Sequentially Composable Rational Proofs

Matteo Campanelli and Rosario Gennaro(✉)

The City University of New York, New York, USA
mcampanelli@gradcenter.cuny.edu, rosario@ccny.cuny.edu

Abstract. We show that Rational Proofs do not satisfy basic compositional properties in the case where a large number of "computation problems" are outsourced. We show that a "fast" incorrect answer is more remunerable for the prover, by allowing him to solve more problems and collect more rewards. We present an enhanced definition of Rational Proofs that removes the economic incentive for this strategy and we present a protocol that achieves it for some uniform bounded-depth circuits.

1 Introduction

The problem of securely outsourcing data and computation has received widespread attention due to the rise of *cloud computing:* a paradigm where businesses lease computing resources from a service (the *cloud provider*) rather than maintain their own computing infrastructure. Small mobile devices, such as smart phones and netbooks, also rely on remote servers to store and perform computation on data that is too large to fit in the device.

It is by now well recognized that these new scenarios have introduced new security problems that need to be addressed. When data is stored remotely, outside our control, how can we be sure of its integrity? Even more interestingly, how do we check that the results of outsourced computation on this remotely stored data are correct. And how do perform these tests while preserving the efficiency of the client (i.e. avoid retrieving the whole data, and having the client perform the computation) which was the initial reason data and computations were outsourced.

Verifiable Outsourced Computation is a very active research area in Cryptography and Network Security (see [8] for a survey), with the goal of designing protocols where it is impossible (under suitable cryptographic assumptions) for a provider to "cheat" in the above scenarios. While much progress has been done in this area, we are still far from solutions that can be deployed in practice.

A different approach is to consider a model where "cheating" might actually be possible, but the provider would have no motivation to do so. In other words while cryptographic protocols prevent any adversary from cheating, one considers protocols that work against rational adversaries whose motivation is to maximize a well defined utility function.

This work was supported by NSF grant CNS-1545759

MHR Khouzani et al. (Eds.): GameSec 2015, LNCS 9406, pp. 270–288, 2015.
DOI: 10.1007/978-3-319-25594-1_15

Previous Work. An earlier work in this line is [3] where the authors describe a system based on a scheme of rewards [resp. penalties] that the client assesses to the server for computing the function correctly [resp. incorrectly]. However in this system checking the computation may require re-executing it, something that the client does only on a randomized subset of cases, hoping that the penalty is sufficient to incentivize the server to perform honestly. Morever the scheme might require an "infinite" budget for the rewards, and has no way to "enforce" payment of penalties from cheating servers. For these reasons the best application scenario of this approach is the incentivization of volunteer computing schemes (such as SETI@Home or Folding@Home), where the rewards are non-fungible "points" used for "social-status".

Because verification is performed by re-executing the computation, in this approach the client is "efficient" (i.e. does "less" work than the server) only in an amortized sense, where the cost of the subset of executions verified by the client is offset by the total number of computations performed by the server. This implies that the server must perform many executions for the client.

Another approach, instead, is the concept of Rational Proofs introduced by Azar and Micali in [1] and refined in subsequent papers [2,6]. This model captures, more accurately, real-world financial "pay-for-service" transactions, typical of cloud computing contractual arrangements, and security holds for a single "stand-alone" execution.

In a Rational Proof, given a function f and an input x, the server returns the value $y = f(x)$, and (possibly) some auxiliary information, to the client. The client will in turn pay the server for its work with a reward which is a function of the messages sent by the server and some randomness chosen by the client. The crucial property is that this reward is maximized in expectation when the server returns the correct value y. Clearly a rational prover who is only interested in maximizing his reward, will always answer correctly.

The most striking feature of Rational Proofs is their simplicity. For example in [1], Azar and Micali show single-message Rational Proofs for any problem in $\#P$, where an (exponential-time) prover convinces a (poly-time) verifier of the number of satisfying assignment of a Boolean formula.

For the case of "real-life" computations, where the Prover is polynomial and the Verifier is as efficient as possible, Azar and Micali in [2] show d-round Rational Proofs for functions computed by (uniform) Boolean circuits of depth d, for $d = O(\log n)$ (which can be collapsed to a single round under some well-defined computational assumption as shown in [6]). The problem of rational proofs for any polynomial-time computable function remains tantalizingly open.

Our Results. Motivated by the problem of volunteer computation, our first result is to show that the definition of Rational Proofs in [1,2] does not satisfy a basic compositional property which would make them applicable in that scenario. Consider the case where a large number of "computation problems" are outsourced. Assume that solving each problem takes time T. Then in a time interval of length T, the honest prover can only solve and receive the reward for a single problem. On the other hand a dishonest prover, can answer up to

T problems, for example by answering at random, a strategy that takes $O(1)$ time. To assure that answering correctly is a rational strategy, we need that at the end of the T-time interval the reward of the honest prover be larger than the reward of the dishonest one. But this is not necessarily the case: for some of the protocols in [1,2,6] we can show that a "fast" incorrect answer is more remunerable for the prover, by allowing him to solve more problems and collect more rewards.

The next questions, therefore, was to come up with a definition and a protocol that achieves rationality both in the stand-alone case, and in the composition described above. We first present an enhanced definition of Rational Proofs that removes the economic incentive for the strategy of fast incorrect answers, and then we present a protocol that achieves it for the case of some (uniform) bounded-depth circuits.

2 Rational Proofs

In the following we will adopt a "concrete-security" version of the "asymptotic" definitions and theorems in [2,6]. We assume the reader is familiar with the notion of interactive proofs [7].

Definition 1 (Rational Proof). *A function $f : \{0,1\}^n \to \{0,1\}^n$ admits a rational proof if there exists an interactive proof (P,V) and a randomized reward function* $\mathsf{rew} : \{0,1\}^* \to \mathbb{R}_{\geq 0}$ *such that*

1. (Rational completeness) *For any input $x \in \{0,1\}^n$,* $\Pr[\mathsf{out}((P,V)(x)) = f(x)] = 1$.
2. *For every prover $\widetilde{P}$, and for any input $x \in \{0,1\}^n$ there exists a $\delta_{\widetilde{P}}(x) \geq 0$ such that $\mathbb{E}[\mathsf{rew}((\widetilde{P},V)(x))] + \delta_{\widetilde{P}}(x) \leq \mathbb{E}[\mathsf{rew}((P,V)(x))]$.*

The expectations and the probabilities are taken over the random coins of the prover and verifier.

Let $\epsilon_{\widetilde{P}} = \Pr[\mathsf{out}((P,V)(x)) \neq f(x)]$. Following [6] we define the reward gap as

$$\Delta(x) = min_{P^*:\epsilon_{P^*}=1}[\delta_{P^*}(x)]$$

i.e. the minimum reward gap over the provers that always report the incorrect value. It is easy to see that for arbitrary prover $\widetilde{P}$ we have $\delta_{\widetilde{P}}(x) \geq \epsilon_{\widetilde{P}} \cdot \Delta(x)$. Therefore it suffices to prove that a protocol has a strictly positive reward gap $\Delta(x)$ for all x.

Examples of Rational Proofs. For concreteness here we show the protocol for a single threshold gate (readers are referred to [1,2,6] for more examples).

Let $G_{n,k}(x_1,\ldots,x_n)$ be a threshold gate with n Boolean inputs, that evaluates to 1 if at least k of the input bits are 1. The protocol in [2] to evaluate this gate goes as follows. The Prover announces the number $\tilde{m}$ of input bits equal to 1, which allows the Verifier to compute $G_{n,k}(x_1,\ldots,x_n)$. The Verifier select

a random index $i \in [1..n]$ and looks at input bit $b = x_i$ and rewards the Prover using Brier's Rule $BSR(\tilde{p}, b)$ where $\tilde{p} = \tilde{m}/n$ i.e. the probability claimed by the Prover that a randomly selected input bit be 1. Then

$$BSR(\tilde{p}, 1) = 2\tilde{p} - \tilde{p}^2 - (1-\tilde{p})^2 + 1 = 2\tilde{p}(2-\tilde{p})$$

$$BSR(\tilde{p}, 0) = 2(1-\tilde{p}) - \tilde{p}^2 - (1-\tilde{p})^2 + 1 = 2(1-\tilde{p}^2)$$

Let m be the true number of input bits equal to 1, and $p = m/n$ the corresponding probability, then the expected reward of the Prover is

$$pBSR(\tilde{p}, 1) + (1-p)BSR(\tilde{p}, 0) \tag{1}$$

which is easily seen to be maximized for $p = \tilde{p}$ i.e. when the Prover announces the correct result. Moreover one can see that when the Prover announces a wrong $\tilde{m}$ his reward goes down by $2(p-\tilde{p})^2 \geq 2/n^2$. In other words for all n-bit input x, we have $\Delta(x) = 2/n^2$ and if a dishonest Prover $\widetilde{P}$ cheats with probability $\epsilon_{\widetilde{P}}$ then $\delta_{\widetilde{P}} > 2\epsilon_{\widetilde{P}}/n^2$.

3 Profit vs. Reward

Let us now define the profit of the Prover as the difference between the reward paid by the verifier and the cost incurred by the Prover to compute f and engage in the protocol. As already pointed out in [2,6] the definition of Rational Proof is sufficiently robust to also maximize the profit of the honest prover and not the reward. Indeed consider the case of a "lazy" prover $\widetilde{P}$ that does not evaluate the function: even if $\widetilde{P}$ collects a "small" reward, his total profit might still be higher than the profit of the honest prover P.

Set $R(x) = \mathbb{E}[\mathsf{rew}((P,V)(x))]$, $\tilde{R}(x) = \mathbb{E}[\mathsf{rew}((\widetilde{P},V)(x))]$ and $C(x)$ [resp. $\tilde{C}(x)$] the cost for P [resp. $\widetilde{P}$] to engage in the protocol. Then we want

$$R(x) - C(x) \geq \tilde{R}(x) - \tilde{C}(x) \implies \delta_{\widetilde{P}}(x) \geq C(x) - \tilde{C}(x)$$

In general this is not true (see for example the previous protocol), but it is always possible to change the reward by a multiplier M. Note that if $M \geq C(x)/\delta_{\widetilde{P}}(x)$ then we have that

$$M(R(x) - \tilde{R}(x)) \geq C(x) \geq C(x) - \tilde{C}(x)$$

as desired. Therefore by using the multiplier M in the reward, the honest prover P maximizes its profit against all provers $\widetilde{P}$ except those for which $\delta_{\widetilde{P}}(x) \leq C(x)/M$, i.e. those who report the incorrect result with a "small" probability $\epsilon_{\widetilde{P}}(x) \leq \frac{C(x)}{M\Delta(x)}$.

We note that M might be bounded from above, by budget considerations (i.e. the need to keep the total reward $MR(x) \leq B$ for some budget B). This point out to the importance of a large reward gap $\Delta(x)$ since the larger $\Delta(x)$ is, the smaller the probability of a cheating prover $\widetilde{P}$ to report an incorrect result must be, in order for $\widetilde{P}$ to achieve an higher profit than P.

Example. In the above protocol we can assume that the cost of the honest prover is $C(x) = n$, and we know that $\Delta(x) = n^2$. Therefore the profit of the honest prover is maximized against all the provers that report an incorrect result with probability larger than n^3/M, which can be made sufficiently small by choosing the appropriate multiplier.

Remark 1. If we are interested in an asymptotic treatment, it is important to notice that as long as $\Delta(x) \geq 1/\mathsf{poly}(|x|)$ then it is possible to keep a polynomial reward budget, and maximize the honest prover profit against all provers who cheat with a substantial probability $\epsilon_{\widetilde{P}} \geq 1/\mathsf{poly}'(|x|)$.

4 Sequential Composition

We now present the main results of our work. First we informally describe our notion of sequential composition of rational proof, via a motivating example and show that the protocols in [1,2,6] do not satisfy it. Then we present our definition of sequential rational proofs, and a protocol that achieves it for circuits of bounded depth.

4.1 Motivating Example

Consider the protocol in the previous section for the computation of the function $G_{n,k}(\cdot)$. Assume that the honest execution of the protocol (including the computation of $G_{n,k}(\cdot)$) has cost $C = n$.

Assume now that we are given a sequence of n inputs $x^{(1)}, \ldots, x^{(i)}, \ldots$ where each $x^{(i)}$ is an n-bit string. In the following let m_i be the Hamming weight of $x^{(i)}$ and $p_i = m_i/n$.

Therefore the honest prover investing $C = n$ cost, will be able to execute the protocol only once, say on input $x^{(i)}$. By setting $p = \tilde{p} = p_i$ in Eq. 1, we see that P obtains reward

$$R(x^{(i)}) = 2(p_i^2 - p_i + 1) \leq 2$$

Consider instead a prover $\widetilde{P}$ which in the execution of the protocol outputs a random value $\tilde{m} \in [0..n]$. The expected reward of $\widetilde{P}$ on any input $x^{(i)}$ is (by setting $p = p_i$ and $\tilde{p} = m/n$ in Eq. 1 and taking expectations):

$$\begin{aligned}
\tilde{R}(x^{(i)}) &= \mathop{\mathbb{E}}_{m,b}[BSR(\frac{m}{n}, b)] \\
&= \frac{1}{n+1} \sum_{m=0}^{n} \mathop{\mathbb{E}}_{b}[BSR(\frac{m}{n}, b] \\
&= \frac{1}{n+1} \sum_{m=0}^{n} (2(2p_i \cdot \frac{m}{n} - \frac{m^2}{n^2} - p_i + 1)) \\
&= 2 - \frac{2n+1}{3n} > 1 \quad \text{for } n > 1.
\end{aligned}$$

Therefore by "solving" just two computations $\widetilde{P}$ earns more than P. Moreover t the strategy of $\widetilde{P}$ has cost 1 and therefore it earns more than P by investing a lot less cost[1].

Note that "scaling" the reward by a multiplier M does not help in this case, since both the honest and dishonest prover's rewards would be multiplied by the same multipliers, without any effect on the above scenario.

We have therefore shown a rational strategy, where cheating many times and collecting many rewards is more profitable than collecting a single reward for an honest computation.

4.2 Sequentially Composable Rational Proofs

The above counterexample motivates the following Definition which formalizes that the reward of the honest prover P must always be larger than the total reward of any prover $\widetilde{P}$ that invests less computation cost than P.

Technically this is not trivial to do, since it is not possible to claim the above for *any* prover $\widetilde{P}$ and *any* sequence of inputs, because it is possible that for a given input $\tilde{x}$, the prover $\widetilde{P}$ has "hardwired" the correct value $\tilde{y} = f(\tilde{x})$ and can compute it without investing any work. We therefore propose a definition that holds for inputs randomly chosen according to a given probability distribution $\mathcal{D}$, and we allow for the possibility that the reward of a dishonest prover can be "negligibly" larger than the reward of the honest prover (for example if $\widetilde{P}$ is lucky and such "hardwired" inputs are selected by $\mathcal{D}$).

Definition 2 (Sequential Rational Proof). *A rational proof (P, V) for a function $f : \{0,1\}^n \to \{0,1\}^n$ is ϵ-*sequentially composable *for an input distribution $\mathcal{D}$, if for every prover $\widetilde{P}$, and every sequence of inputs $x, x_1, \ldots, x_k \in \mathcal{D}$ such that $C(x) \geq \sum_{i=1}^{k} \tilde{C}(x_i)$ we have that $\sum_i \tilde{R}(x_i) - R \leq \epsilon$.*

A few sufficient conditions for sequential composability follow.

Lemma 1. *Let (P, V) be a rational proof. If for every input x it holds that $R(x) = R$ and $C(x) = C$ for constants R and C, and the following inequality holds for every $\widetilde{P} \neq P$ and input $x \in \mathcal{D}$:*

$$\frac{\tilde{R}(x)}{R} \leq \frac{\tilde{C}(x)}{C} + \epsilon$$

then (P, V) is $kR\epsilon$-sequentially composable for $\mathcal{D}$.

Proof. It suffices to observe that, for any k inputs $x_1, ..., x_k$, the inequality above implies

$$\sum_{i=1}^{k} \tilde{R}(x_i) \leq R[\sum_{i=1}^{k}(\frac{\tilde{C}(x_i)}{C} + \epsilon)] \leq R + kR\epsilon$$

where the last inequality holds whenever $\sum_{i=1}^{k} \tilde{C}(x_i) \leq C$ as in Definition 2.

[1] If we think of cost as time, then in the same time interval in which P solves one problem, $\widetilde{P}$ can solve up to n problems, earning a lot more money, by answering fast and incorrectly.

Corollary 1. *Let* (P, V) *and* rew *be respectively an interactive proof and a reward function as in Definition 1; if* rew *can only assume the values* 0 *and* R *for some constant* R*, let* $\tilde{p}_x = \Pr[\mathsf{rew}((\tilde{P}, V)(x)) = R]$*. If for* $x \in \mathcal{D}$

$$\tilde{p}_x \leq \frac{\tilde{C}(x)}{C} + \epsilon$$

then (P, V) *is* $kR\epsilon$*-sequentially composable for* $\mathcal{D}$*.*

Proof. Observe that $\tilde{R}(x) = \tilde{p}_x \cdot R$ and then apply Lemma 1.

4.3 Sequential Rational Proofs in the PCP Model

We now describe a rational proof appeared in [2] and prove that is sequentially composable. The protocol assumes the existence of a trusted memory storage to which both Prover and Verifier have access, to realize the so-called "PCP" (Probabilistically Checkable Proof) model. In this model, the Prover writes a very long proof of correctness, that the verifier checks only in a few randomly selected positions. The trusted memory is needed to make sure that the prover is "committed" to the proof before the verifier starts querying it.

The following protocol for proofs on a binary logical circuit $\mathcal{C}$ appeared in [2]. The Prover writes all the (alleged) values α_w for every wire $w \in \mathcal{C}$, on the trusted memory location. The Verifier samples a single random gate value to check its correctness and determines the reward accordingly:

1. The Prover writes the vector $\{\alpha_w\}_{w \in \mathcal{C}}$
2. The Verifier samples a random gate $g \in \mathcal{C}$.
   - The Verifier reads $\alpha_{g_{out}}, \alpha_{g_L}, \alpha_{g_R}$, with g_{out}, g_L, g_R being respectively the output, left and right input wires of g; the verifier checks that $\alpha_{g_{out}} = g(\alpha_{g_L}, \alpha_{g_R})$;
   - If g in an input gate the Verifier also checks that $\alpha_{g_L}, \alpha_{g_R}$ correspond to the correct input values;

   The Verifier pays R if both checks are satisfied, otherwise it pays 0.

Theorem 1 ([2]). *The protocol above is a rational proof for any boolean function in* $P^{||NP}$*, the class of all languages decidable by a polynomial time machine that can make non-adaptive queries to* NP*.*

We will now show a cost model where the rational proof above is sequentially composable. We will assume that the cost for any prover is given by the number of gates he writes. Thus, for any input x, the costs for honest and dishonest provers are respectively $C(x) = S$, where $S = |\mathcal{C}|$, and $\tilde{C}(x) = \tilde{s}$ where $\tilde{s}$ is the number of gates written by the dishonest prover. Observe that in this model a dishonest prover may not write all the S gates, and that not all of the $\tilde{s}$ gates have to be correct. Let $\sigma \leq \tilde{s}$ the number of correct gates written by $\tilde{P}$.

Theorem 2. *In the cost model above the PCP protocol in [2] is sequentially composable.*

Proof. Observe that the probability $\tilde{p}_x$ that $\widetilde{P} \neq P$ earns R is such that

$$\tilde{p}_x = \frac{\sigma}{S} \leq \frac{\tilde{s}}{S} = \frac{\tilde{C}}{C}$$

Applying Corollary 1 completes the proof.

The above cost model, basically says that the cost of writing down a gate dominates everything else, in particular the cost of *computing* that gate. In other cost models a proof of sequential composition may not be as straightforward. Assume, for example, that the honest prover pays \$1 to compute the value of a single gate while writing down that gate is "free". Now $\tilde{p}_x$ is still equal to $\frac{\sigma}{S}$ but to prove that this is smaller than $\frac{\tilde{C}}{C}$ we need some additional assumption that limits the ability for $\widetilde{P}$ to "guess" the right value of a gate without computing it (which we will discuss in the next Section).

4.4 Sequential Composition and the Unique Inner State Assumption

Definition 2 for sequential rational proofs requires a relationship between the reward earned by the prover and the amount of "work" the prover invested to produce that result. The intuition is that to produce the correct result, the prover must run the computation and incur its full cost. Unfortunately this intuition is difficult, if not downright impossible, to formalize. Indeed for a specific input x a "dishonest" prover $\widetilde{P}$ could have the correct $y = f(x)$ value "hardwired" and could answer correctly without having to perform any computation at all. Similarly, for certain inputs x, x' and a certain function f, a prover $\widetilde{P}$ after computing $y = f(x)$ might be able to "recycle" some of the computation effort (by saving some state) and compute $y' = f(x')$ incurring a much smaller cost than computing it from scratch.

A way to circumvent this problem was suggested in [3] under the name of *Unique Inner State Assumption*: the idea is to assume a distribution $\mathcal{D}$ over the input space. When inputs x are chosen according to $\mathcal{D}$, then we assume that computing f requires cost C from any party: this can be formalized by saying that if a party invests $\tilde{C} = \gamma C$ effort (for $\gamma \leq 1$), then it computes the correct value only with probability negligibly close to γ (since a party can always have a "mixed" strategy in which with probability γ it runs the correct computation and with probability $1 - \gamma$ does something else, like guessing at random).

Assumption 1. *We say that the (C, ϵ)-Unique Inner State Assumption holds for a function f and a distribution $\mathcal{D}$ if for any algorithm $\widetilde{P}$ with cost $\tilde{C} = \gamma C$, the probability that on input $x \in \mathcal{D}$, $\widetilde{P}$ outputs $f(x)$ is at most $\gamma + (1 - \gamma)\epsilon$.*

Note that the assumption implicitly assumes a "large" output space for f (since a random guess of the output of f will be correct with probability 2^{-n} where n is the binary length of $f(x)$).

More importantly, note that Assumption 1 immediately yields our notion of sequential composability, if the Verifier can detect if the Prover is lying or not. Assume, as a mental experiment for now, that given input x, the Prover claims that $\tilde{y} = f(x)$ and the Verifier checks by recomputing $y = f(x)$ and paying a reward of R to the Prover if $y = \tilde{y}$ and 0 otherwise. Clearly this is not a very useful protocol, since the Verifier is not saving any computation effort by talking to the Prover. But it is sequentially composable according to our definition, since $\tilde{p}_x$, the probability that $\widetilde{P}$ collects R, is equal to the probability that $\widetilde{P}$ computes $f(x)$ correctly, and by using Assumption 1 we have that

$$\tilde{p}_x = \gamma + (1-\gamma)\epsilon \leq \frac{\tilde{C}}{C} + \epsilon$$

satisfying Corollary 1.

To make this a useful protocol we adopt a strategy from [3], which also uses this idea of verification by recomputing. Instead of checking every execution, we check only a random subset of them, and therefore we can amortize the Verifier's effort over a large number of computations. Fix a parameter m. The prover sends to the verifier the values $\tilde{y}_j$ which are claimed to be the result of computing f over m inputs $x_1, \ldots, x_m$. The verifier chooses one index i randomly between 1 and m, and computes $y_i = f(x_i)$. If $y_i = \tilde{y}_i$ the verifier pays R, otherwise it pays 0.

Let T be the total cost by the honest prover to compute m instances: cleary $T = mC$. Let $\tilde{T} = \Sigma_i \tilde{C}_i$ be the total effort invested by $\tilde{P}$, by investing $\tilde{C}_i$ on the computation of x_i. In order to satisfy Corollary 1 we need that $\tilde{p}_x$, the probability that $\widetilde{P}$ collects R, be less than $\tilde{T}/T + \epsilon$.

Let $\gamma_i = \tilde{C}_i/C$, then under Assumption 1 we have that $\tilde{y}_i$ is correct with probability at most $\gamma_i + (1-\gamma_i)\epsilon$. Therefore if we set $\gamma = \sum_i \gamma_i/m$ we have

$$\tilde{p}_x = \frac{1}{m}\sum_i [\gamma_i + (1-\gamma_i)\epsilon] = \gamma + (1-\gamma)\epsilon \leq \gamma + \epsilon$$

But note that $\gamma = \tilde{T}/T$ as desired since

$$\tilde{T} = \sum_i \tilde{C}_i = \sum_i \gamma_i C = T \sum_i \gamma_i/m$$

Efficiency of the Verifier. If our notion of "efficient Verifier" is a verifier who runs in time $o(C)$ where C is the time to compute f, then in the above protocol m must be sufficiently large to amortize the cost of computing one execution over many (in particular a constant – in the input size n – value of m would not work). In our "concrete analysis" treatment, if we requests that the Verifier runs in time δC for an "efficiency" parameter $\delta \leq 1$, then we need $m \geq \delta^{-1}$.

Therefore we are still in need of a protocol which has an efficient Verifier, and would still works for the "stand-alone" case ($m = 1$) but also for the case of sequential composability over any number m of executions.

5 Our Protocol

We now present a protocol that works for functions $f : \{0,1\}^n \to \{0,1\}^n$ expressed by an arithmetic circuit $\mathcal{C}$ of size C and depth d and fan-in 2, given as a common input to both Prover and Verifier together with the input x.

Intuitively the idea is for the Prover to provide the Verifier with the output value y and its two "children" y_L, y_R in the gate, i.e. the two input values of the last output gate G. The Verifier checks that $G(y_L, y_R) = y$, and then asks the Prover to verify that y_L or y_R (chosen a random) is correct, by recursing on the above test. The protocol description follows.

1. The Prover evaluates the circuit on x and sends the output value y_1 to the Verifier.
2. **Repeat** r **times:** The Verifier identifies the root gate g_1 and then invokes $Round(1, g_1, y_1)$,

where the procedure $Round(i, g_i, y_i)$ is defined for $1 \leq i \leq d$ as follows:

1. The Prover sends the value of the input wires z_i^0 and z_i^1 of g_i to the Verifier.
2. The Verifiers performs the following
   - Check that y_i is the result of the operation of gate g_i on inputs z_i^0 and z_i^1. If not **STOP** and pay a reward of 0.
   - If $i = d$ (i.e. if the inputs to g_i are input wires), check that the values of z_i^0 and z_i^1 are equal to the corresponding bits of x. Pay reward R to Merlin if this is the case, nothing otherwise.
   - If $i < d$, choose a random bit b, send it to Merlin and invoke $Round(i + 1, g_{i+1}^b, z_i^b)$ where g_{i+1}^b is the child gate of g_i whose output is z_i^b.

5.1 Efficiency

The protocol runs at most in d rounds. In each round, the Prover sends a constant number of bits representing the values of specific input and output wires; The Verifier sends at most one bit per round, the choice of the child gate. Thus the communication complexity is $O(d)$ bits.

The computation of the Verifier in each round is: (i) computing the result of a gate and checking for bit equality; (ii) sampling a child. Gate operations and equality are $O(1)$ per round. We assume our circuits are T-uniform, which allows the Verifier to select the correct gate in time $T(n)$[2]. Thus the Verifier runs in $O(rd \cdot T(n))$ with $r = O(\log C)$.

[2] We point out that the Prover can provide the Verifier with the requested gate and then the Verifier can use the uniformity of the circuit to check that the Prover has given him the correct gate at each level in time $O(T(n))$.

5.2 Proofs of (Stand-Alone) Rationality

Theorem 3. *The protocol in Sect. 5 for $r = 1$ is a Rational Proof according to Definition 1.*

We prove the above theorem by showing that for every input x the reward gap $\Delta(x)$ is positive.

Proof. Let $\widetilde{P}$ a prover that always reports $\tilde{y} \neq y_1 = f(x)$ at Round 1.

Let us proceed by induction on the depth d of the circuit. If $d = 1$ then there is no possibility for $\widetilde{P}$ to cheat successfully, and its reward is 0.

Assume $d > 1$. We can think of the binary circuit $\mathcal{C}$ as composed by two subcircuits $\mathcal{C}_L$ and $\mathcal{C}_R$ and the output gate g_1 such that $f(x) = g_1(\mathcal{C}_L(x), \mathcal{C}_R(x))$. The respective depths d_L, d_R of these subcircuits are such that $0 \leq d_L, d_R \leq d-1$ and $max(d_L, d_R) = d-1$. After sending $\tilde{y}$, the protocol requires that $\widetilde{P}$ sends output values for $C_L(x)$ and $C_R(x)$; let us denote these claimed values respectively with $\tilde{y}_L$ and $\tilde{y}_R$. Notice that at least one of these alleged values will be different from the respective correct subcircuit output: if it were otherwise, V would reject immediately as $g(\tilde{y}_L, \tilde{y}_R) = f(x) \neq \tilde{y}$. Thus at most one of the two values $\tilde{y}_L$, $\tilde{y}_R$ is equal to the output of the corresponding subcircuit. The probability that the $\widetilde{P}$ cheats successfully is:

$$\Pr[\text{V accepts}] \leq \frac{1}{2} \cdot (\Pr[\text{V accepts on } C_L] + \Pr[\text{V acceptson } C_R]) \tag{2}$$

$$\leq \frac{1}{2} \cdot (1 - 2^{-\max(d_L, d_R)}) + \frac{1}{2} \tag{3}$$

$$\leq \frac{1}{2} \cdot (1 - 2^{-d+1}) + \frac{1}{2} \tag{4}$$

$$= 1 - 2^{-d} \tag{5}$$

At Eq. 3 we used the inductive hypothesis and the fact that all probabilities are at most 1.

Therefore the expected reward of $\widetilde{P}$ is $\tilde{R} \leq R(1 - 2^{-d})$ and the reward gap is $\Delta(x) = 2^{-d}R$ (see Remark 2 or an explanation of the equality sign).

The following useful corollary follows from the proof above.

Corollary 2. *If the protocol described in Sect. 5 is repeated $r \geq 1$ times a prover can cheat with probability at most $(1 - 2^{-d})^r$.*

Remark 2. We point out that one can always build a prover strategy P^* which always answers incorrectly and achieves exactly the reward $R^* = R(1 - 2^{-d})$. This prover outputs an incorrect $\tilde{y}$ and then computes one of the subcircuits that results in one of the input values (so that at least one of the inputs is correct). This will allow him to recursively answer with values z_i^0 and z_i^1 where one of the two is correct, and therefore be caught only with probability 2^{-d}.

Remark 3. In order to have a non-negligible reward gap (see Remark 1) we need to limit ourselves to circuits of depth $d = O(\log n)$.

5.3 Proof of Sequential Composability

General Sufficient Conditions for Sequential Composability

Lemma 2. *Let $\mathcal{C}$ be a circuit of depth d. If the (C, ϵ) Unique Inner State Assumption (see Assumption 1) holds for the function f computed by $\mathcal{C}$, and input distribution $\mathcal{D}$, then the protocol presented above with r repetitions is a $k\epsilon R$-sequentially composable Rational Proof for $\mathcal{C}$ for $\mathcal{D}$ if the following inequality holds*

$$(1 - 2^{-d})^r \leq \frac{1}{C}$$

Proof. Let $\gamma = \frac{\tilde{C}}{C}$. Consider $x \in \mathcal{D}$ and prover $\widetilde{P}$ which invests effort $\tilde{C} \leq C$. Under Assumption 1, $\widetilde{P}$ gives the correct outputs with probability $\gamma + \epsilon$ – assume that in this case $\widetilde{P}$ collects the reward R. If $\widetilde{P}$ gives an incorrect output we know (following Corollary 2) that he collects the reward R with probability at most $(1 - 2^{-d})^r$ which by hypothesis is less than γ. So either way we have that $\tilde{R} \leq (\gamma + \epsilon)R$ and therefore applying Lemma 1 concludes the proof.

The problem with the above Lemma is that it requires a large value of r for the result to be true resulting in an inefficient Verifier. In the following sections we discuss two approaches that will allows us to prove sequential composability even for an efficient Verifier:

- Limiting the class of provers we can handle in our security proof;
- Limiting the class of functions/circuits.

Limiting the Strategy of the Prover: Non-adaptive Provers. In proving sequential composability it is useful to find a connection between the amount of work done by a dishonest prover and its probability of cheating. The more a dishonest prover works, the higher its probability of cheating. This is true for our protocol, since the more "subcircuits" the prover computes correctly, the higher is the probability of convincing the verifier of an incorrect output becomes. The question then is: how can a prover with an "effort budget" to spend maximize its probability of success in our protocol?

As we discussed in Remark 2, there is an *adaptive* strategy for the $\widetilde{P}$ to maximize its probability of success: compute one subcircuit correctly at every round of the protocol. We call this strategy "adaptive", because the prover allocates its "effort budget" $\tilde{C}$ on the fly during the execution of the rational proof. Conversely a *non-adaptive* prover $\widetilde{P}$ uses $\tilde{C}$ to compute some subcircuits in $\mathcal{C}$ before starting the protocol. Clearly an adaptive prover strategy is more powerful, than a non-adaptive one (since the adaptive prover can direct its computation effort where it matters most, i.e. where the Verifier "checks" the computation).

Is it possible to limit the Prover to a non-adaptive strategy? This could be achieved by imposing some "timing" constraints to the execution of the protocol: to prevent the prover from performing large computations while interacting with the Verifier, the latter could request that prover's responses be delivered

"immediately", and if a delay happens then the Verifier will not pay the reward. Similar timing constraints have been used before in the cryptographic literature, e.g. see the notion of *timing assumptions* in the concurrent zero-knowledge protocols in [5].

Therefore in the rest of this subsection we assume that non-adaptive strategies are the only rational ones and proceed in analyzing our protocol under the assumption that the prover is adopting a non-adaptive strategy.

Consider a prover $\widetilde{P}$ with effort budget $\tilde{C} < C$. A *DFS* (for "depth first search") prover uses its effort budget $\tilde{C}$ to compute a whole subcircuit of size $\tilde{C}$ and maximal depth d_{DFS}. Call this subcircuit $\mathcal{C}_{\mathcal{DFS}}$. $\widetilde{P}$ can answer correctly any verifier's query about a gate in $\mathcal{C}_{\mathcal{DFS}}$. During the interaction with V, the behavior of a DFS prover is as follows:

- At the beginning of the protocol send an arbitrary output value y_1.
- During procedure $Round(i, g_i, y_i)$:
  - If $g_i \in \mathcal{C}_{\mathcal{DFS}}$ then $\widetilde{P}$ sends the two correct inputs z_i^0 and z_i^1.
  - If $g_i \notin \mathcal{C}_{\mathcal{DFS}}$ and neither of g_i's input gate belongs to $\mathcal{C}_{\mathcal{DFS}}$ then $\widetilde{P}$ sends two arbitrary z_i^0 and z_i^1 that are consistent with y_i, i.e. $g_i(z_i^0, z_i^1) = y_i$.
  - $g_i \notin \mathcal{C}_{\mathcal{DFS}}$ and one of g_i's input gates belongs to $\mathcal{C}_{\mathcal{DFS}}$, then $\widetilde{P}$ will send the correct wire known to him and another arbitrary value consistent with y_i as above.

Lemma 3 (Advantage of a DFS Prover). *In one repetition of the protocol above, a DFS prover with effort budget $\tilde{C}$ investment has probability of cheating $\tilde{p}_{DFS}$ bounded by*

$$\tilde{p}_{DFS} \leq 1 - 2^{-d_{DFS}}$$

The proof of Lemma 3 follows easily from the proof of the stand-alone rationality of our protocol (see Theorem 3).

If a DFS prover focuses on maximizing the depth of a computed subcircuit given a certain investment, *BFS* provers allot their resources to compute all subcircuits rooted at a certain height. A BFS prover with effort budget $\tilde{C}$ computes the value of all gates up to the maximal height possible d_{BFS}. Note that d_{BFS} is a function of the circuit $\mathcal{C}$ and of the effort $\tilde{C}$. Let $\overline{\mathcal{C}}_{BFS}$ be the collection of gates computed by the BFS prover. The interaction of a BFS prover with V throughout the protocol resembles that of the DFS prover outlined above:

- At the beginning of the protocol send an arbitrary output value y_1.
- During procedure $Round(i, g_i, y_i)$:
  - If $g_i \in \overline{\mathcal{C}}_{BFS}$ then $\widetilde{P}$ sends the two correct inputs z_i^0 and z_i^1.
  - If $g_i \notin \overline{\mathcal{C}}_{BFS}$ and neither of g_i's input gate belongs to $\overline{\mathcal{C}}_{BFS}$ then $\widetilde{P}$ sends two arbitrary z_i^0 and z_i^1 that are consistent with y_i, i.e. $g_i(z_i^0, z_i^1) = y_i$.
  - $g_i \notin \overline{\mathcal{C}}_{BFS}$ and both g_i's input gates belong to $\mathcal{C}_{\mathcal{DFS}}$, then $\widetilde{P}$ will send one of the correct wires known to him and another arbitrary value consistent with y_i as above.

As before, it is not hard to see that the probability of successful cheating by a BFS prover can be bounded as follows:

Lemma 4 (Advantage of a BFS Prover). *In one repetition of the protocol above, a BFS prover with effort budget $\tilde{C}$ has probability of cheating $\tilde{p}_{BFS}$ bounded by*

$$\tilde{p} \leq 1 - 2^{-d_{BFS}}$$

BFS and DFS provers are both special cases of the general non-adaptive strategy which allots its investment $\tilde{C}$ among a general collection of subcircuits $\overline{\mathcal{C}}$. The interaction with V of such a prover is analogous to that of a BFS/DFS prover but with a collection of computed subcircuits not constrained by any specific height. We now try to formally define what the success probability of such a prover is.

Definition 3 (Path Experiment). *Consider a circuit $\mathcal{C}$ and a collection $\overline{\mathcal{C}}$ of subcircuits of $\mathcal{C}$. Perform the following experiment: starting from the output gate, flip a unbiased coin and choose the "left" subcircuit or the "right" subcircuit at random with probability 1/2. Continue until the random path followed by the experiment reaches a computed gate in $\overline{\mathcal{C}}$. Let i be the height of this gate, which is the output of the experiment. Define with Π_i the probability that this experiment outputs i.*

The proof of the following Lemma is a generalization of the proof of security of our scheme. Once the "verification path" chosen by the Verifier enters a fully computed subcircuit at height i (which happens with probability $\Pi_i^{\overline{\mathcal{C}}}$), the probability of success of the Prover is bounded by $(1 - 2^{-i})$

Lemma 5 (Advantage of a Non Adaptive Prover). *In one repetition of the protocol above, a generic prover with effort budget $\tilde{C}$ used to compute a collection $\overline{\mathcal{C}}$ of subcircuits, has probability of cheating $\tilde{p}_{\overline{\mathcal{C}}}$ bounded by*

$$\tilde{p}_{\overline{\mathcal{C}}} \leq \sum_{i=0}^{d} \Pi_i (1 - 2^{-i})$$

where Π_i-s are defined as in Definition 3.

Limiting the Class of Functions: Regular Circuits. Lemma 5 still does not produce a clear bound on the probability of success of a cheating prover. The reason is that it is not obvious how to bound the probabilities $\Pi_i^{\overline{\mathcal{C}}}$ that arise from the computed subcircuits $\overline{\mathcal{C}}$ since those depends in non-trivial ways from the topology of the circuit $\mathcal{C}$.

We now present a type of circuits for which it can be shown that the BFS strategy is optimal. The restriction on the circuit is surprisingly simple: we call them regular circuits. In the next section we show examples of interesting functions that admit regular circuits.

Definition 4 (Regular Circuit). *A circuit $\mathcal{C}$ is said to be regular if the following conditions hold:*

- *$\mathcal{C}$ is layered;*
- *every gate has fan-in 2;*
- *the inputs of every gate are the outputs of two distinct gates.*

The following lemma states that, in regular circuits, we can bound the advantage of any prover investing $\tilde{C}$ by looking at the advantage of a BFS prover with the same investment.

Lemma 6 (A Bound for Provers' Advantage in Regular Circuits). *Let $\widetilde{P}$ be a prover investing $\tilde{C}$. Let $\mathcal{C}$ be the circuit being computed and $\delta = d_{BFS}(\mathcal{C}, \tilde{C})$. In one repetition of the protocol above, the advantage of $\widetilde{P}$ is bounded by*

$$\tilde{p} \leq \tilde{p}_{BFS} = 1 - 2^{-\delta}$$

Proof. Let $\overline{\mathcal{C}}$ be the family of subcircuits computed by $\widetilde{P}$ with effort $\tilde{C}$. As pointed out above the probability of success for $\widetilde{P}$ is

$$\tilde{p} \leq \sum_{i=0}^{d} \Pi_i^{\overline{\mathcal{C}}}(1 - 2^{-i})$$

Consider now a prover $\widetilde{P}'$ which uses $\tilde{C}$ effort to compute a different collection of subcircuits $\overline{\mathcal{C}}'$ defined as follows:

- Remove a gate from a subcircuit of height j in $\overline{\mathcal{C}}$: this produces two subcircuits of height $j-1$. This is true because of the regularity of the circuit: since the inputs of every gate are the outputs of two distinct gates, when removing a gate of height j this will produce two subcircuits of height $j-1$;
- Use that computation to "join" two subcircuits of height k into a single subcircuit of height $k+1$. Again we are using the regularity of the circuit here: since the circuit is layered, the only way to join two subcircuits into a single computed subcircuit is to take two subcircuits of the same height.

What happens to the probability $\tilde{p}'$ of success of $\widetilde{P}'$? Let ℓ be the number of possible paths generated by the experiment above with $\overline{\mathcal{C}}$. Then the probability of entering a computed subcircuit at height j decreases by $1/\ell$ and that probability weight goes to entering at height $j-1$. Similarly the probability of entering at height k goes down by $2/\ell$ and that probability weight is shifted to entering at height $k+1$. Therefore

$$\begin{aligned}\tilde{p}' \leq & \sum_{i \neq j, j-1, k, k+1} \Pi_i(1 - 2^{-i}) \\ & + (\Pi_j - \frac{1}{\ell})(1 - 2^{-j}) + (\Pi_{j-1} + \frac{1}{\ell})(1 - 2^{-j+1}) \\ & + (\Pi_k - \frac{2}{\ell})(1 - 2^{-k}) + (\Pi_{k+1} + \frac{2}{\ell})(1 - 2^{-k-1})\end{aligned}$$

$$= \tilde{p} + \frac{1}{\ell 2^j} - \frac{1}{\ell 2^{j-1}} + \frac{1}{\ell 2^{k-1}} - \frac{1}{\ell 2^k}$$
$$= \tilde{p} + \frac{2^k - 2^{k+1} + 2^{j+1} - 2^j}{\ell 2^{j+k}} = \tilde{p} + \frac{2^j - 2^k}{\ell 2^{j+k}}$$

Note that $\tilde{p}'$ increases if $j > k$ which means that it's better to take "computation" away from tall computed subcircuits to make them shorter, and use the saved computation to increase the height of shorter computed subtrees, and therefore that the probability is maximized when all the subtrees are of the same height, i.e. by the BFS strategy which has probability of success $\tilde{p}_{BFS} = 1 - 2^{-\delta}$.

The above Lemma, therefore, yields the following.

Theorem 4. *Let $\mathcal{C}$ be a regular circuit of size C. If the (C, ϵ) Unique Inner State Assumption (see Assumption 1) holds for the function f computed by $\mathcal{C}$, and input distribution $\mathcal{D}$, then the protocol presented above with r repetitions is a $k\epsilon R$-sequentially composable Rational Proof for $\mathcal{C}$ for $\mathcal{D}$ if the prover follows a non-adaptive strategy and the following inequality holds for all $\tilde{C}$*

$$(1 - 2^{-\delta})^r \leq \frac{\tilde{C}}{C}$$

where $\delta = d_{BFS}(\mathcal{C}, \tilde{C})$.

Proof. Let $\gamma = \frac{\tilde{C}}{C}$. Consider $x \in \mathcal{D}$ and prover $\widetilde{P}$ which invests effort $\tilde{C} \leq C$. Under Assumption 1, $\widetilde{P}$ gives the correct outputs with probability $\gamma + \epsilon$ – assume that in this case $\widetilde{P}$ collects the reward R.

If $\widetilde{P}$ gives an incorrect output we can invoke Lemma 6 and conclude that he collects reward R with probability at most $(1 - 2^{-\delta})^r$ which by hypothesis is less than γ. So either way we have that $\tilde{R} \leq (\gamma + \epsilon)R$ and therefore applying Lemma 1 concludes the proof.

6 Results for FFT Circuits

In this section we apply the previous results to the problem of computing FFT circuits, and by extension to polynomial evaluations.

6.1 FFT Circuit for Computing a Single Coefficient

The Fast Fourier Transform is an almost ubiquitous computational problem that appears in many applications, including many of the volunteer computations that motivated our work. As described in [4] a circuit to compute the FFT of a vector of n input elements, consists of $\log n$ levels, where each level comprises $n/2$ *butterflies* gates. The output of the circuit is also a vector of n input elements.

Let us focus on the circuit that computes a single element of the output vector: it has $\log n$ levels, and at level i it has $n/2^i$ butterflies gates. Moreover the circuit is regular, according to Definition 4.

Theorem 5. *Under the (C, ϵ)-unique inner state assumption for input distribution $\mathcal{D}$, the protocol in Sect. 5, when repeated $r = O(1)$ times, yields sequentially composable rational proofs for the FFT, under input distribution $\mathcal{D}$ and assuming non-adaptive prover strategies.*

Proof. Since the circuit is regular we can prove sequential composability by invoking Theorem 4 and proving that for $r = O(1)$, the following inequality holds

$$\tilde{p} = (1 - 2^{-\delta})^r \leq \frac{\tilde{C}}{C}$$

where $\delta = d_{BFS}(\mathcal{C}, \tilde{C})$.

But for any $\tilde{\delta} < d$, the structure of the FFT circuit implies that the number of gates below height $\tilde{\delta}$ is $\tilde{C}_{\tilde{\delta}} = \Theta(C(1 - 2^{-\tilde{\delta}}))$. Thus the inequality above can be satisfied with $r = \Theta(1)$.

6.2 Mixed Strategies for Verification

One of the typical uses of the FFT is to change representation for polynomials. Given a polynomial $P(x)$ of degree $n - 1$ we can represent it as a vector of n coefficients $[a_0, \ldots, a_{n-1}]$ or as a vector of n points $[P(\omega_0), \ldots, P(\omega_{n-1})]$. If ω_i are the complext n-root of unity, the FFT is the algorithm that goes from one representation to the other in $O(n \log n)$ time, rather than the obvious $O(n^2)$.

In this section we consider the following problem: given two polynomial P, Q of degree $n - 1$ in point representation, compute the inner product of the coefficients of P, Q. A fan-in two circuit computing this function could be built as follows:

- two parallel FFT subcircuits computing the coefficient representation of P, Q ($\log n$-depth and $n \log n$) size total for the 2 circuits);
- a subcircuit where at the first level the i-degree coefficients are multiplied with each other, and then all these products are added by a binary tree of additions $O(\log n)$-depth and $O(n)$ size);

Note that this circuit is regular, and has depth $2 \log n + 1$ and size $n \log n + n + 1$.

Consider a prover $\widetilde{P}$ who pays $\tilde{C} < n \log n$ effort. Then, since the BFS strategy is optimal, the probability of convincing the Verifier of a wrong result of the FFT is $(1 - 2^{-\tilde{d}})^r$ where $\tilde{d} = c \log n$ with $c \leq 1$. Note also that $\frac{\tilde{C}}{C} < 1$. Therefore with $r = O(n^c)$ repetitions, the probability of success can be made smaller than $\frac{\tilde{C}}{C}$. The Verifier's complexity is $O(n^c \log n) = o(n \log n)$.

If $\tilde{C} \geq n \log n$ then the analysis above fails since $\tilde{d} > \log n$. Here we observe that in order for $\widetilde{P}$ to earn a larger reward than P, it must be that P has run at least $k = O(\log n)$ executions (since it is possible to find $k + 1$ inputs such that $(k + 1)\tilde{C} \leq kC$ only if $k > \log n$).

Assume for a moment that the prover always executes the same strategy with the same running time. In this case we can use a "mixed" strategy for verification:

- The Verifier pays the Prover only after k executions. Each execution is verified as above (with n^c repetitions);
- Additionally the Verifier uses the "check by re-execution" (from Sect. 4.4) strategy every k executions (verifiying one execution by recomputing it);
- The Verifier pays R if all the checks are satisfied, 0 otherwise;
- The Verifier's complexity is $O(kn^c \log n + n \log n) = o(kn \log n)$ – the latter being the complexity of computing k instances.

Notice that there are many plausible ways to assume that the expected cost $\tilde{C}$ remains the same through the $k+1$ proofs, for example by assuming that the Prover can be "resetted" at the beginning of each execution and made oblivious of the previous interactions.

7 Conclusion

Rational Proofs are a promising approach to the problem of verifying computations in a rational model, where the prover is not malicious, but only motivated by the goal of maximizing its utility function. We showed that Rational Proofs do not satisfy basic compositional properties in the case where a large number of "computation problems" are outsourced, e.g. volunteered computations. We showed that a "fast" incorrect answer is more remunerable for the prover, by allowing him to solve more problems and collect more rewards. We presented an enhanced definition of Rational Proofs that removes the economic incentive for this strategy and we presented a protocol that achieves it for some uniform bounded-depth circuits.

One thing to point out is that our protocol has two additional advantages:

- the honest Prover is always guaranteed a fixed reward R, as opposed to some of the protocols in [1,2] where the reward is a random variable even for the honest prover;
- Our protocol is the first example of a rational proof for arithmetic circuits.

Our work leaves many interesting research directions to explore:

- Is it possible to come up with a protocol that works for any bounded-depth circuit, and not circuits with special "topological" conditions such as the ones imposed by our results?
- Our results hold for "non-adaptive" prover strategies, though that seems more a proof artifact to simplify the analysis, than a technical requirement. Is it possible to lift that restriction?
- Are there other circuits which, like the FFT one, satisfy our notions and requirements?
- What about rational proofs for arbitrary poly-time computations? Even if the simpler stand-alone case?

References

1. Azar, P.D., Micali, S.: Rational proofs. In: 2012 ACM Symposium on Theory of Computing, pp. 1017–1028 (2012)
2. Azar, P.D., Micali, S.: Super-efficient rational proofs. In: 2013 ACM Conference on Electronic Commerce, pp. 29–30 (2013)
3. Belenkiy, M., Chase, M., Erway, C.C., Jannotti, J., Küpçü, A., Lysyanskaya, A.: Incentivizing outsourced computation. In: NetEcon 2008, pp. 85–90 (2008)
4. Cormen, T., Leiserson, C., Rivest, R., Stein, C.: Introduction to Algorithms. MIT Press (2001)
5. Dwork, C., Naor, M., Sahai, A.: Concurrent zero-knowledge. J. ACM **51**(6), 851–898 (2004)
6. Guo, S., Hubacek, P., Rosen, A., Vald, M.: Rational arguments: single round delegation with sublinear verification. In: 2014 Innovations in Theoretical Computer Science Conference (2014)
7. Goldwasser, S., Micali, S., Rackoff, C.: The knowledge complexity of interactive proof-systems. In: Proceedings of the seventeenth Annual ACM Symposium on Theory of computing. ACM (1985)
8. Walfish, M., Blumberg, A.J.: Verifying computations without reexecuting them. Commun. ACM **58**(2), 74–84 (2015)