Two Protocols for Delegation of Computation

Ran Canetti[1,2⋆], Ben Riva[2⋆], and Guy N. Rothblum[3]

[1] Boston University, Boston, MA, USA
[2] Tel Aviv University, Tel Aviv, Israel
[3] Microsoft Research Silicon Valley, Mountain View, CA, USA

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. We present protocols in two relaxed variants of this problem. We first 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. In this model, we show a 1-round statistically sound protocol for any log-space uniform $\mathcal{NC}$ circuit. In contrast, in the single server setting all known one-round succinct delegation protocols are computationally sound. The protocol extends the arithemetization techniques of [Goldwasser-Kalai-Rothblum, STOC 08] and [Feige-Kilian, STOC 97].
Next we consider a simplified view of the protocol of [Goldwasser-Kalai-Rothblum, STOC 08] in the single-server model with a non-succinct, but *public*, offline stage. Using this simplification we construct two computationally sound protocols for delegation of computation of any circuit C with depth d and input length n, *even a non-uniform one*, such that the client runs in time $n \cdot \text{poly}(\log(|C|), d)$. The first protocol is potentially practical and easier to implement for general computations than the full protocol of [Goldwasser-Kalai-Rothblum, STOC 08], and the second is a 1-round protocol with similar complexity, but less efficient server.

1 Introduction

An emerging paradigm in modern computing is *pay-per-use* Cloud Computing, where individuals and companies outsource computations to companies that perform the computations on dedicated servers. This motivates exploring methods for delegating computations reliably, while maintaining the efficiency gains: A weak client delegates his computation to a powerful server; once the server returns the result of the computation, the client should be able to verify the correctness of that result so that (a) the client uses considerably less resources than those required to actually perform the computation from scratch, and (b) the server does not use significantly more resources than those needed for performing the computation.

A classic solution to this problem is to have the server prove correctness of the computation using a universal argument [K92,BG02]. Here the server incurs

⋆ Research supported by the Check Point Institute for Information Security, an ISF grant and an EU Marie Curie grant.

polynomial overhead and the client's complexity is linear in the input length and security parameter, and only polylogarithmic in the complexity of the original computation. This solution is general and applies to any problem in $\mathcal{P}$. It also has the nice property that it is publicly verifiable: The client needs to keep no secret state. However, it has two main drawbacks: First, it is interactive, taking four messages. Second, it is only computationally sound.

Micali [M00] "squashes" Kilian's protocol to one message in the Random Oracle model, while maintaining public verifiability. Further extensions of this construction give a two-message protocol in the plain model, with similar complexity advantages [CL08,BCCR12,GLR11,DFH12]. However, their soundness relies on non-standard hardness assumptions. Furthermore, all of these protocols have only computational soundness. No statistically-sound protocol with a constant number of rounds is known.

An alternative delegation protocol, proposed by Goldwasser, Kalai and Rothblum [GKR08], guarantees statistical soundness. However, the protocol works only for languages that have $\mathcal{L}$-uniform $\mathcal{NC}$ circuits. Furthermore, the number of rounds is quasilinear in the depth of the circuit. 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 polylogarithmic communication. Here, however, soundness is only computational.

Two relaxations of the model have been recently proposed. One relaxation, proposed in [CRR11], lets the client interact with *two or more* servers, and requires 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 the servers 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 before, the servers should incur only polynomial overhead, whereas the client's resources should be much smaller than required to compute the function. This model, called the *Refereed Delegation of Computation (RDoC)* model, directly extends the refereed games model of Feige and Kilian [FK97]. It is also somewhat reminiscent of the MIP model of [BGKW88]. However, the MIP model provides different guarantees (e.g., soundness is guaranteed only as long as no two servers collude, and malicious servers can potentially prevent honest ones from convincing the client.) In that model, and assuming collisions resistant hash functions, [CRR11] shows a computationally sound delegation protocol whose number of rounds is logarithmic in the time needed to compute $f(x)$. The protocol is inspired by a protocol of Feige and Kilian [FK97].

Another relaxation, proposed 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 online stage, the client can delegate the computation of $f(x)$ for any input x, and can verify the result in time that independent of the size of f. The assumption that the client can work "hard" during the offline stage is often justified by

proposing that the client rely on the assistance of a trusted party to perform the offline stage on the client's behalf. We call this model the *Offline/Online Verifiable Delegation of Computation (OVDoC)* model. Several protocols have been proposed in this model [GGP10,CKV10,AIK10,CKLR11,BGV11].

However, in all the recent protocols in this model the offline stage generates information that must be kept secret and used at the online stage (or else soundness will no longer be guaranteed). 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.

1.1 Our Contributions

A statistically sound, 1-roundtrip protocol in the RDoC model. Our main contribution is a *1-roundtrip statistically sound* RDoC protocol for any circuit computable in $\mathcal{L}$-uniform$\mathcal{NC}$. The protocol adapts techniques from a protocol of Feige and Kilian [FK97], where the servers are inefficient even for log-space computations, together with 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.

At a high level, our protocol follows the structure of the [GKR08] interactive proof. That is, initially the servers make claims about the output layer of the circuits. Then, a (very efficient) *sum-check* protocol [LFKN92] is used 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, as in [GKR08], the above 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.

Kol and Raz present an alternative exposition of this protocol [KR11]. They also provide 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. As far as we know, these results have been obtained independently of ours.

A simplification of the [GKR08] protocol in the OVDoC model with a public offline stage. For our second result, we start by describing the model of OVDoC with a public offline stage. Recall that the offline/online delegation of computation model assumes that the client can work "hard" during the offline stage and that the information generated in the offline stage is kept secret. In our model the entire randomness used in the offline stage is made public. This means that: (1) The correctness of the offline computation can be verified by anyone, and, (2) The output of the offline stage can be used by anyone in the online stage to verify the computation of any input.

For instance, 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 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.

As already mentioned, the [GKR08] protocol works only for $\mathcal{L}$-uniform $\mathcal{NC}$ circuits. This is because the client cannot verify claims about the circuit structure for larger circuits since the explicit circuit is too large (larger than the client running time) and (for general non-uniform circuits) there is no shorter implicit representation.[4] Moreover, the full [GKR08] protocol has a large overhead over the plain execution of the computation.

In the relaxed model of OVDoC with a public offline stage we show that we can combine a lighter version of the [GKR08] protocol (namely, only the *bare-bones* protocol. See Sect. 2) with another efficient computationally-sound

[4] We note that Cormode, Mitzenmacher and Thaler [CMT12] argue that many useful problems have succinct circuit representation. In those cases a weak client can verify claims about the circuit structure by herself.

protocol in a way that bypasses the above difficulties. As a result, our protocol can work with *any* circuit C, requiring the client to run in time $n \cdot \mathrm{poly}(\log(|C|), \mathrm{depth}(C))$, where n is the input length. We do not require the circuit to be uniform, resulting in a potentially easier to implement protocol. Furthermore, the simplification reduces the number of executions of the bare-bones protocol by a factor of $O(\mathrm{depth}(C))$, which in practice can be very meaningful. (See [CMT12] for experimental results of a single execution of the bare-bones protocol. Note that they also significantly improve the server's running time using a technique that can be applied to our protocols as well.)

As a more theoretical result, we additionally show that by carefully applying the technique of [KR09] we can construct a 1-round protocol with less efficient server. Previously, assuming standard computational assumptions, it was known how to construct a 1-round protocol only for $\mathcal{L}\text{-uniform}\,\mathcal{NC}$ circuits.

1.2 Organization

Section 2 describes the protocol of [GKR08] which we use extensively in our constructions. Section 3 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. It also presents the construction of a one-round RDoC for any $\mathcal{L}\text{-uniform}\,\mathcal{NC}$ computation. Section 4 describes the simplified construction of the [GKR08] protocol and the construction of a one-round computationally sound protocol for any circuit.

2 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 these protocols. Also, in Appendix A we describe the protocol of [FK97] that we use in our RDoC construction.

2.1 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 $\widetilde{f}$, be the *unique* multi-variate polynomial $\widetilde{f} : F^m \to F$ that satisfies: (1) (low-degree) $\deg(\widetilde{f}) < |H|$ for each variable; and (2) (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 $\boldsymbol{w} = (w_0, \ldots w_{k-1}) \in F^k$, where $k \leq |H|^m$, we can view this vector as a function $f_{\boldsymbol{w}} : H^m \to F$ such that $f_{\boldsymbol{w}}(z) = w_{\alpha(z)}$ when $\alpha(z) \leq k-1$ and $f_{\boldsymbol{w}}(z) = 0$ otherwise. We define the low degree extension of the vector $\boldsymbol{w}$ to be the low degree extension of $f_{\boldsymbol{w}}$.

2.2 The Bare-Bones [GKR08] 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 n.

Given an arithmetic circuit $C : \{0,1\}^n \to \{0,1\}$ of fan-in 2 gates, size S and depth d, the players choose the following parameters: 1) An extension field H of $\mathsf{GF}[2]$ such that $\max(d, \log(S)) \leq |H| \leq \text{poly}(d, \log(S))$, 2) An integer m such that $S \leq |H|^m \leq \text{poly}(S)$, 3) An extension field F of H such that $|F| \leq \text{poly}(|H|)$. (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 $C' : F^n \to F^S$ over the field F with the following properties: 1) C' is of size $\text{poly}(S)$ 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 $(x_1, \ldots, x_n) \in \{0,1\}^n$, $C'(x_1, \ldots, x_n) = (C(x_1, \ldots, x_n), 0, \ldots, 0)$.

Let $\mathsf{spec}_{\mathrm{c}}()$ be the predicate describing a circuit C. That is, $\mathsf{spec}_{\mathrm{c}}(i, b, w_1, w_2, w_3)$ 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 $b \in \{0 = add, 1 = mult\}$.[5] Let $\widetilde{\mathsf{spec}}_{\mathrm{c}(i)}$ be the low degree extension of $\mathsf{spec}_{\mathrm{c}}(i, \cdot, \cdot, \cdot, \cdot)$ with respect to H, F and m, of degree δ (that depends on $\mathsf{spec}_{\mathrm{c}}()$) in each variable. In this section we assume the client has an oracle access to $\widetilde{\mathsf{spec}}_{\mathrm{c}'(i)}$.

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 $C'(x_1, \ldots, x_n)$. v_0 is the circuit result $(C(x_1, \ldots, x_n), 0, \ldots, 0)$ and v_d is the circuit input $(x_1, \ldots, x_n)$. Let $\widetilde{V}_i : F^m \to F$ 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 $\leq \text{poly}(|F|^m) = \text{poly}(S)$. Since v_d is of length n, $\widetilde{V}_d$ can be computed in time $\leq n \cdot \text{poly}(|H|, m)$.

The Protocol. The server claims that $C'(x_1, \ldots, x_n) = (0, \ldots, 0)$. 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. [GKR08,R09] show that there exists a $3m$-variate polynomial $f_u^{(i)} : (F^m)^3 \to F$ of size $\leq$

[5] In general, it is possible to use any gate that can be expressed as a polynomial of its inputs.

poly(S) and degree $\leq 2\delta$ defined by

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

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$(|H|,m)$. Note that given an oracle access to $\widetilde{\mathsf{spec}}_{c'(i)}$ 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$(|H|,m)$.

[GKR08,R09] proves that $\widetilde{V}_i(u_i) = \sum_{p,w,w' \in H^m} f_{u_i}^{(i)}(p,w,w')$. 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, 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).

Complexity. It is shown in [R09] that by taking F such that $|F| \geq 700md\delta = \text{poly}(|H|)$ 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$(|F|,m) + n \cdot$ poly$(|H|,m) = n \cdot$ poly$(d, \log(S))$ and the communication complexity is poly$(|F|,m) =$ poly$(d, \log(S))$.

Cormode, Mitzenmacher and Thaler [CMT12] show that by using a multi-linear extension, the running time of the server can be reduced to $O(S \cdot \log(S))$. Their technique can be applied to our constructions as well. They also suggest that by using large fields, e.g. of size 32 bit, performance of many applications can be significantly improved. However, in both of our constructions the players work in time $\geq |F|^m$, thus making this option inefficient.

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

The above protocol is presented for any circuit given an oracle access to $\widetilde{\mathsf{spec}}_{c'(i)}$. [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 $\widetilde{\mathsf{spec}}_{\mathrm{c}(i)}$ (where C is the circuit that computes L) in time polylog(n). Thus, the client can compute it by himself and, as a result, realize the bare-bones protocol.
2. For a language L in $\mathcal{L}$-$\mathsf{uniform}\,\mathcal{NC}$, 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 $\mathsf{TM}_{\mathsf{spec}(\mathrm{c})}$ be the Turing Machine that computes $\widetilde{\mathsf{spec}}_{\mathrm{c}(i)}$. Since L is $\mathcal{L}$-$\mathsf{uniform}$, $\mathsf{TM}_{\mathsf{spec}(\mathrm{c})}$ is also non-deterministic log-space Turing Machine, and therefore the computation of $\mathsf{TM}_{\mathsf{spec}(\mathrm{c})}$ can be delegated to the server. As a result, the bare-bones protocol is executed once for the delegation of C, and $2d$ times for the delegations of $\mathsf{TM}_{\mathsf{spec}(\mathrm{c})}$.

2.4 The Transformation of [KR09]

Assuming the existence of a poly-logarithmic computational Private Information Retrieval (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 $\mathcal{L}$-$\mathsf{uniform}\,\mathcal{NC}$ circuits.

We denote the verifier's message in this protocol by $\mathsf{GKR\text{-}KR}_{\mathrm{v}}(S, d, \lambda)$ given the circuit size S, depth d and security parameter λ. Similarly, we denote the prover's response by $\mathsf{GKR\text{-}KR}_{\mathrm{p}}(C, x, q)$ for a given circuit C, input x and queries $q = \mathsf{GKR\text{-}KR}_{\mathrm{v}}(S, d, \lambda)$.

3 Refereed Delegation of Computation

3.1 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.

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

Theorem 1 (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 $\varepsilon^{poly(k)}$, p is negligible. □

From Two Servers to N 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.

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

Feige and Kilian [FK97] construct a one-round refereed game for the sum-check task. In Appendix A we describe their protocol in detail (though our description below is self-contained).

The Protocol, Given a Circuit Specification Oracle. The intuition behind our protocol is as follows. We assume the referee has an oracle access to $\widetilde{\mathsf{spec}}_{C'(i)}$. 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 protocol of [FK97] assumes the referee can compute $f^{(i)}_{u_i}$ by himself for any point, but here, $f^{(i)}_{u_i}$ itself is too complex for the referee to compute. 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 are interested in 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 a random parametric curve $\varphi_i(t)$ and a random point z_i. ($\gamma_i(z_i)$ corresponds to the point u_i of [GKR08].) The referee sends $\varphi_i(z_i), \varphi_i(t)$ to P_1 and P_2, respectively, and asks the servers for the values of $\widetilde{V}_i(\varphi_i(z_i))$ and $\widetilde{V}_i(\varphi_i(t))$. Next, he checks whether those answers 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 $\widetilde{V}_i(\varphi_i(z_i))$. As we said, a subtle issue here is how the referee checks the correctness of the sum-checks without being able to compute $f^{(i)}_{\varphi_i(z_i)}$ by himself. Specifically, in the protocol of [FK97] the referee needs to compute $f^{(i)}_{\varphi_i(z_i)}$ on a point that is not known to P_1. In order to solve this problem we use the polynomial $\widetilde{V}_i(\varphi_i(t))$ to get this value "implicitly" from the servers themselves. When the referee believes that the answer on $\widetilde{V}_{i+1}(\varphi_{i+1}(t))$ for layer $i+1$ is correct, he takes few random points on $\widetilde{V}_{i+1}(\varphi_{i+1}(t))$ and uses that to compute the value of $f^{(i)}_{\varphi_i(z_i)}$ on the required point. The solution requires increasing by one the degrees of the polynomials of the protocol of [FK97] in order to keep the added point secret (see Appendix A). Note that P_2 can easily win if it knows the intersection point z_i, thus, we ask P_2 to answer the sum-check challenges for *all* the points on $\varphi_i(t)$, including the value at z_i. Then, our referee calls the referee of the one-round sum-check protocol of [FK97] to determine who is the honest server.

The detailed protocol (P_1, P_2, R) is presented in Fig. 1 and Fig. 2. 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 $poly(|F|, m, d, |H|) + n \cdot poly(|H|, m) = n \cdot poly(d, log(S))$, 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 2. *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 1. *Let V_1 be the view of P_1 and let i be a round in the protocol. For all $\alpha, \alpha', \beta, \beta' \in F$ and $j \in [1 \dots 3m]$*

$$Pr[a_j = \alpha | V_1] = Pr[a_j = \alpha' | V_1] \tag{1}$$
$$Pr[r = \beta | V_1] = Pr[r = \beta' | V_1]. \tag{2}$$

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

$$Pr[z_i = \alpha | V_2] = Pr[z_i = \alpha' | V_2] \tag{3}$$
$$Pr[a_j = \beta | V_2] = Pr[a_j = \beta' | V_2] \tag{4}$$
$$Pr[b_j = \gamma | V_2] = Pr[b_j = \gamma' | V_2] \tag{5}$$

Proof. The lemma follows from inspecting the protocol.

1. $D_j(t)$ 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.
2. Since P_1 has no information on w, w' besides from the curve C_{3m}, and, $\varphi_i(t)$ is of degree 2, r is simply a random point on that curve from his point of view.
3. $C_j(t)$ 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. Similar argument is true for b_j.

□

Proof (Theorem 2). Using Lemma 1, let's see how much a malicious server can cheat without knowing the intersection points z_i, a_j and b_j. For a fixed input x and a fixed circuit C, 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 $C(x)$).

Publicly known parameters

H, F, m, d, S, n, δ as in Sect. 2.

Initialization

For $i = 1, \ldots d$, R randomly picks $z_i \in F$ and a random degree-2 parametric curve $\varphi_i(t) \in F[t]^m$.

He also sets $\varphi_0 \equiv 0$ and $z_0 = 0$, and computes $Q_d(t) = \widetilde{V}_d(\varphi_d(t))$ and $M_d = \widetilde{V}_d(\varphi_d(z_d))$.

For $i = d, \ldots, 1$

R's computations :

R sets $w = \varphi_i(0), w' = \varphi_i(1)$ and randomly chooses $p, r \in F$.

For $1 \leq j \leq 3m$, R chooses random vectors $A_j, B_j \in F^j$ and random elements $a_j, b_j \in F$.

For $1 \leq j \leq 3m-1$ 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

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

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

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

We define

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

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

$C_j(t)$, for $1 \leq j \leq 3m$, and the point m_{i-1} where $m_{i-1} = \varphi_{i-1}(z_{i-1})$.

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

Define $F_j(t) = \Phi_{j,m_{i-1}}(C_j(t))$.

P_1 sends $F_j(t)$ for $1 \leq j \leq 3m$, and, M_{i-1} where $M_{i-1} = \widetilde{V}_{i-1}(m_{i-1})$.

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

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

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

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

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

Fig. 1. One-round RDoC protocol: initialization and interactive phase

Let E_i be the event that $Q_i(t)$ is indeed $\widetilde{V}_i(\varphi_i(t))$, and let T_i be the event that $Q_i(z_i)$ is indeed $\widetilde{V}_i(\varphi_i(z_i))$. Then,

$$Pr[S_2] \leq Pr[\exists i \in [d-1] \; s.t. \; \neg T_i \wedge T_{i+1}] \leq \sum_{i=0}^{d-1} Pr[\neg T_i \wedge T_{i+1}].$$

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

P_1 is declared as the cheater if for some j, $F_j(t)$ has degree greater than $\deg(C_j) \cdot j \cdot 2\delta$. P_2 is declared as the cheater if $Q_{i-1}(t)$ has degree bigger than $2m \cdot (|H| - 1)$ or if for some j, $G_{j,z_{i-1}}(t)$ has degree greater than $\deg(D_j) \cdot j \cdot 2\delta$.

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

R computes $f_{z_{i-1}}^{(i-1)}(p \circ w \circ w')$ using $Q_i(0), Q_i(1)$ (and the oracle).
Now, R verifies the sum-check of $M_{i-1} = \sum_{\hat{p},\hat{w},\hat{w}' \in H^m} f_{m_{i-1}}^{(i-1)}(\hat{p}, \hat{w}, \hat{w}')$ using the referee from [FK97]. Concretely:

- In case for all j, $G_{j,z_{i-1}}(a_j) = F_j(a_j)$, if $\sum_{h=0}^{|H|-1} F_1(h) = M_{i-1}$ then P_1 wins. Otherwise, P_2 wins.
- Denote by j the largest number such that $G_{j,z_{i-1}}(a_j) \neq F_j(a_j)$:
  - In case $1 \leq j < 3m$, if $\sum_{h=0}^{|H|-1} F_{j+1}(h) \neq G_{j,z_{i-1}}(b_j)$ then P_1 wins. Otherwise, P_2 wins.
  - In case $j = 3m$, if $F_{3m}(r) = f_{z_{i-1}}^{(i-1)}(p \circ w \circ w')$ then P_1 wins. Otherwise, P_2 wins.

Outputting the result

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

Fig. 2. One-round RDoC protocol: verification of answers

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

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

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

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

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

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

Therefore, we get that (assuming $2 \leq m$)

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

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

$$Pr[S_2] \leq d \cdot \frac{(|H|+1) \cdot 37m^2 \cdot \delta}{|F|}.$$

Let S_1 be the event that although P_1 is the malicious server, the referee outputs a wrong result (i.e., M_0 that is not equal to $C(x)$). The only step P_1 can cheat is in some of the executions of the [FK97] protocol. Thus, by union bound we get that $Pr[S_1]$ is also bounded by the same probability.

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

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 $\widetilde{\mathsf{spec}}_{\mathrm{c}(i)}$ of C_L can be computed by a log-space TM, which means that $\widetilde{\mathsf{spec}}_{\mathrm{c}(i)}$ can be computed by an $\mathcal{NL}$ circuit, $C_{\mathsf{spec}(L)}$. As shown in [GKR08], the circuit specification function $\widetilde{\mathsf{spec}}_{\mathrm{c}(i)}$ of circuits in $\mathcal{NL}$ can be computed in poly-logarithmic time. This means that the referee can compute $\widetilde{\mathsf{spec}}_{\mathrm{c}(i)}$ of $C_{\mathsf{spec}(L)}$ by himself, and execute the protocol from Sect. 3.2 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 referee runs the bare-bones protocol for verifying C_L, and asks the server for the required values of the circuit specification function $\widetilde{\mathsf{spec}}_{\mathrm{c}(i)}$ (i.e. the server acts as the oracle). Then, the referee checks each of those claimed values by executing the bare-bones protocol for the circuit $C_{\mathsf{spec}(L)}$ (for which he can compute the oracle answers by himself).

Now, if we try to follow this idea for extending the protocol from Sect. 3.2 to work with $\mathcal{L}$-uniform $\mathcal{NC}$ circuits, and try to run in parallel the protocol also for verification of $C_{\mathsf{spec}(L)}$, we get contradicting requirements. On the one hand, for verification of $C_{\mathsf{spec}(L)}$, both servers have to know p, w, w' 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 P_1.

In order to tackle this problem, we use a similar idea to the one used in the previous protocol. The referee asks P_1 to answer on many points, without revealing the actual p, w, w'. Note that p, w, w' is implicitly known to P_1 from $C_{3m}(t)$. P_2 already knows w, w' and we can send him also the value p 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 Sect. 3.2, where P_1' runs P_1 and P_2' runs P_2, with two modifications: 1) The referee sends to P_2' also the values of p for all layers, and, 2) P_2' sends the (claimed) values of $\widetilde{\mathsf{spec}}_{\mathrm{c}(i)}(b, p, w, w')$ for all layers.

For each of the answers $\widetilde{\mathsf{spec}}_{\mathrm{c}(i)}(b, p, w, w')$, the referee executes the protocol from Sect. 3.2 for verification of those claimed values using the circuit $C_{\mathsf{spec}(L)}$

(for which he can compute the circuit specification by himself). As we mentioned before, p, w, w' is not explicitly known to P_1'. So instead we ask it to answer on many points instead of the specific p, w, w'. Specifically, for verification of $\widetilde{\mathsf{spec}}_{\mathrm{c}(i)}(b, p, w, w')$ of layer i of C_L, we execute the protocol from Sect. 3.2, where P_2' plays the role of P_2 and knows p, w, w', and P_1' plays the role of P_1 for *all the points* on the curve $C_{3m}(t)$ as the possible inputs for $C_{\mathsf{spec}(L)}$. (There are at most $|F|$ points on this curve). This means that P_1' does not know the specific p, w, w'. However, since $C_{3m}(t)$ passes through p, w, w', one of those answers will be the needed P_1''s answer for the input p, w, w'.

When the referee receives the messages from both servers (for verification of $C_{\mathsf{spec}(L)}$ and of C_L), he checks if they agree on all the values of $\widetilde{\mathsf{spec}}_{\mathrm{c}(i)}(b, p, w, w')$. If they disagree on some of the values, then the referee checks one of those disagreements using the referee R from Sect. 3.2 and outputs according to his answer. If the servers agree on all the values of $\widetilde{\mathsf{spec}}_{\mathrm{c}(i)}$, 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 $\widetilde{\mathsf{spec}}_{\mathrm{c}(i)}$ he got before. He runs the checking phase of the referee from Sect. 3.2 and outputs according to his answer.

The overhead of this solution is only polynomial in all parameters since for each layer we have two invocations of the protocol form Sect. 3.2 where P_1' executes the protocol for $|F|$ points. Summing over all layers, the running time is increased by a factor of $2d \cdot |F|$ which is still poly-logarithmic in the size of the input.

Theorem 3. *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 $C_{\mathsf{spec}(L)}$ is independent of the messages for the verification of C_L. Those proofs share only one piece of information, the values of p, w, w' as the inputs for the circuit $C_{\mathsf{spec}(L)}$.

Let's assume P_1' is the cheater. He can cheat either on some value of $\widetilde{\mathsf{spec}}_{\mathrm{c}(i)}$ 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 Sect. 3.2. For the second case, if P_1' cheats on the computation of C_L (while the values of $\widetilde{\mathsf{spec}}_{\mathrm{c}(i)}$ are correct), then it means he can cheat in the protocol from Sect. 3.2 in the case where he has an oracle access to $\widetilde{\mathsf{spec}}_{\mathrm{c}(i)}$.

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

□

By Theorem 1 the error probability can be reduced exponentially using parallel repetition.

4 Offline/Online Verifiable Delegation of Computation with a Public Offline Stage

Previous constructions and definitions (e.g., [GGP10,CKV10,AIK10,CKLR11,BGV11]) 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.

A natural extension of this model is 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 powerful enough in that matter) can verify these keys and in particular publish an accusation proof in case some of the keys are invalid.
- A weak 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).

4.1 Splitting the [GKR08] Protocol

Given a circuit specification oracle, the bare-bones protocol from Sect. 2.2 requires the client to run in time proportional to $n \cdot \mathrm{poly}(d, \log(S))$. 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.

If we are interested in a larger class of languages, we can use an efficient computationally-sound protocol for realizing the oracle, and use the bare-bones protocol only for the verification of the computation itself. Specifically, given a circuit C, the client (or some other third party) computes in the offline stage the polynomials $\widetilde{\mathsf{spec}}_{\mathrm{c}(i)}$ 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). The total running time for computing this root is $\mathrm{poly}(S)$. Later on, in the (possibly many) online stages, the client and the server run the bare-bones protocol. When the client needs a value of the circuit specification predicate, he asks this value from the server, which returns it 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. Also, since the computation of the offline stage is deterministic, this computation can be done by the server and be verified by any interested third party (following the above example).

The combined protocol is similar to the one presented next, thus we omit further details.

4.2 One-round OVDoC with Public Offline Stage

Using a similar split and by utilizing the transformation of [KR09] we can construct a 1-round computationally-sound protocol with public offline stage for more than $\mathcal{L}$-uniform $\mathcal{NC}$ circuits. (Note that the [KR09] transformation of the [GKR08] protocol requires only a single round, without an offline stage. However, it works only with $\mathcal{L}$-uniform $\mathcal{NC}$ circuits.) The idea follows the transformation of [KR09]: Execute the bare-bones protocol under cPIR queries for verifying the computation of $C(x)$, and, query the values of the circuit specification predicate also under cPIR queries.

The Model. A One-Round Offline/Online Verifiable Delegation of Computation with Public Offline Stage scheme (denoted by 1RPDoC) 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 $\leq \text{poly}(T(F), \lambda)$.
- $\mathsf{ProbGen}(x, PK, \lambda) \rightarrow (k_x, c_x)$: 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 $\leq \text{poly}(T(F), \lambda)$.
- $\mathsf{VerifyResult}(PK, k_x, y, \pi_y) \rightarrow y \cup \perp$: Using the secret key k_x, the verification algorithm verifies the server's proof π_y and if succussed 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 $\text{poly}(T(F), \lambda)$ time) whether PK is a valid encoding of the target function F.

As for completeness and soundness, we require the following:

Definition 2 (ε-secure 1RPDoC). *We say that a scheme is a ε-secure 1RPDoC 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^*}, $Pr[\mathsf{VerifyResult}(PK, k_x, y^*, \pi_{y^*}) = \perp] \geq 1 - \varepsilon$.*

In concurrent work, Parno, Raykova and Vaikuntanathan [PRV12] propose two notions, *public delegatability* and *public verifiability*, and show a construction based on attribute-based encryption. The first notion is similar to ours except that in their definition the player that generates the public key (in the offline stage) is trusted. Their second notion allows *any* player to verify the computation, instead of just the player that delegated it. We note that in our definition the client uses fresh coins for each delegation, and therefore, in case he want to prove the server is cheating, he can publish his coins along with the server's answer. Thus, we can get the stronger notion by adding another message to our protocol. We remark that in the definition (and the construction) of [PRV12], a malicious client can frame the server by simply generating an invalid verification key, whereas in the above extension of our protocol, which can be applied also to the [PRV12] protocol, a malicious client cannot do that.

The Protocol. The protocol is presented in Fig. 3.

Theorem 4. *For any circuit C and for any constant error probability ε, the protocol from Fig. 3 is ε-secure 1RPDoC for the circuit C. In the online stage, the client runs in $n \cdot \mathrm{poly}(\mathrm{depth}(C), \log(|C|))$ time and the server in $\mathrm{poly}(|C|)$ time.*

Proof. Completeness follows from inspecting the protocol.

As for soundness, we 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

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 Sect. 2.2.

- Compute the polynomials $\widetilde{\mathsf{spec}}_{\mathrm{c}(i)}$ and the values of $\widetilde{\mathsf{spec}}_{\mathrm{c}(i)}(b, w_1, w_2, w_3)$ 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 $\mathsf{H}(\widetilde{\mathsf{spec}}_{\mathrm{c}(i)}(b, w_1, w_2, w_3))$.
- Return $PK = [ID_F, \mathrm{size}(C), \mathrm{depth}(C), \lambda, h, root)]$ 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 $m_1 = \mathsf{GKR\text{-}KR}_{\mathrm{v}}(\mathrm{size}(C), \mathrm{depth}(C), \lambda)$ 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 $i = 0 \ldots \mathrm{depth}(C)$) be the quadruplets that the protocol queries for their $\widetilde{\mathsf{spec}}_{\mathrm{c}(i)}$ 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 $\mathsf{GKR\text{-}KR}_{\mathrm{v}}$ 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 $\widetilde{\mathsf{spec}}_{\mathrm{c}(i)}(b, w_1, w_2, w_3)$ augmented with the Merkle Hash Tree proofs of consistency. I.e., the database consists of $2|F|^{3m} = \mathrm{poly}(\mathrm{size}(C))$ entries, where in the (a, b, c, d) entry there is the value of $\widetilde{\mathsf{spec}}_{\mathrm{c}(i)}(a, b, c, d)$ and the path in Merkle Hash Tree from $\mathsf{H}(\widetilde{\mathsf{spec}}_{\mathrm{c}(i)}(a, b, c, d))$ to the root.
- Let $(m_1, m_2) = c_x$. Compute $m_1' = \mathsf{GKR\text{-}KR}_{\mathrm{p}}(C, x, m_1)$ (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)**:**
Given the claimed output y, the proof $(m_1', m_2') = \pi_y$ and his 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 $\widetilde{\mathsf{spec}}_{\mathrm{c}(i)}$ is needed, use the answers from m_2'.

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

Fig. 3. 1RPDoC protocol

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]).

To the best of our knowledge, the resulting protocol is the only delegation protocol with a public offline stage and a single round in the online stage for more than $\mathcal{L}$-uniform $\mathcal{NC}$ circuits (based on standard assumptions). We note that the same technique can be applied to the protocols of [CKLR11] for memory and streaming delegation that are based on the protocol of [GKR08].

References

[AIK10] Applebaum, B., Ishai, Y., Kushilevitz, E.: From secrecy to soundness: efficient verification via secure computation. In: Proceedings of the 37th international colloquium conference on Automata, languages and programming, Springer-Verlag (2010) 152–163

[BCCR12] Bitansky, N., Canetti, R., Chiesa, A., Tromer, E.: From extractable collision resistance to succinct non-interactive arguments of knowledge, and back again. In: Proceedings of the 3rd Innovations in Theoretical Computer Science Conference, ACM (2012) 326–349

[BG02] Barak, B., Goldreich, O.: Universal arguments and their applications. SIAM J. Comput. **38** (December 2008) 1661–1694

[BGKW88] Ben-Or, M., Goldwasser, S., Kilian, J., Wigderson, A.: Multi-prover interactive proofs: how to remove intractability assumptions. In: Proceedings of the twentieth annual ACM symposium on Theory of computing, ACM (1988) 113–131

[BGV11] Benabbas, S., Gennaro, R., Vahlis, Y.: Verifiable delegation of computation over large datasets. In: Proceedings of the 31st annual conference on Advances in cryptology, Springer-Verlag (2011) 111–131

[BIN97] Bellare, M., Impagliazzo, R., Naor, M.: Does parallel repetition lower the error in computationally sound protocols? In: Proceedings of the 38th Annual Symposium on Foundations of Computer Science, IEEE Computer Society (1997) 374–383

[CHS05] Canetti, R., Halevi, S., Steiner, M.: Hardness amplification of weakly verifiable puzzles. In: Proceedings of the second Theory of Cryptography Conference, Springer-Verlag (2005) 17–33

[CKLR11] Chung, K.M., Kalai, Y.T., Liu, F.H., Raz, R.: Memory delegation. In: Proceedings of the 31st annual conference on Advances in cryptology, Springer-Verlag (2011) 151–165

[CKV10] Chung, K.M., Kalai, Y., Vadhan, S.: Improved delegation of computation using fully homomorphic encryption. In: Proceedings of the 30th annual conference on Advances in cryptology, Springer-Verlag (2010) 483–501

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

[CMT12] Cormode, G., Mitzenmacher, M., Thaler, J.: Practical verified computation with streaming interactive proofs. In: Proceedings of the 3rd Innovations in Theoretical Computer Science Conference, ACM (2012) 90–112

[CRR11] Canetti, R., Riva, B., Rothblum, G.N.: Practical delegation of computation using multiple servers. In: Proceedings of the 18th ACM conference on Computer and communications security, ACM (2011) 445–454

[DFH12] Damgård, I., Faust, S., Hazay, C.: Secure two-party computation with low communication. In: Proceedings of the 9th Theory of Cryptography Conference, Springer (2012) 54–74

[FK97] Feige, U., Kilian, J.: Making games short (extended abstract). In: Proceedings of the twenty-ninth annual ACM symposium on Theory of computing, ACM (1997) 506–516

[GGP10] Gennaro, R., Gentry, C., Parno, B.: Non-interactive verifiable computing: outsourcing computation to untrusted workers. In: Proceedings of the 30th annual conference on Advances in cryptology, Springer-Verlag (2010) 465–482

[GKR08] Goldwasser, S., Kalai, Y.T., Rothblum, G.N.: Delegating computation: interactive proofs for muggles. In: Proceedings of the 40th annual ACM symposium on Theory of computing, ACM (2008) 113–122

[GLR11] Goldwasser, S., Lin, H., Rubinstein, A.: Delegation of computation without rejection problem from designated verifier cs-proofs. Cryptology ePrint Archive, Report 2011/456 (2011) http://eprint.iacr.org/.

[K92] Kilian, J.: A note on efficient zero-knowledge proofs and arguments (extended abstract). In: Proceedings of the twenty-fourth annual ACM symposium on Theory of computing, ACM (1992) 723–732

[KR09] Kalai, Y.T., Raz, R.: Probabilistically checkable arguments. In: Proceedings of the 29th Annual International Cryptology Conference on Advances in Cryptology, Springer-Verlag (2009) 143–159

[KR11] Kol, G., Raz, R.: Competing provers protocols for circuit evaluation. Technical Report TR11-122, Electronic Colloquium on Computational Complexity (September 14 2011) http://eccc.hpi-web.de/.

[LFKN92] Lund, C., Fortnow, L., Karloff, H., Nisan, N.: Algebraic methods for interactive proof systems. J. ACM **39** (October 1992) 859–868

[M00] Micali, S.: Computationally sound proofs. SIAM J. Comput. **30** (October 2000) 1253–1298

[PRV12] Parno, B., Raykova, M., Vaikuntanathan, V.: How to delegate and verify in public: Verifiable computation from attribute-based encryption. In: Proceedings of the 9th Theory of Cryptography Conference, Springer (2012) 422–439

[R09] Rothblum, G.N.: Delegating computation reliably: paradigms and constructions. Ph.D. Thesis, Massachusetts Institute of Technology (2009)

A 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.[6] The referee can evaluate f by himself in polynomial time in the size of f. Server 1 claims that

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

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. 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

[6] The [FK97] protocol considers $f : \{0,1\}^k \to F$. We extend it to $f : F^k \to F$. Furthermore, the last stage of the protocol was simplified following a suggestion from an anonymous reviewer.

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 Fig. 4.

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: $C_1(t), \ldots, C_k(t)$.
P_1 sends to R: $N_0, F_1(t), \ldots, F_k(t)$, where $F_j(t) = \Phi_j(C_j(t))$.
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: P_1 loses immediately if for some j, $F_j(t)$ has degree greater than $|H| \cdot j \cdot d$. P_2 loses immediately if for some j, $G_j(t)$ has degree greater than $j \cdot d$.

In case for all j, $G_j(a_j) = F_j(a_j)$, if $\sum_{i=0}^{|H|-1} F_1(i) = N_0$ then P_1 wins. Otherwise, P_2 wins.

Denote by j the largest number such that $G_j(a_j) \neq F_j(a_j)$:

- In case $1 \leq j < k$, if $\sum_{i=0}^{|H|-1} F_{j+1}(i) \neq G_j(b_j)$ then P_1 wins. Otherwise, P_2 wins.
- In case $j = k$, if $F_k(r) = f(C_k(r))$ for a randomly chosen $r \in F$, then P_1 wins. Otherwise, P_2 wins.

Fig. 4. One-round refereed game for the sum-check task

Theorem 5. *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 $\leq \frac{|H| \cdot 2k^2 \cdot d}{|F|}$*

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 $f(C_k(r))$

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

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 calculation, we have that

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

The only differences are that in this case the degrees of $G_j(t)$ are smaller than the degrees of $F_j(t)$ by a factor of $|H|$, and, that we do not check $G_k(t)$ on a random point r. Therefore, the soundness of the protocol is bounded by $\frac{|H| \cdot 2k^2 \cdot d}{|F|}$. □

We remark that our protocol from Sect. 3.2 has another difference compared to the above protocol. We increase by one the degree of the curve C_k. Using a similar argument to the above it can be shown that the soundness of that protocol is bounded by $\frac{(|H|+1)\cdot 2k^2\cdot d}{|F|}$.