible. □

4.1.2 From Two Servers to N Servers

In the next section we show a protocol for RDoC with two servers. Here we show how, given a RDoC with two servers and negligible error probability, one can construct a RDoC with N servers and negligible error probability, where we only need to assume that at least *one* of them is honest. The idea is to execute the RDoC with two servers between each pair of servers. By the soundness of the RDoC with two servers, with high probability there exists an honest server P_i that convinces the referee in *all* of his "games". The referee outputs the claimed result of P_i.

This solution can be executed in parallel for all pairs, and therefore keeps the number of rounds the same. However, it requires $\frac{N \cdot (N-1)}{2}$ different executions of the protocol.

4.2 One-round RDoC for Any $\mathcal{L}$-uniform $\mathcal{NC}$ Computation

We start by describing the one-round refereed game for the sum-check task of [FK97].

4.2.1 The Protocol of [FK97]

Intuition. We present a variant of the one-round refereed game from [FK97] for the sum-check task. In this task we have a finite field F, a subset of F denoted by H, a fixed number k and a multivariate polynomial $f : F^k \to F$ of degree $\leq d$ in each variable.[2] The referee can evaluate f by himself in polynomial time in the size of f. Server 1 claims that

$$\sum_{x_1, x_2, \ldots, x_k \in H} f(x_1, x_2, \ldots, x_k) = N_0$$

[2]The [FK97] protocol considers $f : \{0,1\}^k \to F$. We extend it to $f : F^k \to F$.

for some value N_0 and Server 2 claims otherwise (denote this value by N_0').

Lund *et al.* [LFKN92] show an interactive proof with one server for the sum-check task. Their protocol requires k rounds. In the first round, the server sends to the client the univariate polynomial $g_1(x) = \sum_{x_2,\ldots,x_k \in H} f(x, x_2, \ldots, x_k)$ and the client checks if $\sum_{x \in H} g_1(x) = N_0$. Then, the client chooses a random element $c_1 \in F$ and sends it to the server. The protocol continues to the next rounds, where in round i (for $i \in [2..k]$) the server sends to the client $g_i(x) = \sum_{x_i,\ldots,x_k \in H} f(c_1, \ldots, c_{i-1}, x, x_{i+1} \ldots, x_k)$ and client checks if $\sum_{x \in H} g_i(x) = g_{i-1}(c_{i-1})$. Then, the client chooses another random element $c_i \in F$ and sends it to the server. In the last round, the client does not send c_k to the server. Instead, he computes $f(c_1, \ldots, c_k)$ by himself, and checks whether it equals to $g_k(c_k)$. Note that the correctness of the protocol requires that the server cannot guess the c_i-s in advance as they are randomly chosen by the client. Actually, this is why the protocol requires k rounds. If the client would have send all the c_i-s in one round, the server could easily cheat.

In order to reduce the number of rounds, the protocol of [FK97] uses information from both servers. Intuitively, instead of asking the server for a fixed prefix along the rounds (i.e., $c_1, c_2, \ldots c_{i-1}$ is the prefix for round i), for each $i \in [1..k]$ the referee asks on many random prefixes of length i. This allows the referee to send all those prefixes in a single round. However now, since the prefixes are not fixed, the referee cannot efficiently do the *consistency check* between $g_i(x)$ and $g_{i-1}(x)$ (i.e., checking that $\sum_{x \in H} g_i(x) = g_{i-1}(c_{i-1})$). So, the referee uses the second server for that. The consistency check is done by asking both servers for the polynomials g_i-s for random prefixes, such that for each length i there is one prefix that both servers receive from the referee. If both servers answer the same for that specific prefix, then by the assumption that one of the servers is honest, this answer is correct.

The Protocol. Following the intuition behind the protocol, we now describe the protocol in detail. Recall that we have a finite field F, a subset of F denoted by H, a fixed number k and a multivariate polynomial $f : F^k \to F$ of degree $\leq d$ in each variable. Server 1 claims that

$$\sum_{x_1,x_2,\ldots,x_k \in H} f(x_1, x_2, \ldots, x_k) = N_0$$

for some value N_0 and Server 2 claims otherwise (denote this value by N_0').

For simplicity, we use the shorthand $a \circ b$ for a vector that is a concatenation of a and b (where a, b are vectors or single elements). We assume the elements of H are $0, 1, \ldots, |H| - 1$. Instead of working with prefixes, all computations are done using low degree parametric curves, which is a more compact representation. (A parametric curve of degree d in $F[t]^j$ is a tuple of j one-parameter polynomials over the field F, each one of degree $\leq d$.)
The protocol is as presented in Figure 1.

Theorem 3. *Let F be a finite field and H subset of F. Let $f : F^k \to F$ be a multivariate polynomial of degree $\leq d$ in each variable and let $N = \sum_{x_1,x_2,\ldots,x_k \in H} f(x_1, x_2, \ldots, x_k)$. The above protocol is a refereed game with the following properties:*

- *If P_1 claims that $N_0 = N$, then he will be declared as the winner with probability $\geq 1 - \frac{|H| \cdot 2k^2 \cdot d}{|F|}$.*
- *If P_1 claims that $N_0 \neq N$, then he will be declared as the winner with probability $\leq \frac{|H| \cdot 2k^2 \cdot d}{|F|}$*

R's computations: For $1 \leq j \leq k$, R chooses random vectors $A_j, B_j \in F^j$ and random elements $a_j, b_j \in F$. Let $C_j(t) \in F[t]^j$ be the unique degree-$|H|$ parametric curve going through

$$(0, A_{j-1} \circ 0), \ldots, (|H|-1, A_{j-1} \circ (|H|-1)), (|H|, B_j)$$

and let $D_j(t) \in F[t]^j$ be the canonical representation of the unique degree-1 parametric curve going though

$$(a_j, C_j(a_j)), (b_j, A_j).$$

For $j = 1 \ldots k$ we define the functions

$$\Phi_j(x_1, \ldots, x_j) = \sum_{x_{j+1}, \ldots, x_k \in H} f(x_1, \ldots, x_j, x_{j+1} \ldots, x_k).$$

Note that

$$\Phi_j(x_1, \ldots, x_j) = \sum_{x_{j+1} \in H} \Phi_{j+1}(x_1, \ldots, x_{j+1}).$$

R sends to P_1: $A_1, \ldots, A_k, B_1, \ldots, B_k$.

P_1 sends to R: $N_0, N_1, \ldots, N_k, F_1(t), \ldots, F_k(t)$, where $N_j = \Phi_j(A_j)$ and $F_j(t) = \Phi_j(C_j(t))$ for $j \geq 1$.

R sends to P_2: $D_1(t), \ldots, D_k(t)$.

P_2 sends to R: $N_0', G_1(t), \ldots, G_k(t)$, where $G_j(t) = \Phi_j(D_j(t))$.

R declares the winner: R chooses random $r \in F$. P_1 losses immediately if: 1) $N_k \neq f(A_k)$, or 2) $F_k(r) \neq f(C_k(r))$, or 3) For some j, $F_j(t)$ has degree greater than $|H| \cdot j \cdot d$ or $\sum_{i=0}^{|H|-1} F_j(i) \neq N_{j-1}$.

P_2 losses immediately if for some j, $G_j(t)$ has degree greater than $j \cdot d$, or, $G_k(r) \neq f(D_k(r))$.

Let's denote by j the smallest number such that $G_j(b_j) = N_i$. If $G_j(a_j) = F_j(a_j)$ then P_1 is declared as the winner, otherwise P_2.

Figure 1: One-round refereed game for the sum-check task

The referee is polynomial in $|H|$ and k, the (honest) servers are polynomial in $|F|^k$ and the communication complexity is polynomial in $|F|$ and k.

Proof (sketch). Let S_1 be the event that

$$\sum_{x_1, x_2, \ldots, x_k \in H} f(x_1, x_2, \ldots, x_k) \neq N_0$$

but P_1 is declared as the winner (i.e., P_2 is the honest server). Let U_i be the event that $F_i(t)$ is indeed $\Phi_i(C_i(t))$, let E_i be the event that $F_i(a_i)$ is indeed $\Phi_i(C_i(a_i))$ and let E' be the event that $F_k(r)$ is indeed $f(C_k(r))$.

$$Pr[S_1] \leq Pr[E' \wedge \neg U_k] + Pr[\exists i \in [1..k] \; s.t. \; E_i \wedge \neg U_i] \leq Pr[E' \wedge \neg U_k] + \sum_{i=1}^{k} Pr[E_i \wedge \neg U_i].$$

By the fact that two distinct univariate degree-t polynomials agree on at most t points we get that

$$Pr[E' \wedge \neg U_k] \leq \frac{|H| \cdot k \cdot d}{|F|},$$

and that

$$Pr[E_i \wedge \neg U_i] \leq \frac{|H| \cdot i \cdot d}{|F|} \leq \frac{|H| \cdot k \cdot d}{|F|}.$$

Thus,

$$Pr[S_1] \leq \frac{|H| \cdot k \cdot d}{|F|} + k \cdot \frac{|H| \cdot k \cdot d}{|F|} \leq (k+1) \cdot \frac{|H| \cdot k \cdot d}{|F|}.$$

Let S_2 be the event that P_1 is the honest server but P_2 is declared to be the winner. Using a similar proof, we have that

$$Pr[S_2] \leq (k+1) \cdot \frac{k \cdot d}{|F|}.$$

The only difference is that in this case the degrees of $G_j(t)$ are smaller than the degrees of $F_j(t)$ by a factor of $|H|$.

Therefore, the soundness of the protocol is bounded by $\frac{|H| \cdot 2k^2 \cdot d}{|F|}$. □

We remark that our protocol from Section 4.2.2 has another difference compared to the above protocol. We increase by one the degrees of the curves C_k and D_k. Using a similar argument to the above it can be shown that the soundness of that protocol is bounded by $\frac{(|H|+1) \cdot 2k^2 \cdot d}{|F|}$.

4.2.2 The Protocol, Given a Circuit Specification Oracle

The intuition behind our protocol is as follows. We assume the referee has an oracle access to $\widetilde{spec}_{c'(i)}$. We use the idea of [GKR08] to check the entire computation by checking the sum-checks between each two consecutive layers. We use the protocol of [FK97] to run each sum-check in a single round of communication. Ideally, we would like to execute all the sum-checks in parallel, in a single round. But, we cannot do that directly since the security of the [GKR08] protocol requires that the referee sends the random u_i to the server only *after* the execution of the $(i-1)$-th sum-check. Still, in order to use the protocol of [FK97], the servers have to know u_i since this value determines the function for the sum-check test. Thus, we change the "linking" between the layers.

For simplicity, we now describe the protocol as a sequential protocol with several rounds. However, since we want a one-round protocol, all servers actually execute all rounds of the this protocol *together*, in a single round. The referee chooses his messages for all rounds together and sends them to the servers in one message. Then, the servers answer all rounds together. Last, the referee reads all answers, starting from the input layer towards the output layer, and checks the servers' answers until he finds who is the honest server. (In our protocol the direction of the "linking" reductions is different than in [GKR08].) We denote this protocol by (P_1, P_2, R).

Given an input x, for each layer i the referee chooses two random parametric curves $\gamma_i(t), \varphi_i(t)$ that intersect at point z_i ($\gamma_i(z_i)$ corresponds to the point u_i of [GKR08]). The referee sends $\gamma_i(t), \varphi_i(t)$ to P_1 and P_2, respectively, and asks the servers for the polynomials $\widetilde{V}_i(\gamma_i(t))$ and $\widetilde{V}_i(\varphi_i(t))$. Next, he checks whether those polynomials agree on z_i. If they agree, then he assumes both answers are correct and continues to checking the next layer, $i-1$. Otherwise, he executes a one-round sum-check protocol *a la.* [FK97] to determine the correct value of $\widetilde{V}_i(\gamma_i(z_i))$. (Actually, we use a variation of the protocol of [FK97] which we describe in Section 4.2.1.) But, as we mentioned before, we do not want to explicitly give the servers the value of z_i. Instead, the servers answer the sum-check challenges for *all* the points on $\gamma_i(t)$ and $\varphi_i(t)$, including the value at z_i. Then, the referee uses the referee of the one-round sum-check protocol of [FK97] to determine who is the honest server.

A subtle issue here is how the referee checks the correctness of the sum-checks without being able to compute $f^{(i)}_{z_i}$ by himself. The protocol of [FK97] assumes the referee can compute $f^{(i)}_{z_i}$ by himself for any point, but here, $f^{(i)}_{z_i}$ itself is too complex for the referee to compute. Specifically, in the protocol of [FK97]

the referee needs to compute $f_{z_i}^{(i)}$ on three points: two points that are known only to P_1 and one point that is known only to P_2. In order to solve this problem we again use the *point-on-a-line* technique to get those values "implicitly" from the servers themselves. When the referee believes that the answers on $\widetilde{V}_{i+1}(\gamma_{i+1}(t))$ and $\widetilde{V}_{i+1}(\varphi_{i+1}(t))$ for layer $i+1$ are correct, he takes few random points on those polynomials and uses that to compute the values of $f_{z_i}^{(i)}$ on those three points. The solution requires increasing by one the degrees of the polynomials of the protocol of [FK97] in order to keep the added points secret (see Section 4.2.1).

The detailed protocol (P_1, P_2, R) is presented in Figures 2 and 3. Since some of the polynomials conceal secret intersection points, when the referee sends some polynomial to the servers, we require that he sends the canonical representation of that polynomial.

The referee's running time is $poly(|F|, m, d, |H|) + k \cdot poly(|H|, m) = k \cdot poly(d, log(S))$, the servers running time is $poly(S, |F|, m, d) = poly(S)$ and the communication complexity is $poly(|F|, m, d) = poly(d, log(S))$.

Theorem 4. *Let L be a language in $\mathcal{NC}$ and let C_L be the circuit that decides on L. For any input x and for any constant error probability ε, given a circuit specification oracle for C_L, the protocol $(P_1(x), P_2(x), R(x))$ is ε-RDoC with two servers for the circuit C_L.*

The crucial point of the proof is that a server can cheat with high probability only if he knows the curves' intersection points. Let's see what information each server has about the other server's curves.

Lemma 5. *Let V_1 be the view of P_1 and let i be a round in the protocol. For all $\alpha, \alpha', \beta, \beta', \gamma, \gamma' \in F$ and $j \in [1 \dots 3m]$*

$$\begin{aligned} Pr[z_i = \alpha | V_1] &= Pr[z_i = \alpha' | V_1] && (1)\\ Pr[a_j = \beta | V_1] &= Pr[a_j = \beta' | V_1] && (2)\\ Pr[r = \gamma | V_1] &= Pr[r = \gamma' | V_1]. && (3) \end{aligned}$$

Let V_2 be the view of P_2 and let i be a round in the protocol. For all $\alpha, \alpha', \beta, \beta', \gamma, \gamma', \zeta, \zeta' \in F$ and $j \in [1 \dots 3m]$

$$\begin{aligned} Pr[z_i = \alpha | V_2] &= Pr[z_i = \alpha' | V_2] && (4)\\ Pr[a_j = \beta | V_2] &= Pr[a_j = \beta' | V_2] && (5)\\ Pr[b_j = \gamma | V_2] &= Pr[b_j = \gamma' | V_2] && (6)\\ Pr[r = \zeta | V_2] &= Pr[r = \zeta' | V_2]. && (7) \end{aligned}$$

Proof. The lemma follows from inspecting the protocol.

1. $\varphi_i(t)$ is of degree-2. Even if we give P_1 the exact values of w_1, w_1' (and not only implicitly as part of C_{3m}), there is still one degree of secret information, and therefore $\varphi_i(t)$ can still go through any possible point $(z_i, \gamma_i(z_i))$ for all $z_i \in F$.

2. $D_j(t)$ is of degree at least 1. Even if we give P_1 the value of b_j, he does not have enough information to recover $D_j(t)$, so any $(a_j, C_j(a_j))$ is a possible intersection point.

3. Since P_1 has no information on w_1, w_1' besides from the curve C_{3m}, r is simply a random point on the curve from his point of view.

4. $\gamma_i(t)$ is of degree-4. Even if we give P_2 the exact values of w_0, w_0', w_2, w_2' (and not only implicitly as part of D_{3m}), there is still one degree of secret information, and therefore $\gamma_i(t)$ can still go through any possible point $(z_i, \varphi_i(z_i))$.

Publicly known parameters

H, F, m, d, S, k, δ as in Section 3.

Initialization

For $i = 1, \ldots d$, R randomly picks $z_i \in F$, a random degree-4 parametric curve $\gamma_i(t) \in F[t]^m$, and a random degree-2 parametric curve $\varphi_i(t) \in F[t]^m$ going through $(z_i, \gamma_i(z_i))$.
He also sets $\gamma_0 = \varphi_0 \equiv 0$ and $z_0 = 0$, and computes $M_d(t) = \widetilde{V}_d(\gamma_d(t))$ and $Q_d(t) = \widetilde{V}_d(\varphi_d(t))$.

For $i = d, \ldots, 1$

R's computations :

R sets $w_0 = \gamma_i(0), w_0' = \gamma_i(1), w_1 = \varphi_i(0), w_1' = \varphi_i(1), w_2 = \gamma_i(2), w_2' = \gamma_i(3)$ and randomly chooses $p_0, p_1, p_2 \in F$.

For $1 \le j \le 3m$, R chooses random vectors $A_j, B_j \in F^j$ and random elements $a_j, b_j \in F$. R sets A_{3m} to $p_0 \circ w_0, \circ w_0'$ and chooses a random $r \in F$.

For $1 \le j \le 3m-1$ let $C_j(t) \in F[t]^j$ be the unique degree-$|H|$ parametric curve going through

$$(0, A_{j-1} \circ 0), \ldots, (|H|-1, A_{j-1} \circ (|H|-1)), (|H|, B_j)$$

and let $C_{3m}(t) \in F[t]^{3m}$ be the unique degree-$(|H|+1)$ parametric curve going through

$$(0, A_{3m-1} \circ 0), \ldots, (|H|-1, A_{3m-1} \circ (|H|-1)), (|H|, B_{3m}), (r, p_1 \circ w_1 \circ w_1').$$

For $1 \le j \le 3m-1$, let $D_j(t) \in F[t]^j$ be the unique degree-1 parametric curve going through

$$(a_j, C_j(a_j)), (b_j, A_j)$$

and let $D_{3m}(t) \in F[t]^{3m}$ be the unique degree-2 parametric curve going through

$$(r, p_2 \circ w_2 \circ w_2'), (a_{3m}, C_{3m}(a_{3m})), (b_{3m}, A_{3m}).$$

We define

$$\Phi_{q,j}(x_1, \ldots, x_j) = \sum_{x_{j+1}, \ldots, x_{3m} \in H} f_q^{(i-1)}(x_1, \ldots, x_j, x_{j+1}, \ldots, x_k).$$

R sends to $\mathbf{P}_1$:

$C_j(t), A_j$, for $1 \le j \le 3m$, and the curve $\gamma_{i-1}(t)$.

$\mathbf{P}_1$ sends to R :

For all $q \in F$ define $N_{j,q} = \Phi_{\gamma_{i-1}(q),j}(A_j)$ and $F_{j,q}(t) = \Phi_{\gamma_{i-1}(q),j}(C_j(t))$.

P_1 sends $N_{j,q}, F_{j,q}(t)$ for $1 \le j \le 3m$ and all $q \in F$, and, $M_{i-1}(t)$ where $M_{i-1}(t) = \widetilde{V}_{i-1}(\gamma_{i-1}(t))$.

R sends to $\mathbf{P}_2$:

$D_j(t)$ for $1 \le j \le 3m$, and the curve $\varphi_{i-1}(t)$.

$\mathbf{P}_2$ sends to R :

For all $q \in F$ define $G_{j,q}(t) = \Phi_{\varphi_{i-1}(q),j}(t))$.

P_2 sends $G_{j,q}(t)$ for $1 \le j \le 3m$ and all $q \in F$, and, $Q_{i-1}(t)$ where $Q_{i-1}(t) = \widetilde{V}_{i-1}(\varphi_{i-1}(t))$.

Figure 2: One-round RDoC protocol: initialization and interactive phase

5. $C_j(t)$ is of degree at least $|H|$. Even if we give P_2 the value of b_j, he does not have enough information to recover $C_j(t)$, so any $(a_j, D_j(a_j))$ is a possible intersection point.

Checking layer i for $i = d, \ldots, 1$

P_1 is declared as the cheater if $M_{i-1}(t)$ has degree bigger than $4 \cdot m \cdot (|H| - 1)$. P_2 is declared as the cheater if $Q_{i-1}(t)$ has degree bigger than $2 \cdot m \cdot (|H| - 1)$.

If $M_{i-1}(z_{i-1}) = Q_{i-1}(z_{i-1})$ the referee continues to the proof of layer $i-1$. Otherwise, he continues as follows.

Given $M_i(t)$, R computes:

- $f_{z_{i-1}}^{(i-1)}(A_{i,3m})$ and $f_{z_{i-1}}^{(i-1)}(p_2 \circ w_2 \circ w_2')$ using $M_i(0), M_i(1)$, and $M_i(2), M_i(3)$ (and the oracle).
- $f_{z_{i-1}}^{(i-1)}(p_1 \circ w_1 \circ w_1')$ using $Q_i(0), Q_i(1)$ (and the oracle).

Now, R verifies the sum-check of $M_{i-1}(z_{i-1}) = \sum_{p,w,w' \in H^m} f_{z_{i-1}}^{(i-1)}(p, w, w')$ using the referee from [FK97].

Concretely:

- The referee sets $N_{0,z_{i-1}} = M_{i-1}(z_{i-1})$.
- P_1 is declared as the cheater if: 1) $N_{3m,z_{i-1}} \neq f_{z_{i-1}}^{(i-1)}(A_{3m})$, or 2) $F_{3m,z_{i-1}}(r) \neq f_{z_{i-1}}^{(i-1)}(p_1 \circ w_1 \circ w_1')$, or 3) For some j, $F_{j,z_{i-1}}(t)$ has degree greater than $deg(C_j) \cdot j \cdot 2\delta$ or $\sum_{m=0}^{|H|-1} F_{j,z_{i-1}}(m) \neq N_{j-1,z_{i-1}}$.
- P_2 is declared as the cheater if for some j, $G_{j,z_{i-1}}(t)$ has degree greater than $deg(D_j) \cdot j \cdot 2\delta$, or, $G_{3m,z_{i-1}}(r) \neq f_{z_{i-1}}^{(i-1)}(p_2 \circ w_2 \circ w_2')$.
- Let's denote by j the smallest number such that $G_{j,z_{i-1}}(b_j) = N_{j,z_{i-1}}$. If $G_{j,z_{i-1}}(a_j) = F_{j,z_{i-1}}(a_j)$ then P_1 is declared as the honest, otherwise P_2 is declared as the honest.

Outputting the result

If P_1 was declared as the honest server or P_2 was declared as the cheater, R outputs $M_0(0)$, otherwise he outputs $Q_0(0)$. (Recall that $M_0(0)$ is the claimed result of P_1 and $Q_0(0)$ is the claimed result of P_2.)

Figure 3: One-round RDoC protocol: verification of answers

6. Similar argument as in (5) goes for b_j.
7. Since P_2 has no information on w_2, w_2' besides from the curve D_{3m}, r is simply a random point on the curve from his point of view.

□

Proof of Theorem 4. Using Lemma 5, let's see how much a malicious server can cheat without knowing the intersection points (z_i, a_j, b_j). For a fixed input x and a fixed circuit C, let S_1 be the event that although P_1 is the malicious server and the referee outputs a wrong result (i.e., $M_0(0)$ that is not equal to $C(x)$). Let T_i be the event that $M_i(t)$ is indeed $\widetilde{V}_i(\gamma_i(t))$, and let E_i be the event that $M_i(z_i)$ is indeed $\widetilde{V}_i(\gamma_i(z_i))$. Then,

$$Pr[S_1] \leq Pr[\neg T_0 \wedge T_{d-1}] \leq Pr[\exists i \in [d-1] \; s.t. \; \neg T_i \wedge T_{i+1}] \leq \sum_{i=0}^{d-1} Pr[\neg T_i \wedge T_{i+1}].$$

For every $i \in [d-1]$,

$$Pr[\neg T_i \wedge T_{i+1}] = Pr[\neg T_i \wedge T_{i+1} \wedge E_i] + Pr[\neg T_i \wedge T_{i+1} \wedge \neg E_i].$$

By the soundness property of the protocol from [FK97] (see Section 4.2.1), we have that

$$Pr[\neg T_i \wedge T_{i+1} \wedge \neg E_i] \leq Pr[T_{i+1} \wedge \neg E_i] \leq \frac{(|H|+1) \cdot 2(3m)^2 \cdot 2\delta}{|F|} = \frac{(|H|+1) \cdot 36m^2 \cdot \delta}{|F|}.$$

By the fact that two distinct univariate degree-t polynomials agree on at most t points we get that

$$Pr[\neg T_i \wedge T_{i+1} \wedge E_i] \leq Pr[\neg T_i \wedge E_i] \leq \frac{4m \cdot (|H| - 1)}{|F|}.$$

Therefore, we get that (assuming $m > 4$)

$$Pr[\neg T_i \wedge T_{i+1}] \leq \frac{(|H| + 1) \cdot 36m^2 \cdot \delta}{|F|} + \frac{4m \cdot (|H| - 1)}{|F|} \leq \frac{(|H| + 1) \cdot 37m^2 \cdot \delta}{|F|}.$$

Thus, summing the error probabilities for all layers, we get

$$Pr[S_1] \leq d \cdot \frac{(|H| + 1) \cdot 37m^2 \cdot \delta}{|F|}.$$

Let S_2 be the event that although P_2 is the malicious server the referee outputs a wrong result (i.e., $Q_0(0)$ that is not equal to $C(x)$). Using a similar proof, we have that $Pr[S_2]$ is also bounded by the same probability. The only difference is that in this case the degrees of $Q_i(t)$ are smaller than the degrees of $M_i(t)$ by a factor of 2.

Thus, for any constant soundness ε we can take F to be of size $\geq \frac{(|H|+1)\cdot 37m^2\cdot\delta}{\varepsilon}$ which is $poly(|H|)$. $\square$

4.2.3 Realizing the Oracle for $\mathcal{L}$-uniform $\mathcal{NC}$ Circuits

For any language $L \in \mathcal{L}$-uniform $\mathcal{NC}$ there exists a circuit C_L of poly-size and polylogarithmic-depth that computes L. Furthermore, the polynomials $\widetilde{spec}_{c(i)}$ of C_L can be computed by a log-space TM, which means that $\widetilde{spec}_{c(i)}$ can be computed by an $\mathcal{NL}$ circuit, $C_{spec(L)}$. As shown in [GKR08], the circuit specification function $\widetilde{spec}_{c(i)}$ of circuits in $\mathcal{NL}$ can be computed in poly-logarithmic time. This means that the referee can compute $\widetilde{spec}_{c(i)}$ of $C_{spec(L)}$ by himself, and execute the protocol from Section 4.2.2 without an oracle assistance.

Recall the idea of [GKR08] for extending the *bare-bones* protocol to $\mathcal{L}$-uniform $\mathcal{NC}$ circuits. In order to verify the computation of the circuit C_L, the client runs the bare-bones protocol for verifying C_L, and asks the server for the required values of the circuit specification function $\widetilde{spec}_{c(i)}$ (i.e. the server acts as the oracle). Then, the client checks each of those claimed values by executing the bare-bones protocol for the circuit $C_{spec(L)}$ (for which he can compute the oracle answers by himself).

Now, if we try to follow this idea for extending the protocol from Section 4.2.2 to work with $\mathcal{L}$-uniform $\mathcal{NC}$ circuits, and try to run in parallel the protocol also for verification of $C_{spec(L)}$, we get contradicting requirements. On the one hand, for verification of $C_{spec(L)}$, both servers have to know p_j, w_j, w'_j for $j = 0, 1, 2$ as those are the *inputs* for the specification circuit (and the protocol assumes those inputs are known to both servers), but on the other hand, for verification of C_L, the soundness of the protocol requires that those values will not be known to both servers.

In order to tackle this problem, we use a similar idea to the one used in the previous protocol. The referee asks the servers to answer on many points, without revealing the actual p_j, w_j, w'_j. Note that each one of p_j, w_j, w'_j is at least "implicitly" known to both servers. E.g., p_0, w_0, w'_0 is implicitly known to P_2 from $D_{3k}(t)$ (and is explicitly known to P_1 by A_{3m}). Also, we can explicitly send the values of p_0 and p_2 to P_1, and the values of p_1 to P_2, without ruining the soundness of the previous protocol.

Using those two observations, we construct a protocol (P'_1, P'_2, R') for any language in $\mathcal{L}$-uniform $\mathcal{NC}$. For verification of the output of C_L, the referee executes the protocol from Section 4.2.2 with two modifications: 1) The referee sends to P'_1 also the values of p_0, p_2 and to P'_2 also the values of p_1 for all layers, and, 2) P'_1 sends the (claimed) values of $\widetilde{spec}_{c(i)}(b, p_0, w_0, w'_0)$ and $\widetilde{spec}_{c(i)}(b, p_2, w_2, w'_2)$ for all layers, and P'_2 sends the (claimed) values of $\widetilde{spec}_{c(i)}(b, p_1, w_1, w'_1)$ for all layers.

For each of the answers $\widetilde{spec}_{c(i)}(b, p_j, w_j, w'_j)$, the referee executes the protocol from Section 4.2.2 for verification of those claimed values using the circuit $C_{spec(L)}$ (for which he can compute the circuit specification by himself). As we mentioned before, neither of p_j, w_j, w'_j (for $j = 0, 1, 2$) is explicitly known to both servers. So instead we ask one of the servers to answer on many points instead of the specific p_j, w_j, w'_j. Specifically, for verification of $\widetilde{spec}_{c(i)}(b, p_0, w_0, w'_0)$ of layer i of C_L, we execute the protocol from Section 4.2.2, where P'_1 plays the role of P_1 and knows p_0, w_0, w'_0, and P'_2 plays the role of P_2 for *all the points* on the curve $D_{3m}(t)$ as the possible inputs for $C_{spec(L)}$. (There are at most $|F|$ points on this curve). This means that P'_2 does not know the specific p_0, w_0, w'_0. However, since $D_{3m}(t)$ passes through p_0, w_0, w'_0, one of those answers will be the needed P'_2's answer for the input p_0, w_0, w'_0. Similarly, the same roles go for p_2, w_2, w'_2 (which is also included in $D_{3m}(t)$). For p_1, w_1, w'_1 for all layers of C_L, P'_2 plays the role of P_1 in the protocol of Section 4.2.2 and P'_1 plays the role of P_2 for *all the points* on the curve $C_{3m}(t)$ (which includes p_1, w_1, w'_1).

When the referee receives the messages from both servers (for verification of $C_{spec(L)}$ and of C_L), he checks if they agree on all the values of $\widetilde{spec}_{c(i)}(b, p_j, w_j, w'_j)$. If they disagree on some of the values, then the referee checks one of those disagreements using the referee R from Section 4.2.2 and outputs according to its answer. If the servers agree on all the values of $\widetilde{spec}_{c(i)}$, then by the assumption that one of them is honest, those values are correct. Then, the referee verifies the computation of the circuit C_L given the values of $\widetilde{spec}_{c(i)}$ he got before. He runs the checking phase of the referee from Section 4.2.2 and outputs according to its answer.

The overhead of this solution is only polynomial in all parameters. For each layer we have three invocations of the protocol form Section 4.2.2 where one of the servers executes the protocol for $|F|$ points. Summing over all layers, the total overhead is $\leq 3 \cdot d \cdot |F| \cdot depth(C_{spec(L)})$ which is still poly-logarithmic in the size of the input.

Theorem 6. *Let L be a language in $\mathcal{L}$-uniform $\mathcal{NC}$. For any input x and for any constant error probability ε, the protocol $(P'_1(x), P'_2(x), R'(x))$ is ε-RDoC with two servers for the circuit C_L.*

Proof. Note that the information that the referee sends for the verification of $C_{spec(L)}$ is independent of the messages for the verification of C_L. Those proofs share only one piece of information, the values of p_j, w_j, w'_j as the inputs for the circuit $C_{spec(L)}$.

Let's assume P'_1 is the cheater. He can cheat either on some value of $\widetilde{spec}_{c(i)}$ or on the computation of C_L. In the first case, he will be caught with high probability by the soundness of the protocol from Section 4.2.2. For the second case, if P'_1 cheats on the computation of C_L (while the values of $\widetilde{spec}_{c(i)}$ are correct), then it means he can cheat in the protocol from Section 4.2.2 in the case where he has an oracle access to $\widetilde{spec}_{c(i)}$.

By a union bound of the cheating probabilities of the $3d + 1$ invocations of the protocol, we can bound the probability of cheating by $(3d + 1) \cdot d \cdot \frac{(|H|+1) \cdot 37m^2 \cdot \delta}{|F|}$. Thus, for any constant soundness ε we can take F to be of size $\geq (3d + 1) \cdot d \cdot \frac{(|H|+1) \cdot 37m^2 \cdot \delta}{\varepsilon}$ which is $poly(|H|)$.

□

5 Publicly Verifiable Delegation of Computation (PVDoC)

Previous constructions and definitions (e.g., [GGP10, CKV10, AIK10, Chu11, Ben11]) allow the client to work longer in the offline stage, and compute some secret key which he could later use in the online stage. This assumes that at some point in time, the client *can* work harder or has access to a trusted third-party. Furthermore, the server must not learn any information about this key, so if the client uses the assistance of a trusted third-party he has to get a unique secret key, for his use only.

We present a natural extension of this model, where instead of generating a secret key in the offline stage, the computing party outputs a public key that can be used by anyone. In addition, any (powerful

enough) player can verify that key. An example for a real-world scenario is the following:

- Google publishes a (singed) set of public keys that corresponds to a set of functions.
- Google's competitors (or anyone in that matter) can verify these keys and in particular publish an accusation proof in case some of the keys are invalid.
- A client can delegate his computation to *any* server, using the published public keys. He does not have to run the offline stage by himself.
- In case the proof of the server is invalid, the client can publish the transcript and its own coins, and prove that the server is a cheater. We stress that by publishing his coins, the client does not loss privacy of any other key (as happens with some of the previous constructions).

5.1 The Model

A Publicly Verifiable Delegation of Computation (PVDoC) scheme consists of offline and online stages. The offline stage is executed only once before the online stage whereas the online stage can be executed many times. The algorithms are as following:

- $\mathsf{KeyGen}(F, \lambda) \rightarrow PK$: Based on the security parameter λ, the deterministic key generation algorithm generates a public key that encodes the function F, which is used by a client to verify delegations of F.

  Let $T(F)$ be the time bound required to compute F on any input. We require that the running time of the algorithm will be $\leq poly(T(F), \lambda)$.

- $\mathsf{ProbGen}(x, PK, \lambda) \rightarrow (k_x, c_x)$: The problem generation algorithm uses the public key and the input x to generate a challenge c_x that is given to the server, and a secret key k_x that is kept private by the client.

- $\mathsf{Compute}(x, PK, c_x) \rightarrow (y, \pi_y)$: Using the public key and the input, the server computes the function's output $y = F(x)$ along with a proof of correctness π_y given the challenge c_x.

  We require that the running time of the algorithm will be $\leq poly(T(F), \lambda)$.

- $\mathsf{VerifyResult}(PK, k_x, y, \pi_y) \rightarrow y \cup \perp$: Using the secret key k_x, the verification algorithm verifies the server's proof π_y and if successed outputs y. Otherwise it outputs $\perp$.

  We require that the sum of the running times of this algorithm and $\mathsf{ProbGen}()$ for the same input, will be $o(T(F), \lambda)$.

Note that since $\mathsf{KeyGen}()$ is deterministic, anyone can verify (in $poly(T(F), \lambda)$ time) whether PK is a valid encoding of the target function F.

As for completeness and soundness, we require the following:

Definition 7 (ε-secure PVDoC)**.** *We say that a scheme is a ε-secure PVDoC for a circuit C if after the offline stage (and given a valid PK) the following holds for any input x and $(k_x, c_x) = \mathsf{ProbGen}(x, PK, \lambda)$:*

- *(Completeness) If $(y, \pi_y) = \mathsf{Compute}(x, PK, c_x)$ then $\mathsf{VerifyResult}(PK, k_x, y, \pi_y) = y$.*
- *(Soundness) For any $y^* \neq C(x)$ and π_{y^*}, $Pr[\mathsf{VerifyResult}(PK, k_x, y^*, \pi_{y^*}) = \perp] \geq 1 - \varepsilon$.*

5.2 One-round PVDoC for More Than $\mathcal{L}$-uniform $\mathcal{NC}$

Given a circuit specification oracle, the bare-bones protocol from Section 3.2 requires the client to run in time proportional to $max(d, \log(S))$. However, it remains to see how to realize the oracle. As discussed above, [GKR08] shows a way to realize the oracle for $\mathcal{L}$-uniform $\mathcal{NC}$ languages, and using the transformation of [KR09] one can get a one-round computational sound PVDoC protocol for these languages.

If we are interested in a larger class of languages, we can use the next technique. Given a circuit C, in the offline stage, the client (or some other third party) computes the polynomials $\widetilde{spec}_{c(i)}$ and the evaluation of these polynomials on all their domains. Then, the client computes the root of the Merkle Hash Tree on these values (i.e., the tree leaves are the values of the polynomials).

Later on, in the online stage, the client and the server run the bare-bones protocol under cPIR queries. However, instead of delegating also the computation of the circuit specification predicate, the client asks this value from the server, along with a proof of consistency with the root of the Merkle Hash Tree.

A useful property of this protocol is that the information computed in the offline stage can be used by any interested client, and, as such, the protocol is secure also against adaptive adversaries. It is publicly verifiable as opposed to previous solutions in the offline/online model.

Figure 4 presents the detailed protocol.

Theorem 8. *For any circuit C and for any constant error probability ε, the protocol from Figure 4 is ε-secure PVDoC for the circuit C. In the online stage, the client runs in $poly(depth(C), log(size(C)))$ time and the server in $poly(size(C))$ time.*

Proof. Completeness follows from inspecting the protocol.

As for soundness, we will look on the protocol as a composition of two different protocols. The first is the protocol of GKR-KR where the verification of $C(x)$ is reduced to the correctness of a random point on the low degree extension of the input x, given a circuit specification oracle. We denote by s_1 the soundness of GKR-KR (can be arbitrarily small, see [GKR08, KR09]). The second protocol is the cPIR queries on a database which includes the circuit specification truth table augmented with proofs of consistency. The soundness of this protocol s_2 is negligible from the collision resistancy of the hash function

Since we hide the queries using cPIR queries, the requested b^i, w_1^i, w_2^i, w_3^i in the second protocol are computationally indistinguishable, and therefore, the server in the first protocol does not get useful information about it (otherwise, we can break the security of the cPIR).

We claim that the soundness of the composition is bounded by $s_1 + s_2 + neg(\lambda)$. Suppose there is an adversary A that breaks the protocol with probability $p \geq s_1 + s_2 + neg(\lambda) + c$ where $c > 0$ is a constant. In order to cheat, the adversary has to cheat (at least) in either the first or the second protocol. Let A_1 be an adversary for the GKR-KR protocol that simulates the second protocol and executes A on both the GKR-KR messages and the simulated ones and let p_1 be the probability it cheats. Similarly, let A_2 be an adversary for the second protocol where it simulates the GKR-KR messages, and let p_2 be its cheating probability. Since $Pr[A \text{ cheats}] \leq Pr[A_1 \text{ cheats}] + Pr[A_2 \text{ cheats}]$ we get that $p_1 + p_2 \geq p \geq s_1 + s_2 + neg(\lambda) + c$. Hence, one of p_i is at least $s_i + c/2 + neg(\lambda)$, which contradicts the assumption about the soundness of the original protocols.

By carefully picking the parameters of the GKR-KR protocol and the hash function we use, it is possible to get any constant soundness.

□

By parallel repetition the error probability can be reduced exponentially (using the results of Bellare *et al.* [BIN97] and Canetti *et al.* [CHS05]).

KeyGen(F, λ)**:**
This algorithm is called in the offline stage. Let C be the circuit that computes the function F, and let h be a collision resistant hash function given the security parameter λ. Also, define the GKR-KR protocol parameters as in Section 3.2.

- Compute the polynomials $\widetilde{spec}_{c(i)}$ and the values of $\widetilde{spec}_{c(i)}(b, w_1, w_2, w_3)$ for all $b \in \{0, 1\}$ and $w_1, w_2, w_3 \in F^m$.
- Construct a Merkle Hash Tree where the leaves of the tree are the values $h(\widetilde{spec}_{c(i)}(b, w_1, w_2, w_3))$.
- Return $PK = [ID_F, size(C), depth(C), \lambda, h, root)]$ where ID_F is a short string that identifies the function F, and $root$ is the root of the merkle hash tree.

Note that any player can compute all the above values. In particular, once PK is published, other players can verify it. However, this computation requires polynomial time (in the size of C).

ProbGen(x, PK)**:**
When the client wants to delegate a computation (in the online stage), he computes the following:

- Let $m_1 =$ GKR-KR$_c(size(C), depth(C), \lambda)$ be the first message sent by the client in the GKR-KR protocol for the bare-bones protocol only (i.e., for verification of the computation of C only), and let $(b^{(i)}, w_1^{(i)}, w_2^{(i)}, w_3^{(i)})$ (for $i = 0 \dots depth(C)$) be the quadruplets that the protocol queries for their $\widetilde{spec}_{c(i)}$ values.
- Compute cPIR queries for each quadruplet $(b^{(i)}, w_1^{(i)}, w_2^{(i)}, w_3^{(i)})$. Denote the resulting set of queries by m_2.
- Let k_x be the random coins used for the computation of GKR-KR$_c$ and the cPIR queries. Return $c_x = [m_1, m_2]$ and k_x.

Note that the above steps do not depend on the input x.

Compute(x, PK, c_x)**:**
The server receives the input x, the public key and the challenge, and does the following:

- Evaluate $y = C(x)$.
- Prepare two databases. The first is a database with the answers to the GKR-KR protocol, and the second is a database that includes the values of $\widetilde{spec}_{c(i)}(b, w_1, w_2, w_3)$ augmented with the Merkle Hash Tree proofs of consistency. I.e., the database consists of $2|F|^{3m} = poly(size(C))$ entries, where in the (a, b, c, d) entry there is the value of $\widetilde{spec}_{c(i)}(a, b, c, d)$ and the path in Merkle Hash Tree from $h(\widetilde{spec}_{c(i)}(a, b, c, d))$ to the root.
- Let $(m_1, m_2) = c_x$. Compute $m_1' =$ GKR-KR$_p(C, x, m_1)$ (i.e. the answers according to the GKR-KR protocol and the first database).
- Compute the cPIR answers for the queries m_2 and the second database. Denote the results by m_2'.
- Return $y, [m_1', m_2']$.

VerifyResult(PK, k_x, y, π_y)**:**
Let $(m_1', m_2') = \pi_y$. Using the secret key k_x, the client does the following:

- Verify that the answers in m_2' are consistent with the root from PK.
- Run the verifier of the GKR-KR protocol on m_1' where each time a value of $\widetilde{spec}_{c(i)}$ is needed, use the answers from m_2'.

If both checks successed, output y. Otherwise, output $\perp$.

Figure 4: One-round PVDoC protocol

References

[AIK10] Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz, *From secrecy to soundness: efficient verification via secure computation*, Proceedings of the 37th international colloquium conference on Automata, languages and programming, Springer-Verlag, 2010, pp. 152–163.

[BCCT11] Nir Bitansky, Ran Canetti, Alessandro Chiesa, and Eran Tromer, *From extractable collision resistance to succinct non-interactive arguments of knowledge, and back again*, Cryptology ePrint Archive, Report 2011/443, 2011, `http://eprint.iacr.org/`.

[Ben11] *Verifiable delegation of computation over large datasets*, CRYPTO, vol. 6841, 2011, p. 110.

[BFLS91] László Babai, Lance Fortnow, Leonid A. Levin, and Mario Szegedy, *Checking computations in polylogarithmic time*, Proceedings of the twenty-third annual ACM symposium on Theory of computing, ACM, 1991, pp. 21–32.

[BIN97] Mihir Bellare, Russell Impagliazzo, and Moni Naor, *Does parallel repetition lower the error in computationally sound protocols?*, Proceedings of the 38th Annual Symposium on Foundations of Computer Science, IEEE Computer Society, 1997, pp. 374–383.

[BOGKW88] Michael Ben-Or, Shafi Goldwasser, Joe Kilian, and Avi Wigderson, *Multi-prover interactive proofs: how to remove intractability assumptions*, Proceedings of the twentieth annual ACM symposium on Theory of computing, ACM, 1988, pp. 113–131.

[CHS05] Ran Canetti, Shai Halevi, and Michael Steiner, *Hardness amplification of weakly verifiable puzzles*, Proceedings of the second Theory of Cryptography Conference, Springer-Verlag, 2005, pp. 17–33.

[Chu11] *Memory delegation*, CRYPTO, vol. 6841, 2011, p. 147.

[CKV10] Kai Min Chung, Yael Kalai, and Salil Vadhan, *Improved delegation of computation using fully homomorphic encryption*, Proceedings of the 30th annual conference on Advances in cryptology, Springer-Verlag, 2010, pp. 483–501.

[CL08] Giovanni Crescenzo and Helger Lipmaa, *Succinct NP proofs from an extractability assumption*, Proceedings of the 4th conference on Computability in Europe: Logic and Theory of Algorithms, Springer-Verlag, 2008, pp. 175–185.

[CRR11a] Ran Canetti, Ben Riva, and Guy N. Rothblum, *QUIN: Practical delegation of computation using multiple clouds (to appear)*, Proceedings of the 18th ACM Conference on Computer and Communications Security, 2011.

[CRR11b] Ran Canetti, Ben Riva, and Guy N. Rothblum, *Refereed delegation of computation*, 2011, `http://www.cs.tau.ac.il/~canetti/CRR11.pdf`.

[CRR11c] Ran Canetti, Ben Riva, and Guy N. Rothblum, *Verifiable computation with two or more clouds*, Workshop on Cryptography and Security in Clouds, 2011, `http://www.zurich.ibm.com/~cca/csc2011/submissions/riva.pdf`.

[FK97] Uriel Feige and Joe Kilian, *Making games short (extended abstract)*, Proceedings of the twenty-ninth annual ACM symposium on Theory of computing, ACM, 1997, pp. 506–516.

[GGP10] Rosario Gennaro, Craig Gentry, and Bryan Parno, *Non-interactive verifiable computing: outsourcing computation to untrusted workers*, Proceedings of the 30th annual conference on Advances in cryptology, Springer-Verlag, 2010, pp. 465–482.

[GKR08] Shafi Goldwasser, Yael Tauman Kalai, and Guy N. Rothblum, *Delegating computation: interactive proofs for muggles*, Proceedings of the 40th annual ACM symposium on Theory of computing, ACM, 2008, pp. 113–122.

[GLR11] Shafi Goldwasser, Huijia Lin, and Aviad Rubinstein, *Delegation of computation without rejection problem from designated verifier cs-proofs*, Cryptology ePrint Archive, Report 2011/456, 2011, `http://eprint.iacr.org/`.

[Kil92] Joe Kilian, *A note on efficient zero-knowledge proofs and arguments (extended abstract)*, Proceedings of the twenty-fourth annual ACM symposium on Theory of computing, ACM, 1992, pp. 723–732.

[KR09] Yael Tauman Kalai and Ran Raz, *Probabilistically checkable arguments*, Proceedings of the 29th Annual International Cryptology Conference on Advances in Cryptology, Springer-Verlag, 2009, pp. 143–159.

[KR11] Gillat Kol and Ran Raz, *Competing provers protocols for circuit evaluation*, Tech. Report TR11-122, Electronic Colloquium on Computational Complexity, September 14 2011, `http://eccc.hpi-web.de/`.

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

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

[Rot09] Guy N. Rothblum, *Delegating computation reliably: paradigms and constructions*, Ph.D. Thesis, Massachusetts Institute of Technology, 2009.