How to Delegate Computations Publicly*

Yael Tauman Kalai † Omer Paneth ‡ Lisa Yang §

May 30, 2019

Abstract

We construct a delegation scheme for all polynomial time computations. Our scheme is publicly verifiable and completely non-interactive in the common reference string (CRS) model.

Our scheme is based on an efficiently falsifiable decisional assumption on groups with bilinear maps. Prior to this work, publicly verifiable non-interactive delegation schemes were only known under knowledge assumptions (or in the Random Oracle model) or under non-standard assumptions related to obfuscation or multilinear maps.

We obtain our result in two steps. First, we construct a scheme with a long CRS (polynomial in the running time of the computation) by following the blueprint of Paneth and Rothblum (TCC 2017). Then we bootstrap this scheme to obtain a short CRS. Our bootstrapping theorem exploits the fact that our scheme can securely delegate certain non-deterministic computations.

*An earlier version of this work which includes a different set of results appears on the ePrint Archive `https://eprint.iacr.org/2018/776`.

†Microsoft Research, email `yael@microsoft.com`.

‡MIT, email `omerpa@mit.edu`. Supported by NSF Grants CNS-1413964, CNS-1350619 and CNS-1414119, and the Defense Advanced Research Projects Agency (DARPA) and the U.S. Army Research Office under contracts W911NF-15-C-0226 and W911NF-15-C-0236.

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

Contents

1 Introduction

The theory of computation has been shaped by the study of proof systems that allow a powerful, yet untrusted prover, to convince a weak verifier of the validity of a computational statement. Examples include NP proofs [Coo71, Kar72], single-prover and multi-prover interactive proofs [GMR88, BGKW88], probabilistically checkable proofs [FGL+91, BFLS91, AS92, ALM+98], and zero-knowledge proofs [GMR88].

In this work we focus on the following setting motivated by the problem of securely outsourcing computation. A verifier wishes to evaluate a program M (represented, for example, as a Turing machine) on an input x but it is either not capable or not willing to spend the computational resources required to evaluate $M(x)$. Instead, the verifier delegates this computation to an untrusted prover that provides the verifier with the output $y = M(x)$ together with a proof Π convincing the verifier that the output is indeed correct. Importantly, verifying this proof should be much easier than evaluating $M(x)$. Additionally, the resources required to generate the proof should not be much greater than the resources required to perform the computation. In the literature, such systems are often referred to as doubly-efficient proofs or proofs for delegating computation.

Classic results on interactive proofs give rise to interactive proof systems with very efficient verification [Sha92, DFK+92, GKR08, Kil92]. However, in many settings, it is crucial that proofs are non-interactive and consist of a single message from the prover to the verifier. It is known that, under standard complexity theoretic assumptions, non-interactive delegation schemes require both computational assumptions and a common reference string [Sha92, GH98]. The common reference string (CRS) is generated once and can subsequently be used to generate and verify proofs. While for every CRS, there exist accepting proofs for false statements, it should be computationally infeasible to find such proofs for an honestly generated CRS.

Another key property we often require is public verifiability: anyone should be able to verify the proof, and no secret information, such as a trapdoor on the CRS, is needed for verification. Delegation schemes that are both non-interactive and publicly verifiable are particularly useful in applications: they let us compute short certificates of correctness for complex computations that can be easily verified by anyone at anytime.

In the literature, many delegation schemes have been proposed offering different tradeoffs between security and functionality. These schemes can be roughly divided into three groups:

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

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

Designated verifier schemes. A line of works starting from [KRR13, KRR14] and continuing with [KP16, BHK17, BKK+18] designed delegation schemes based on standard assumptions (such as computational private-information retrieval). These schemes, however, are not publicly verifiable. The CRS is generated together with a secret verification key required to verify the proof. Moreover, the CRS is not reusable: an adversary that is able to learn whether its proofs are accepted or not can eventually break soundness.

Interactive schemes. In the interactive setting, we can achieve publicly verifiable schemes under standard assumptions, and even unconditionally. For example, [GKR15] and [RRR16] give interactive delegation schemes for bounded depth and bounded space computations with unconditional soundness. The work

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

of [Kil92] gives a four message protocol from collision-resistant hashing, and [PRV12] uses attribute-based encryption to delegate low-depth circuits using two messages in addition to a (hard to compute) CRS.

We therefore ask:

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

1.1 Our Result

We construct a publicly verifiable non-interactive delegation scheme for all polynomial time deterministic computations. Security is proven under a new decisional assumption on groups with bilinear pairings. This assumption is efficiently falsifiable and holds in the generic group model. Our assumption is over a group G of prime order p equipped with a bilinear map:

Assumption 1.1. *For every $\alpha(\kappa) = O(\log \kappa)$, given the following 3-by-α matrix of group elements:*

$$\left(g^{s^j t^i}\right)_{\substack{i\in[0,2]\\ j\in[0,\alpha]}} = \begin{pmatrix} g^{s^0} & g^{s^1} & \dots & g^{s^\alpha} \\ g^{s^0 t} & g^{s^1 t} & \dots & g^{s^\alpha t} \\ g^{s^0 t^2} & g^{s^1 t^2} & \dots & g^{s^\alpha t^2} \end{pmatrix} ,$$

for random $g \in G$ and $s \in \mathbb{Z}_p$, it is computationally hard to distinguish between the case where $t = s^{2\alpha+1}$ and the case where t is a random independent element in $\mathbb{Z}_p$.

Under this assumption we construct a publicly verifiable delegation scheme for all polynomial time deterministic computations. The soundness of this scheme is adaptive: it holds even when the adversary can choose the computation as a function of the CRS.

Theorem 1.2 (Informal)**.** *For every constant $\epsilon > 0$ and polynomial $T = T(\kappa)$ there exists a publicly verifiable non-interactive delegation scheme with adaptive soundness for any time-T Turing machine under Assumption 1.1. The CRS and proof are of length T^ϵ, the prover runs in time $\mathrm{poly}(T, \kappa)$, and the verification run time is $n \cdot T^\epsilon$ where n is the input length.*

We emphasize that Theorem 1.2 only relies on a polynomial hardness assumption. We note, however, that in the soundness proof we reduce any attack on the delegation scheme with security parameter κ to an attack on Assumption 1.1 with a much smaller (but still polynomially related) parameter κ^δ for a constant δ that depends only on ϵ. More generally, we prove the following theorem.

Theorem 1.3 (Informal)**.** *For security parameter κ and every function $T = T(\kappa)$ there exists a publicly verifiable non-interactive delegation scheme with adaptive soundness for any time-T Turing machine under the T-hardness of Assumption 1.1 . The CRS and proof are of length $L = T^{O(1/\log_2 \log_\kappa T)}$, the prover's run time is $\mathrm{poly}(T, \kappa)$, and the verifier's run time is $O(L) + n \cdot \mathrm{poly}(\kappa)$ where n is the input length.*

By T-hardness of Assumption 1.1 we mean that the any adversary running in time $T^{O(1)}$ has distinguishing advantage $T^{-\omega(1)}$ and $\alpha = O(\log T)$. We derive Theorem 1.2 from Theorem 1.3 by setting $\kappa = T^\delta$ for a sufficiently small constant δ. By setting $\kappa = \mathrm{polylog}(T)$ we get a protocol with CRS and proof of length $T^{1/\log\log T}$ from sub-exponential hardness of Assumption 1.1.

Delegating non-deterministic computations. Beyond deterministic computation, previous work show how to delegate several sub-classes of **NP** in the designated-verifier setting under standard assumptions. The work of [BKK+18] give delegation for non-deterministic bounded-space computations, where the proof length grows with the space, from sub-exponential private information retrieval (PIR). Priorly, the works of [BHK17, BK18] give non-adaptive delegation for conjunctions and monotone formulas over **NP** statements, where the proof length grows with the length of a single witness, from polynomially secure PIR.

We construct publicly verifiable non-interactive delegation scheme for the same sub-classes under Assumption 1.1 instead of PIR. Unlike in our scheme for deterministic computation, here we use a long CRS (polynomial in the running time of the computation).

A note on earlier versions. An earlier version of this work [KPY18] constructed a publicly verifiable non-interactive delegation scheme with a long CRS for bounded depth computations, under a constant-size search assumption over groups with bilinear maps. The current version strengthens the previous one in two ways: first we show how to delegate arbitrary deterministic computations (as well as some non-deterministic ones), and second, our delegation scheme uses a short CRS. We rely on a decisional assumption over groups with bilinear maps.

Concurrent work. In a recent work, Canetti et al. [CCH+18] show a publicly verifiable non-interactive delegation scheme for log-space uniform NC computations, based on fully homomorphic encryption with very strong security, that they call optimal security. In contrast, our scheme supports all polynomial-time computation and security is based on a polynomial hardness assumption. One advantage of their scheme is that the CRS is just a random string, while in our scheme the CRS has more structure.

2 Technical Overview

Our delegation construction follows a template known as *bootstrapping* [Val08, BCCT13]. The idea is to first construct a delegation scheme where the length of the CRS is polynomial in the running time of the computation but verification is efficient (requiring only a small part of the CRS). This scheme is then bootstrapped to obtain a delegation scheme with a short CRS.

The work of [BCCT13] introduced a bootstrapping technique for strong proof systems known as SNARKs. A SNARK is a publicly verifiable succinct non-interactive argument of knowledge for proving **NP** statements. Succinctness means that the proof length and verification time are much smaller than the NP witness. SNARKs are only known under so-called knowledge assumptions and constructing them under falsifiable assumptions is subject to black-box impossibility results [GW11]. In contrast, our focus in this work is on constructing delegation schemes that only argue about computations in **P**, from falsifiable assumptions. Therefore, we follow a different route from the one in [BCCT13].

Our construction relies on a relaxation of SNARKs that we call *quasi-arguments*. Instead of the standard argument of knowledge requirement, our quasi-arguments only satisfy a weak soundness requirement. Using this notion, we prove Theorem 1.2 in two steps:

1. We construct a publicly verifiable succinct non-interactive quasi-argument for **NP** with a long CRS under Assumption 1.1.
2. We show how to bootstrap any such quasi-argument into a publicly verifiable delegation scheme for **P** with full soundness and a short CRS.

We start by describing the notion of quasi-arguments in Section 2.1. In Section 2.2 we give more details on the bootstrapping step, and in Section 2.3 we overview the quasi-argument construction.

2.1 Quasi-arguments.

Relaxing the notion of arguments of knowledge, in quasi-arguments the standard knowledge extraction requirement is replaced by a weaker requirement that we call *no-signaling extraction*. This notion was implicit in the work of [KRR14] and was formalized by [PR17] under the name local soundness.

The notion of no-signaling extraction is somewhat technical and it is designed to capture the soundness properties we are able to achieve under falsifiable assumptions. We therefore view this notion, not as a goal, but as a stepping stone towards full soundness. We start by recalling the motivation behind this relaxation. For simplicity, we focus first on the non-adaptive setting, and then discuss the adaptive definition.

A proof system for **NP** is an argument of knowledge if for every computationally bounded (possibly cheating) prover, there exists a computationally bounded extractor E such that if the prover convinces the verifier to accept a statement x with noticeable probability, then E extracts a witness w for the validity of x. In a succinct non-interactive argument, however, extracting a witness is extremely challenging. Intuitively,

since the prover's message is much shorter than the witness, we cannot hope to extract a witness from a single proof. The extractor may try to feed the prover with multiple CRS's, and reconstruct a witness from the prover's responses. However, the prover may compute each proof using a different witness.[2] Another approach is to extract a witness directly from the code of the prover. In all existing solutions, such non-black-box extraction is enabled by non-falsifiable knowledge assumptions.

In the weaker notion of no-signaling extraction we only require that there exists a local extractor L that, roughly speaking, extracts small parts of a witness. In more detail, the local extractor L takes as input a set $S \subseteq [|w|]$ of size at most K, and outputs a partial witness $w_S : S \to \{0,1\}$ mapping every position in S to a bit. The bound K on the size of S is called the locality parameter and it is typically set to be smaller than the proof length. We allow L to be probabilistic and produce different partial witnesses in different executions. Importantly, these partial witnesses need not be consistent with each other. We continue to describe the properties the extractor L must satisfy.

Local consistency. It is natural to require that $\mathsf{L}(S)$ always outputs a partial witness w_S that is consistent with some global witness. That is, there exists a valid witness w such that $w_S = w|_S$. However, this turns out to be a very strong requirement that we will not be able to satisfy. Instead we require that, intuitively, the partial witness w_S satisfies all the local constraints defined by the statement x. To make this concrete we first need to fix a particular **NP**-complete language. We use the language defined by a 3CNF formula φ (or rather an ensemble of formulas, one for each input length) such that an instance x is in the language if and only if the formula $\varphi_x = \varphi(x, \cdot)$ is satisfiable. Note that in a succinct quasi-argument for φ, the verification time is proportional to $|x|$ but much smaller than $|\varphi|$.

Given a subset S of the variables of φ_x, the local extractor $\mathsf{L}(S)$ outputs a partial witness w_S for x which is a partial assignment to the variables in S. We require that a partial assignment w_S sampled by $\mathsf{L}(S)$ *locally satisfies* φ_x with all but negligible probability. That is, for every clause in φ_x, if S contains all three variables of the clause, then the assignment to these variables satisfies the clause.

No-signaling. A partial witness w_S may locally satisfy φ_x but still be inconsistent with any full witness. In fact, if the locality parameter K is significantly smaller than $|\varphi|$, it may be easy to locally satisfy any subset of at most K variables, even when the formula φ_x is unsatisfiable. As a remedy, we put an additional *no-signaling* requirement on the local extractor.[3] Let U and V be disjoint sets of variables such that $S = U \cup V$ is of size at most K and let w_S be a partial witness sampled from $\mathsf{L}(S)$. The values w_S assigns to variables in U may depend on the entire input set S (including on the variables in V) as well as on L's random coins. However, no-signaling requires that the values w_S assigns to the variables of U give almost no information about the input set V. That is, the partial assignment w_S restricted to variables in U is computationally indistinguishable from a partial assignment w_U sampled from $\mathsf{L}(U)$.

Adaptive no-signaling extraction. We also consider the adaptive setting where the prover can choose the instance x adaptively based on the CRS. In this setting we let the no-signaling extractor L output, in addition to a local witness, the instance x. We require that if the prover convinces the verifier to accept with non-negligible probability, then, for any set S, the statement sampled by $\mathsf{L}(S)$ is computationally indistinguishable from the statement sampled by the prover, conditioned on it producing an accepting proof.

The no-signaling condition must also be modified accordingly: for every disjoint sets U, V such that $S = U \cup V$ is of size at most K, and for an instance x and a partial assignment w_S sampled from $\mathsf{L}(S)$, we require that the instance x and the values w_S assigned to the variables of U together give almost no information about the set V.

[2] In the adaptive setting, the prover may even choose a different statement for each CRS.

[3] No-signaling strategies were first studied in physics in the context of Bell inequalities by Khalfin and Tsirelson [KT85] and Rastall [Ras85]. More recently, they were extensively studied in the context of multi-prover interactive proofs (see [KRR14] and references therein). Similarly to the work of [BHK17] the no-signaling notion used in this work is a computational one.

From no-signaling extraction to soundness. It is easy to see that any argument of knowledge is also a quasi-argument. In the converse direction, however, a no-signaling extractor with locality $K << |\varphi|$ is unlikely to imply the standard notion of extraction or even soundness for a general formula φ (see discussion in [KRR14]). Still, it turns out that for some formulas such an implication does hold. For example, [KRR14, PR17, BHK17] show that for every *deterministic* time-T Turing machine $\mathcal{M}$, there exists a formula φ of size $\text{poly}(T)$ such that $\mathcal{M}$ accepts x if and only if φ_x is satisfiable, and moreover, φ_x is satisfiable if and only if there exists a no-signaling extractor L for φ_x with locality $K = \text{polylog}(T)$. The work of [KRR14, BHK17] used this fact to turn a succinct non-interactive quasi-argument for **NP** into a designated-verifier delegation for **P** with adaptive soundness. Beyond deterministic computations, quasi-arguments were used to construct designated-verifier delegation schemes for interesting sub-classes of **NP** [KP15, BHK17, BKK$^+$18, BK18]. Following the same blueprint, the work of [PR17] constructs a publicly verifiable quasi-argument for **NP** from multilinear maps and turns it into a publicly verifiable delegation scheme.

Outline. We prove our main result given by Theorem 1.2 in two steps. First we construct publicly verifiable succinct non-interactive adaptive quasi-argument for **NP** with a long CRS.

Theorem 2.1 (Informal)**.** *For security parameter κ, 3CNF formula φ of polynomial size $T = T(\kappa)$ and locality parameter $K = K(\kappa) \leq T(\kappa)$ there exists a publicly verifiable succinct non-interactive quasi-argument for φ with adaptive no-signaling extraction under Assumption 1.1. The CRS is of length* $\text{poly}(\kappa, K, T)$ *and the proof is of length* $L = \text{poly}(\kappa, K)$*. The verification time is* $O(L) + n \cdot \text{poly}(\kappa)$ *where n is the input length.*

In combination with previous work [KP15, BHK17, BKK$^+$18, BK18], Theorem 2.1 (or, in some case, a sub-exponential version of it) already gives delegation for deterministic Turing machines, RAM machines and several sub-classes of **NP** computations. This is done by replacing the designated-verifier quasi-argument used in these previous work with our publicly verifiable one.

Going beyond a long CRS, our second step shows how to bootstrap any such quasi-argument to a publicly verifiable non-interactive delegation scheme for **P** with a short CRS.

Theorem 2.2 (Informal)**.** *Assuming a collision-resistant hash family and a quasi-argument for **NP** with a long CRS as in Theorem 2.1, there exists a delegation scheme for **P** with a short CRS as in Theorem 1.2.*

Since Assumption 1.1 already implies collision-resistant hashing, Theorem 1.2 follows from these two theorems. We elaborate on the proofs of Theorems 2.1 and 2.2 in Sections 2.3 and 2.2 respectively.

2.2 The Bootstrapping Theorem.

To explain our bootstrapping technique, in this section we focus on a simplified version of Theorem 2.2 that only gives delegation for *low-space* computations. We say that a Turing machine is low-space if the size of its input n, and the size of its work tapes are bounded by $\text{poly}(\kappa)$ where κ is the security parameter of the delegation scheme. In particular, in such a delegation scheme, the verification time exceeds the space. In Section 2.2.3 we explain how to extend this construction to support arbitrary computations based on techniques for delegating RAM computations [KP16, BHK17].

In what follows, we give an overview of the proof of Theorem 2.2 for polynomial-time low-space computations. The proof is by induction. In the base of the induction we construct a delegation scheme with a long CRS. In the inductive step we turn a delegation scheme into a new delegation scheme with a shorter CRS. Both steps rely on the quasi-argument for **NP** given in Theorem 2.1.

Theorem 2.3 (Base case, informal)**.** *Assuming a quasi-argument for **NP** with a long CRS as in Theorem 2.1, there exists a publicly verifiable non-interactive delegation scheme for time-T low-space Turing machines with adaptive soundness, CRS of length* $\text{poly}(\kappa, T)$*, and verification time* $\text{poly}(\kappa)$*.*

We give an overview of the proof of Theorem 2.3 in Section 2.2.1. In the inductive step, roughly speaking, we can shrink the CRS by a factor B of our choice, collecting only an additive $\text{poly}(B)$ term. Each step also increases the size of the proof polynomially.

Theorem 2.4 (Inductive step, informal)**.** *Assume there exists a quasi-argument for* ***NP*** *with a long CRS as in Theorem 2.1, and a publicly verifiable non-interactive delegation scheme for time-T low-space Turing machines with adaptive soundness, CRS of length $L_1(\kappa, T)$, and verification time $L_2(\kappa)$. Then for every polynomial $B = B(\kappa)$ there exists a publicly verifiable non-interactive delegation scheme for time-T low-space Turing machines with adaptive soundness, CRS of length $L'_1(\kappa, T)$, and verification time $L'_2(\kappa)$ for:*

$$L'_1(\kappa, T) = L_1(\kappa, T/B) + \mathrm{poly}(B, L_2(\kappa)) \quad , \quad L'_2(\kappa) = \mathrm{poly}(L_2(\kappa)) \ .$$

For any d, starting with the base delegation scheme in Theorem 2.3, after d applications of Theorem 2.4 with $B = T^{1/d}$ we get a publicly verifiable non-interactive delegation scheme for time-T Turing machine with adaptive soundness, CRS of length $T^{O(1/d)} \cdot \kappa^{2^{O(d)}}$, and verification time $\kappa^{2^{O(d)}}$. Setting d that balances the CRS and proof lengths we get a delegation scheme with parameters as in the statement of Theorem 1.3 (for a polynomial-time low-space computations). We give an overview of the proof of Theorem 2.4 in Section 2.2.1.

2.2.1 The base case.

We start with the base of the induction given in Theorem 2.3. This step follows techniques introduced in previous works [KRR14, PR17, BHK17]. We describe these techniques in detail since we use them also in the inductive step.

We construct a publicly verifiable non-interactive delegation scheme for a time-T low-space Turing machine $\mathcal{M}$ with adaptive soundness, CRS of length $\mathrm{poly}(\kappa, T)$, and verification time $\mathrm{poly}(\kappa)$. The construction relies on a quasi-argument for **NP** with a long CRS as in Theorem 2.1. The first step is to translate $\mathcal{M}$ into a 3CNF formula φ of size $\mathrm{poly}(T)$ such that:

1. If $\mathcal{M}$ accepts an input $x \in \{0,1\}^n$ then $\varphi_x = \varphi(x, \cdot)$ has a satisfying assignment that can be computed from x in time $\mathrm{poly}(T)$.
2. For any computationally bounded adaptive no-signaling extractor L for φ with locality parameter $K = O(\kappa)$ and any set S of at most K variables, the probability that $\mathsf{L}(S)$ samples a rejecting input x is negligible.

We defer the description of the formula φ with the above two required properties to later, and first show how such φ is used to construct a delegation scheme for $\mathcal{M}$.

The delegation scheme. Our delegation scheme for $\mathcal{M}$ simply invokes the quasi-argument for φ. In more detail, we generate a CRS for the quasi-argument of length $\mathrm{poly}(\kappa, K, |\varphi|) = \mathrm{poly}(\kappa, T)$. The prover, given an input x of length $\mathrm{poly}(\kappa)$, computes a satisfying assignment for φ_x, follows the strategy of the quasi-argument's prover and outputs a proof of length $\mathrm{poly}(\kappa, K) = \mathrm{poly}(\kappa)$. The verifier follows the strategy of the quasi-argument's verifier running in time $\mathrm{poly}(\kappa)$.

For soundness, consider a computationally bounded adaptive prover for the delegation scheme that convinces the verifier to accept with non-negligible probability. The quasi-argument guarantees a computationally bounded adaptive no-signaling extractor L such that for every set S of at most K variables, $\mathsf{L}(S)$ samples an input x that is computationally indistinguishable from the prover's input, conditioned on the prover producing an accepting proof. By Property 2 of φ, the probability $\mathsf{L}(S)$ samples a rejecting input x is negligible. Since we can decide if x is accepting or rejecting efficiently (in time $\mathrm{poly}(T)$), it follows that the prover can only produce an accepting proof for a rejecting x with negligible probability.

The formula φ. The construction of φ follows the standard Cook-Levin reduction. For every $i \in [T]$, φ contains a set S_i of variables that encode the state of $\mathcal{M}$ and all its tapes after exactly i steps. The formula φ is satisfied if and only if:

1. The initial state and tapes encoded by S_1 are consistent with the input x.
2. The state and tapes encoded by S_i are consistent with these encoded by S_{i+1}.

3. The final state encoded by S_T is accepting.

It is easy to verify that φ is of size poly(T) and if $\mathcal{M}$ accepts x we can compute from x in time poly(T) a satisfying assignment w_x for φ_x. We refer to w_x as the *correct* assignment. Note that even for rejecting inputs we can naturally define the correct assignment w_x that satisfies all the clauses of φ_x except for the clauses checking that the final state encoded by S_T is accepting.

It remains to show that φ satisfies Property 2: for any adaptive no-signaling extractor L for φ with a sufficiently large locality parameter $K = \text{poly}(\kappa)$ and any set S, the probability that $\mathsf{L}(S)$ samples a rejecting input is negligible. Very roughly, the high-level proof strategy is as follows. For every i, use L to sample an input x and a partial assignment w_i to the variables in $S_i \cup S_{i+1}$. Then argue that for each i, the partial assignment w_i must be consistent with the correct assignment w_x (with all but negligible probability). This is proved by induction on i. For $i = 1$ this follows directly from the local consistency property of L. For $i > 1$, local consistency implies that if the partial assignment w_i is correct on S_i then it must also be correct on S_{i+1}. Then, using the no-signaling property of L, one can argue that the partial assignment w_{i+1} must also be correct on S_{i+1}. Finally, if the last partial assignment to the last set S_T is correct then the input x must be accepting. Throughout this argument, when invoking no-signaling, we crucially rely on the fact that given x and a partial assignment w_S, one can efficiently decide if w_S is correct. This holds because φ encodes a deterministic computation and, therefore, the correct assignment w_x is efficiently computable.

Arguing Property 2 in detail. For the sake of completeness, we give a detailed overview of the proof that φ satisfies Property 2 sketched above. To skip these details the reader can move directly to Section 2.2.2.

Fix an adaptive no-signaling extractor L for φ with locality parameter $K = \text{poly}(\kappa)$ such that $K \geq |S_i \cup S_{i+1}|$ for all i. For every set S of at most K variables we need to show that $\mathsf{L}(S)$ samples a rejecting input x with negligible probability. First, we observe that by no-signaling, for any two sets S and S' the inputs sampled by $\mathsf{L}(S)$ and by $\mathsf{L}(S')$ are computationally indistinguishable (they are both indistinguishable from the input sampled by $\mathsf{L}(\emptyset)$). Therefore, since we can efficiently decide if an input x is accepting or rejecting, it is sufficient to show that for *some* set S, $\mathsf{L}(S)$ samples a rejecting input only with negligible probability.

For every $i \in [T]$ consider an input x_i and a partial assignment $w_i : S_i \to \{0,1\}$ sampled by $\mathsf{L}(S_i)$. We say that the partial assignment w_i is correct if it encodes the same state and tapes as the correct assignment w_{x_i} on S_i. We will argue that for every $i \in [T]$, w_i is correct with all but negligible probability. Before proving this fact we use it to conclude the argument.

By definition, the final state encoded by the correct assignment w_{x_T} on S_T is accepting if and only if x_T is accepting. Additionally, if the partial assignment w_T locally satisfies φ_{x_T} then it must encode an accepting final state. Therefore, if w_T is both correct and locally satisfies φ_{x_T} then x_T must be accepting. By the local consistency of L, w_T must locally satisfy φ_{x_T} with all but negligible probability. Therefore, if $\mathsf{L}(S_T)$ samples w_T that is correct with all but negligible probability then it can only sample a rejecting x_T with negligible probability.

It remains to show that w_i is correct with all but negligible probability. We argue this inductively. For $i = 1$, $\mathsf{L}(S_1)$ outputs a partial assignment w_1 that locally satisfies φ_{x_1} with all but negligible probability. Since φ_{x_1} checks consistency of the initial state and x_1, it follows that w_1 is also correct with the same probability.[4] For $i < T$, assuming w_i is correct with all but negligible probability, we need to show that the same holds for w_{i+1}. Consider an input x and a partial assignment $w : S_i \cup S_{i+1} \to \{0,1\}$ sampled by $\mathsf{L}(S_i \cup S_{i+1})$. For $j \in \{i, i+1\}$ we say that w is j-correct if the partial assignment w restricted to S_j encodes the correct state and tapes as the correct assignment w_x on S_j. First, we argue that the probability that w is j-correct and the probability that w_j is correct are negligibly close. This follows from the no-signaling property of L and from the fact that given an input $x' \in \{x, x_j\}$ we can efficiently compute the correct assignment $w_{x'}$ and decide if a given partial assignment to S_j encodes the same state and tapes. It remains

[4]Here we assume that the clauses checking the consistency of the initial state only involve variables in S_1. In what follows we make similar assumptions on φ's structure.

to argue that if w is i-correct then it is also $(i+1)$-correct with all but negligible probability. This holds since w locally satisfies φ_x with all but negligible probability and since φ_x checks consistency of S_i and S_{i+1}.

2.2.2 The inductive step.

We continue to describe the inductive step given in Theorem 2.4. In the inductive step we compose the quasi-argument for **NP** with a delegation scheme to get a new delegation scheme with a shorter CRS.

We are given a publicly verifiable non-interactive delegation scheme for any time-T low-space Turing machine with adaptive soundness, CRS of length $L_1(\kappa, T)$, and verification time $L_2(\kappa)$. We refer to this delegation scheme as the original delegation scheme. We transform this original scheme into a new delegation scheme with CRS of length $L'_1(\kappa, T)$ and verification time $L'_2(\kappa)$, where:

$$L'_1(\kappa, T) = L_1(\kappa, T/B) + \text{poly}(B, L_2(\kappa)) \quad , \quad L'_2(\kappa) = \text{poly}(L_2(\kappa)) \ ,$$

for a parameter B of our choice. This construction again relies on a quasi-argument for **NP** with a long CRS as in Theorem 2.1.

The main idea is as follows. Recall that in the base case, we constructed a delegation scheme for a time-T low-space machine $\mathcal{M}$ by applying a quasi-argument to the formula φ. The variables of φ encode all the intermediate configurations of $\mathcal{M}$ (each configuration contains the machine's state and tapes) and the formula checks that every two consecutive configurations are consistent (as well as the validity of the first and last configurations). The CRS grows with the size of φ which is $\text{poly}(T)$. The verification time grows with the locality parameter K of the quasi-argument. To get soundness we need to set K proportional to the computation space, and therefore the the verification is $\text{poly}(\kappa)$.

To construct the new delegation scheme with a shorter CRS, we consider a new formula ϕ. The variables of ϕ only encode B of $\mathcal{M}$'s configurations at steps $i \cdot T/B$ for $i \in [B]$. Naively checking the consistency of two configuration that are T/B steps apart would require time $\text{poly}(T/B)$ and result in a formula ϕ that is again of size $\text{poly}(T)$. Instead, for each $i \in [B]$, ϕ also contains variables encoding a proof, under the original delegation scheme, asserting that $\mathcal{M}$ indeed transitions between the configurations at steps $i \cdot T/B$ and $(i+1) \cdot T/B$. The formula simply checks that all the proofs are accepting. To this end, the CRS of the new scheme includes a CRS of the original scheme for (T/B)-time computations and we hardwire into ϕ the part of this CRS required for verification.

Overall, the size of the formula ϕ is $O(B \cdot L_2(\kappa))$. Our new CRS contains both the CRS of the quasi-argument which is of length $\text{poly}(|\phi|) = \text{poly}(B \cdot L_2(\kappa))$ and the CRS of the original delegation scheme for (T/B)-time computations which is of length $L_1(\kappa, T/B)$. To get soundness we need to set the locality parameter K to be $O(L_2(\kappa))$ so it is higher than the number of variables required to encode two configurations of $\mathcal{M}$ as well as the original delegation proof that connects them. This results in a proof of length $\text{poly}(L_2(\kappa))$.

Soundness of the new scheme. The soundness proof for the new delegation scheme is very similar to the proof in the base case. The only modification is in proving that ϕ satisfies Property 2. Recall that in the proof of the base case we used the no-signaling extractor L to sample an input x and partial assignment w to variables describing two consecutive configurations. Since φ checks the consistency of the configurations, it follows that if w locally satisfies φ_x and the value assigned to the first configuration is correct, then the the value assignment to the next configuration must also be correct.

In the analysis of the new construction we consider a partial assignment w to the variables describing the two configurations in steps $i \cdot T/B$ and $(i+1) \cdot T/B$ as well as the original delegation proof asserting that $\mathcal{M}$ indeed transitions between these two configurations. Since φ checks that the proof is accepting, it follows that if w locally satisfies ϕ then it must contain an accepting proof. Note that since the original delegation scheme is only computationally sound, the value assigned to the first configuration may be correct while the value assignment to the next configuration is incorrect, and yet the proof is accepting. However, if this happens with non-negligible probability, we can turn the computationally bounded no-signaling extractor L into a prover breaking the adaptive soundness of the original delegation scheme.

2.2.3 Beyond low-space computations.

To delegate computations with arbitrary large space we rely on techniques from [KP16, BHK17] introduced for delegating RAM computations. The basic idea is to emulate a high-space machine $\mathcal{M}$ via a RAM machine $\mathcal{R}$ that has a small internal memory, and access to a large but untrusted external memory. To ensure the integrity of the external memory we use the classic notion of online memory checking based on hash trees [BEG+94]. Roughly, at every step, the external memory holds a hash tree of $\mathcal{M}$'s tapes, and its internal memory $\mathcal{R}$ saves $\mathcal{M}$'s state (including the positions of its heads) as well as the hash tree root, which is of size $O(\kappa)$. When $\mathcal{M}$ reads or writes, $\mathcal{R}$ accesses the external memory and obtains the values read, the updated root after writes, and a proof of size $\text{poly}(\kappa)$ authenticating these values against the current root.

Based on this idea, we strengthen Theorem 2.3 and Theorem 2.4 and get delegation for RAM computations instead of low-space computations. This immediately implies delegation for any large-space Turing machine as well. In a delegation scheme for RAM computations, the prover can convince the verifier that a RAM machine $\mathcal{R}$ transitions from configuration x (including $\mathcal{R}$'s state and external memory) to configuration y in T steps. The verifier only needs to hold a short hash or a digest of each configuration $(\mathsf{h}_x, \mathsf{h}_y)$.

It is natural to define adaptive soundness by requiring that a computationally bounded prover given the CRS cannot produce an accepting proof for a false statement $(\mathsf{h}_x, \mathsf{h}_y)$ with non-negligible probability. However, this notion is meaningless since the digests $\mathsf{h}_x, \mathsf{h}_y$ may correspond to exponentially many different configurations. Instead we require that such a prover cannot produce a configuration x, a digest h, and a proof such that with non-negligible probability:

- The verifier accepts the proof for the statement $(\mathsf{h}_x, \mathsf{h})$ where h_x is the digest of x.
- However, h is not the digest of the correct configuration y that $\mathcal{R}$ reaches from x within T steps.

Next we explain how to modify the proof of Theorem 2.3 (the induction's base) to get a delegation scheme for RAM computations with a long CRS. The proof of Theorem 2.4 is modified similarly. We consider a different formula φ whose variables encode the digests of $\mathcal{R}$'s intermediate configurations. Now, φ cannot directly check that two consecutive digests are consistent. Therefore, we add to φ variables encoding a hash-tree based proof authenticating the new digest under the old one. This is similar to the formula ϕ we used in the inductive step, except that here we use a hash-tree proof instead of a delegation proof to verify the consistency between each two consecutive steps.

We also need to augment the soundness proof as follows. Recall that in order to prove that φ satisfies Property 2, we crucially relied on the fact that we can efficiently compute the correct assignment for φ given only the input x. Now, the input consists only of the digests $(\mathsf{h}_x, \mathsf{h}_y)$ and cannot be used to compute the correct assignment. To overcome this, we rely on the fact that in order to break soundness a cheating prover must produce, in addition to the digests $(\mathsf{h}_x, \mathsf{h}_y)$, also the full initial configuration x. To make use of this fact, we extend our notion of quasi-argument to account for auxiliary input: if the adaptive prover produces an input $(\mathsf{h}_x, \mathsf{h}_y)$ together with some auxiliary input x (in our case the full initial configuration), then the no-signaling extractor samples both the input and auxiliary input (our quasi-argument satisfies this notion). In the proof, we can use the auxiliary input x to efficiently compute the correct assignment for φ.

On previous notions of RAM delegation. We note that our definition of RAM delegation is incomparable to that in [KP16, BHK17]. Previous works considered a weaker notion of adaptivity, but achieved a stronger notion of soundness for malicious digests. In more detail, in previous works, to break soundness, the cheating prover could produce an arbitrary initial digest h without producing the actual initial configuration x (a configuration with digest h may not even exist). The cheating prover only needs to produce accepting proofs for two statement $(\mathsf{h}, \mathsf{h}')$ and $(\mathsf{h}, \mathsf{h}'')$ where $\mathsf{h}' \neq \mathsf{h}''$. In contrast, in our notion of soundness for RAM delegation, the prover must output the initial configuration x. Since we use RAM delegation to get delegation for high-space computations, the adaptive cheating prover we consider anyways produces the computations's input x used as the initial configuration of the RAM computation.

As for adaptivity, in previous works, the initial digest must be fixed independently of the CRS, while we consider fully adaptive cheating provers. We note that we crucially rely on full adaptivity in the proof of Theorem 2.4.

2.3 The Quasi-argument Construction.

In this section we overview the proof of Theorem 2.1 constructing a publicly verifiable succinct non-interactive quasi-argument for **NP** with adaptive soundness and a long CRS under Assumption 1.1. Our construction follows the blueprint introduced in the work of Paneth and Rothblum (PR) [PR17] which gives a publicly verifiable quasi-argument with a short CRS based on multilinear maps. We show how to instantiate their blueprint based only on bilinear maps. The rest of this section is organized as follows. In Section 2.3.1 we describe the high-level idea of replacing multilinear with bilinear maps without going into the details of the PR quasi-argument. In Section 2.3.2 we describe the PR quasi-argument in detail and in Section 2.3.3 we provide more details on our quasi-argument from bilinear maps.

2.3.1 Replacing multilinear with bilinear maps.

The quasi-argument of PR is based on multilinear maps. Without going into the details of their construction, we explain how multilinear maps are used in their construction and give the high level idea of replacing them with bilinear maps at the price a longer CRS.

Very roughly, a degree-δ multilinear map lets us encode elements from a large field $\mathbb{F}$ such that we can homomorphically evaluate any polynomial of degree at most δ over these encodings, and test if the result encodes zero or not. However, we assume that evaluating computations of degree higher than δ is hard. We can think of bilinear maps as multilinear maps with degree 2. In the PR quasi-argument the CRS and the proof consists of elements encoded via a degree-δ multilinear map for $\delta > 2$. We denote the field elements encoded in the CRS and in the proof by $\{\alpha_i\}$ and $\{\beta_j\}$ respectively. The prover homomorphically computes each β_j as a polynomial in the α's and the verifier evaluates polynomials in the α's and β's together. Finally, the verifier tests if the result encodes zero or not.

Our idea is to replace the multilinear encodings with bilinear encodings. To allow for degree-δ homomorphic computations we add to the CRS the bilinear encodings of all possible monomials in the α's of degree at most δ. Similarly, we add to the proof the bilinear encoding of all possible monomials in the β's of degree at most δ. Now, the prover can evaluate any degree-δ polynomial in the α's by evaluating a linear function over the monomials encoded in the CRS. To evaluate a degree-δ polynomial in the α's and β's together, the verifier uses to bilinear map to evaluate a quadratic function over the α-monomials in CRS and the β-monomials in the proof.

We note that if we were to follow this approach naively, then the CRS and proof length in our quasi-argument would become super-polynomial. Instead, we carefully analyze the PR quasi-argument and only add the monomials that the prover and verifier actually use. This leads to a quasi-argument with CRS and proof length as in Theorem 2.1. In terms of security, we augment the analysis of PR and show how to reduce the security of our quasi-argument to Assumption 1.1. The analysis of the CRS and proof length of our quasi-argument, as well as the reduction to Assumption 1.1, are outlined in Section 2.3.3. However, we must first describe the PR quasi-argument in more detail.

2.3.2 The PR quasi-argument.

In this section we give a detailed overview of the quasi-argument constructed in PR. We start by describing works in the designated-verifier setting that led up to this quasi-argument.

Designated-verifier quasi-arguments. Implicit in the work of Kalai, Raz and Rothblum [KRR13] is a construction of a *designated-verifier* quasi-argument. This construction can be based on any computational private information retrieval scheme, however, for simplicity, we describe it using fully homomorphic encryption (FHE).

We start with the following naive attempt: for a 3CNF formula φ and locality parameter K, the CRS contains a random set of K variables of φ, where each variable name is encrypted under the FHE using a different key. Given an input x and a satisfying assignment w to φ_x, the prover homomorphically evaluates for every encrypted variable, the bit w assigns to it, and sends the encrypted bits to the verifier. Using the

verifier's secret state that contains all of the secret keys, it decrypts and obtains a partial assignment to the K variables encrypted in the CRS. The verifier accepts if this partial assignment locally satisfies φ_x.[5]

We attempt to demonstrate a no-signaling extractor L for this protocol (for simplicity, we focus on the non-adaptive setting). Given a set S of at most K variables, L generates a CRS by encrypting the variables in S, runs the prover with the CRS, obtains an encrypted partial assignment, and outputs the decrypted assignment. The fact that L satisfies the no-signaling condition follows, almost by definition, from the semantic security of the FHE. Intuitively, this is because the assignment to a variable encrypted under one key cannot signal information about the variable encrypted under another key. However, L may not satisfy local consistency. By definition, any accepted proof decrypts to a partial assignment that locally satisfies φ_x. However, even if the prover convinces the verifier to accept with high probability for a completely random CRS, it may not do so for a CRS encrypting a specific set S. For example, if the prover starts from a satisfying assignment and flips the value of one variable, it will only produce rejecting proofs for CRS's that encrypt the flipped variable.

One proposal to fix this naive protocol is based on an idea from [BMW99]. Roughly speaking, the idea is to encode φ with a probabilistically checkable proof (PCP) and encrypt random PCP queries in the CRS. Given a set S, the extractor L generates a CRS by encrypting random queries from which a local assignment to S can be locally decoded. The intuition is that, since each query is encrypted under a different key, the prover must answer each query independently. Then, using the soundness of the PCP, deduce that if the prover's answers to random queries are accepting, then L should decode a locally satisfying assignment for any set S. While this intuition turns out to be misleading [DLN$^+$04], the work of [KRR13] demonstrated a particular PCP for which the resulting protocol is in fact a quasi-argument.

Publicly verifiable quasi-arguments. The work of PR proposed another approach to constructing quasi-arguments in the publicly verifiable setting. Consider the naive protocol described above (without the PCP) and suppose there was a *public* verification procedure, that decides if to accept the prover's answer without using the secret keys. That is, suppose we could efficiently test if the prover's encrypted partial assignment locally satisfies φ_x or not. Yet, at the same time, semantic security of the CRS ciphertexts still holds. PR observe that if such a public verification procedure exists, then we can prove that the naive protocol is in fact a quasi-argument even without using PCP encoding of φ.

The idea is to use the public verification procedure to construct a no-signaling extractor L by modifying the flawed extractor described above. Given a set S of at most K variables, L samples a CRS by encrypting the variables in S, runs the prover with the CRS, and obtains an encrypted partial assignment. Now, instead of decrypting, the extractor uses the public verification procedure to check prover's answer. If the proof is rejected, L repeats the above steps with fresh randomness until an accepting proof is found. Only then L decrypts and outputs the partial assignment.

To show that L is indeed a no-signaling extractor, consider a prover that convinces the verifier to accept with non-negligible probability for an honestly generated CRS. Since verification is public, it follows from semantic security that the same holds for a CRS encrypting any set S. Therefore, L outputs a partial assignment that locally satisfies φ_x in expected polynomial time. Moreover, since the added verification step does not use the secret keys, the no-signaling property of L follows directly from the semantic security of the encryption.

To get public verification, PR modify the naive protocol and rely on a stronger notion of *zero-testable* homomorphic encryption (ZTHE) which they construct from multilinear maps. In what follows, we describe their approach.

Towards public verification. - Given a proof that consists of an encrypted partial assignment w, our goal is to publicly test if w locally satisfies φ_x without the secret keys. The main idea is to rely on the strong features of the ZTHE, to verify the proof in two steps:

[5]Here, we assume for simplicity that the verifier holds an implicit representation of φ that allows it to efficiently check if φ_x is locally satisfied. Our full protocol does not rely on this assumption and supports general formulas.

1. Test if w locally satisfies φ_x homomorphically, under the encryption.
2. Given the encrypted result, recover the result in the clear.

The first task requires a multi-key homomorphic encryption. Recall that every bit of w in the prover's answer is encrypted under a different secret key. Therefore, in order to homomorphically test if w locally satisfies φ_x we must compute on ciphertexts encrypted under different keys. Since φ is a 3CNF, it is sufficient to verify that w restricted to any three of its variables locally satisfies φ_x. Therefore, we only need a multi-key homomorphic encryption for three keys.

For the second task, we need a procedure that given the encrypted result (but not the secret key) tests if it encrypts zero or not. Since, in general, such a zero-test would render the encryption completely insecure, we compromise for a weak zero-test that can only recognize a particular subset of "trivial" ciphertexts encrypting zero. In more detail, the weak zero-test should satisfy the following completeness and soundness properties. Soundness says that the test fails on any ciphertext that does not decrypt to zero, even if the ciphertext is not generated honestly. Completeness says that the test passes on any trivial zero chipertext. A ciphertext is said to be trivial zero if it is computed from an honestly generated ciphertext c by homomorphically evaluating a circuit A that computes the all-zero function as an arithmetic circuit over $\mathbb{Z}$. We emphasize that the test can either pass or fail on ciphertexts that decrypt to zero but are not a trivial zero ciphertext. Note that such a weak zero test does not contradict semantic security since, intuitively, the test outcome only depends on the evaluated computation A and not on what is encrypted in c.

We note that in defining trivial zeros, we let A be identically zero over $\mathbb{Z}$ and not, for instance, over the binary field. Otherwise, we could use the ZTHE to efficiently decide satisfiability, as follows: given a boolean circuit $A : \{0,1\}^n \to \{0,1\}$, generate a ciphertext c encrypting 0^n, homomorphically evaluate A on c and zero-test the result. If A is not satisfiable, that is, if A computes the all zero function over the binary field, then the zero test is guaranteed to pass. Otherwise, there exists x such that $A(x) \neq 0$. If c was an encryption of x, by the soundness property, the zero-test would have failed. Therefore, the test must also fail when c is encrypting 0^n, since otherwise we could have used this test to break semantic security.

Quasi-arguments from ZTHE. Putting the pieces together, we describe the publicly verifiable quasi-argument of PR based on a 3-key homomorphic encryption scheme with a weak zero test. We start with a oversimplified version. Given a 3CNF formula φ over 2^m variables, we index its variables by m-bit strings. For locality parameter K, the CRS contains K variables $z_1, \ldots, z_K \in \{0,1\}^m$, each encrypted under the ZTHE using a different key. Given an input x and an assignment $w : \{0,1\}^m \to \{0,1\}$ satisfying φ_x, the prover homomorphically evaluates the value $b_i = w(z_i)$ assigned to each z_i, under the encryption, and sends the encrypted bits to the verifier. For every three variables z_i, z_j, z_k, the verifier homomorphically evaluates the consistency test $V_x(z_i, z_j, z_k, b_i, b_j, b_k)$ that outputs zero if and only if assigning the variables z_i, z_j, z_k with the values b_i, b_j, b_k locally satisfies φ_x. The verifier obtains the evaluated ciphertext $c_{i,j,k}$ encrypting the output of V_x and zero-tests it. If the zero test passes for every triplet of CRS variables, then the verifier accepts.

The fact that every accepted proof encrypts a partial assignment that locally satisfies φ_x follows directly from the soundness of the zero-test. As explained above, this fact together with the encryption's semantic security implies a no-signaling extractor. However, there are still two issues with the proposed construction. First, is does not satisfy *completeness*. The issue is that, when interacting with the honest prover, the verifier obtains ciphertexts $c_{i,j,k}$ that decrypt to zero, but are not trivial zeros and therefore they may not pass the zero test. The second issue is the efficiency of the verifier. To check the proof, the verifier homomorphically evaluates the consistency test V_x. For a general formula φ this may require time poly($|\varphi|$). Next we explain how to overcome these two issues.

Completeness via sum-check. We first explain why the ciphertext $c_{i,j,k}$ is not a trivial zero. Recall that when the prover is honest, the ciphertext $c_{i,j,k}$ is obtained from the CRS ciphertexts encrypting z_i, z_j, z_k by first evaluating the assignment w on each encrypted variable, and then evaluating the consistency test V_x on the encrypted variables and their assigned values. Let A_0 be the circuit that takes as input three

variables $u_1, u_2, u_3 \in \{0,1\}^m$ and outputs the value $V_x(u_1, u_2, u_3, w(u_1), w(u_2), w(u_3))$. First, observe that if the assignment w indeed satisfies φ_x, then A_0, as a boolean circuit, indeed computes the all-zero function. However, as an arithmetic circuit over $\mathbb{Z}$, A_0 may not be identically zero and, therefore, the evaluated ciphertext $c_{i,j,k}$ may not pass the zero test.

To achieve completeness, the idea in PR is to help the verifier check that $c_{i,j,k}$ indeed decrypts to zero by providing it with a proof of that fact. Very roughly, this proof will alow the verifier to compute a sequence of ciphertexts and zero-tests them. If $c_{i,j,k}$ decrypts to zero and the proof is computed honestly, all the ciphertexts will be trivial zeros. However, if $c_{i,j,k}$ does not decrypt to zero, then the ciphertexts computed from the proof will not all decrypt to zero. PR demonstrate such a proof based on the sum-check component of the classic PCP of Babai et al. [BFLS91]. We describe the details next.

Let $\ell = 3m$, let $z = (u_1, u_2, u_3) \in \{0,1\}^\ell$ be a triplet of variables encrypted in the CRS, and let c_0 be the ciphertext obtained from homomorphically evaluating A_0 on the encryption of z. Our goal is to convince the verifier that c_0 indeed decrypts to zero. To prove that, we rely on the fact that $A_0(y) = 0$ for every $y \in \{0,1\}^\ell$ (however, A_0 may not be all-zero over $\mathbb{Z}$) and on the fact that the polynomial computed by A_0 (viewed as an arithmetic circuit) is of low individual degree δ (the length of the proof will grow with δ). Later in this section we show that in fact $\delta = 2$.

The proof that c_0 decrypts to zero is computed as follows. For every $i \in [\ell]$ let $A_i : \mathbb{Z}^\ell \to \mathbb{Z}$ be the polynomial:

$$A_i(y_1, \ldots, y_\ell) \equiv \sum_{v \in \{0,1\}} \mathsf{ID}(y_i, v) \cdot A_{i-1}(y_1, \ldots, y_{i-1}, v, y_{i+1}, \ldots, y_\ell) \ , \tag{1}$$

where ID is the multilinear bivariate polynomial that computes the equality predicate on $\{0,1\}^2$. It is easy to verify that $A_i(y) = 0$ for every $y \in \{0,1\}^\ell$. Since the polynomial A_ℓ is multilinear and $A_\ell(y) = 0$ for every $y \in \{0,1\}^\ell$, A_ℓ must be the all-zero polynomial (over $\mathbb{Z}$).

The proof contains the ciphertext c_i encrypting $A_i(z)$ for every $i \in [\ell]$. Now the verifier can zero-test c_ℓ and, since $A_\ell \equiv 0$, the test passes. Therefore, to convince the verifier that c_0 decrypts to zero, it is sufficient to convince it that for every $i < \ell$, c_i and c_{i+1} decrypt to the same value.The high-level idea is to use the fact that (1) holds over $\mathbb{Z}$. To implement this idea we add more ciphertexts to the proof. In more detail, for every $i \in [\ell]$ let $A^z_{i-1} : \mathbb{Z} \to \mathbb{Z}$ be the univariate polynomial:

$$A^z_{i-1}(v) \equiv A_{i-1}(z_1, \ldots, z_{i-1}, v, z_{i+1}, \ldots, z_\ell) \ . \tag{2}$$

Since A_0 is of individual degree δ, the degree of A^z_{i-1} is also δ. For every $i \in [\ell]$, the prover homomorphically evaluates the $\delta + 1$ coefficients of the polynomial A^z_{i-1} and sends them to the verifier (in addition to the ciphertext $c_0, \ldots, c_\ell$). Given these encrypted polynomials, the verifier homomorphically evaluates the value $A^z_{i-1}(z_i)$. The verifier then checks that this is indeed the value encrypted in c_{i-1} by subtracting the two ciphertexts and zero-testing the result. If c_{i-1} and A^z_{i-1} are computed honestly then, by (2), the zero-test is guaranteed to pass. Next the verifier homomorphically evaluates the value $\sum_{v \in \{0,1\}} \mathsf{ID}(z_i, v) \cdot A^z_{i-1}(v)$. The verifier then checks that this is indeed the value encrypted in c_i by subtracting the two ciphertexts and zero-testing the result. If c_i and A^z_{i-1} are computed honestly then, by (1) and (2), the zero-test is again guaranteed to pass.

For soundness, consider an adversarially chosen ciphertext $\tilde{c}_0$ encrypting a value $\tilde{\alpha}_0$, and a proof that contains the ciphertexts $\tilde{c}_1, \ldots, \tilde{c}_\ell$ encrypting values $\tilde{\alpha}_1, \ldots, \tilde{\alpha}_\ell$, as well as encrypted polynomials $\tilde{A}^z_0, \ldots, \tilde{A}^z_{\ell-1}$. We argue that if all of the verifier's zero-tests pass then $\tilde{\alpha}_0 = 0$. First, if $\tilde{c}_\ell$ passes then $\tilde{\alpha}_\ell = 0$. Next, for every $i \in [\ell]$, if the verifier's zero-tests pass then:

$$\tilde{\alpha}_{i-1} = \tilde{A}^z_{i-1}(z_i) \quad , \quad \tilde{\alpha}_i = \sum_{v \in \{0,1\}} \mathsf{ID}(z_i, v) \cdot \tilde{A}^z_{i-1}(v) \ .$$

Since $z_i \in \{0,1\}$ it follows that $\tilde{\alpha}_{i-1} = \tilde{\alpha}_i$. Overall we have that $\tilde{\alpha}_0 = \cdots = \tilde{\alpha}_\ell = 0$.

The consistency test V_x. We complete the description of the PR quasi-argument by explaining how the verifier can evaluate the consistency test V_x efficiently. Recall that the test V_x is given the names of three variable $u_1, u_2, u_3 \in \{0,1\}^m$ of φ and three bits $b_1, b_2, b_3 \in \{0,1\}$ and it outputs zero if and only if the partial assignment mapping u_i to b_i locally satisfies φ_x.

Previous works [KRR13, KRR14, PR17], gave quasi-arguments only for log-space uniform formulas φ. For such formulas the test V_x can be implemented by a small circuit of size $|x| \cdot \mathrm{poly}(m)$ and, therefore, the verifier can evaluate V_x on its own. In contrast, in this work we construct quasi-arguments for arbitrary formulas where V_x may be as large as the formula φ itself.

Our main idea is to add to the CRS values that help the verifier evaluate V_x on all the required inputs. Since the input x in not fixed at the time the CRS is generated, the CRS should help the verifier evaluate V_x for every possible input x. In more detail, we first show how to write the function V_x as a sum $V_x = V + V_{1,x_1} + \cdots + V_{n,x_n}$ where, roughly speaking, the function V checks consistency with the formula φ[6] (but not with x), and $V_{\ell,b}$ checks that the assignments to the ℓ-th input bit is b. We then modify the CRS as follows. For every three variables z_i, z_j, z_k encrypted in the CRS, and for every three bits $b_1, b_2, b_3 \in \{0,1\}$, we homomorphically evaluate $V(z_i, z_j, z_k, b_1, b_2, b_3)$ and add the resulting ciphertext to the CRS. The circuits $\{V_{\ell,b}\}$ are simple and the verifier can evaluate them on its own.

On the degree of A_0. Next we argue that the circuit A_0 is of individual degree $\delta = 2$. Recall that the quasi-argument proof length grows with δ. Looking ahead, the individual degree of A_0 also plays a role in basing the security of our quasi-argument on bilinear maps satisfying Assumption 1.1. Recall that A_0 takes as input three variables $u_1, u_2, u_3 \in \{0,1\}^m$ and outputs the value $V_x(u_1, u_2, u_3, w(u_1), w(u_2), w(u_3))$ where w is the provers assignment and $V_x = V + V_{1,x_1} + \cdots + V_{n,x_n}$ is the consistency test. We observe that the functions V and $\{V_{\ell,b}\}$ can be computed by a multilinear arithmetic circuit. The size of the multilinear circuit for V may be exponential in the input size, however, since the input size is $O(m)$, the evaluation time is only $\mathrm{poly}(|\varphi|)$. Since V is not evaluated by the verifier (but only during the CRS generation) this does not compromise efficiency. Similarly, the assignment $w : \{0,1\}^m \to \{0,1\}$ evaluated by the prover can be computed by a multilinear arithmetic circuit. Therefore, A_0 is of individual degree 2.

The PR encryption scheme. As described above, PR base their quasi-argument on a 3-key homomorphic encryption with a weak zero-test. Next we describe their ZTHE construction from multilinear maps. Very roughly, we think of degree-δ multilinear maps as letting us encode elements from a large field $\mathbb{F}$, and allowing us to homomorphically evaluate any polynomial of degree at most δ over these encodings, and test if the result encodes zero or not. However, we assume that evaluating computations of degree higher than δ is hard. We can think of bilinear maps as degree 2 multilinear maps.

The PR encryption scheme supports homomorphic computations of bounded degree δ. The secret key is a random field element $s \in \mathbb{F}$. To encrypt a message $\mu \in \{0,1\}$, sample a random degree-δ polynomial $C : \mathbb{F} \to \mathbb{F}$ such that $C(s) = \mu$. Then, encode each of the $\delta + 1$ coefficients of C with a degree-δ multilinear map. The ciphertext consists of this encoding of C that we denote by $[C]$. To decrypt, homomorphically evaluate $[C(s)]$ (this is a linear function since s is in the clear) and test if the result encodes zero or not.[7]

Given ciphertexts $[C_1], [C_2]$ encrypting messages μ_1, μ_2 respectively, an encryption of $\mu_1 \circ \mu_2$ (where $\circ$ is either addition or multiplication over $\mathbb{F}$) is given by the encoded polynomial $[C_1 \circ C_2]$. The encoded coefficients of $C_1 \circ C_2$ are computed homomorphically from the coefficients of C_1 and C_2. Therefore, the encryption supports homomorphic computations of degree at most δ. As for multi-key homomorphism, if $[C_1]$ encrypts μ_1 under s_1 and $[C_2]$ encrypts μ_2 under s_2, then an encryption of $\mu_1 \circ \mu_2$ under both s_1 and s_2 is given by the encoded bivariate polynomial $C(x, y) = C_1(x) \circ C_2(y)$. Note that the number of coefficients in the ciphertext grows exponentially with the number of secret keys. Since we only require 3-key homomorphism this is not an issue.

Given a ciphertext $[C]$, the weak zero-test simply tests that all the encoded coefficients of C are zero. For soundness, if $[C]$ passes the zero test then C must decrypt to zero. To get soundness for maliciously generated

[6] We need to make minor changes to the formula φ as discussed in Section 5.

[7] This scheme can encrypt messages in $\mathbb{F}$, however, we only have efficient decryption for messages in $\{0,1\}$.

ciphertexts PR rely on so-called "clean" multilinear maps. To argue completeness, consider an arithmetic circuit $A : \mathbb{Z}^\ell \to \mathbb{Z}$ computing the all-zero function over $\mathbb{Z}$. Evaluating A on ciphertexts $[C_1], \ldots, [C_\ell]$ results in the encoded polynomial $[A(C_1, \ldots, C_\ell) \equiv 0]$ that passes the zero test.

In PR the semantic security of this scheme is shown under an appropriate hardness assumption on the multilinear map. Note that for security it is crucial that the degree of the ciphertext polynomial C is at least the degree δ of the multilinear map. Otherwise, given δ encoded polynomials of degree, say, $\delta - 1$ encrypting either 0^δ or a random message, we can distinguish the two cases by homomorphically computing the determinant of the δ-by-δ coefficient matrix, and testing if it is zero.

2.3.3 Quasi-arguments from bilinear maps.

In this section we complete the description of our publicly verifiable succinct non-interactive quasi-argument for **NP** with adaptive soundness and a long CRS. As explained in Section 2.3.1, starting from the PR quasi-argument, our high-level idea is to replace the multilinear encoding of the elements in the CRS and proof with bilinear encodings of monomials in these elements. In this section, we explain in more detail how we change the PR protocol. In particular, we analyze the size of our CRS and proof and explain how to base security on Assumption 1.1.

Since in the PR construction multilinear maps are only used to instantiate the ZTHE, it is natural to try and construct ZTHE directly from bilinear maps. This would allow us to abstract the use of bilinear maps and present our protocol in a way that closely follows the PR blueprint. Unfortunately, we do not know how to use the idea of encoding monomials to construct ZTHE from bilinear maps. The issue is that to compute homomorphically on multiple ZTHE ciphertexts, we need encoding of monomials that depend on all the ciphertexts together. However, since each ciphertext is encrypted independently, we cannot provide such encoded monomials as part of the ciphertext. Nonetheless, we define a limited version of ZTHE that is sufficient for quasi-argument, and instantiated from bilinear maps.

The main limitation of our encryption scheme is that it only supports arity-one homomorphic operations. That is, given a ciphertext encrypting a long message, it is possible to compute homomorphically on the bits of the message, but it is not possible to compute over multiple messages encrypted independently. Very roughly, the construction is as follows. Recall that in the ZTHE of PR we encrypt a message $\mu \in \{0,1\}^\ell$ under secret key s by sampling polynomials $C_1, \ldots, C_\ell$ such that $C_i(s) = \mu_i$ and encoding their coefficients with a multilinear map. In our scheme, we compute products of the polynomials $C_1, \ldots, C_m$ and encode the coefficients of the resulting polynomials with a bilinear map. To prevent the ciphertext from growing too much, we only compute the products that are used by the quasi-argument. We note that our arity-one encryption must support multi-key homomorphic operations. To this end, we define multi-key ciphertexts: a single multi-key ciphertext encrypts multiple messages under multiple secret keys. In the rest of this overview we assume, for simplicity, that all ciphertext are under a single key.

Next we explain how to use such arity-one homomorphic encryption to get a quasi-argument with the required efficiency: for security parameter κ, formula φ of size T, and locality parameter K, the CRS should be of length $\mathrm{poly}(\kappa, K, T)$ and the proof should be of length $\mathrm{poly}(\kappa, K)$. Recall that in the PR quasi-argument the CRS contains encryptions of K of φ's variables. Each variable is represented by $m = O(\log T)$ bits and the CRS contains ZTHE encryption of each bit. For every three of the K variables in the CRS, the prover homomorphically evaluates polynomials over the $3m$ encrypted bits describing these variables. The proof contains, for every three variables, $O(m)$ ciphertexts, each encrypting $O(1)$ elements. Therefore, in our protocol, the CRS includes, for every three variables, one ciphertext encrypting $3m$ bits and the prover homomorphically computes over one CRS ciphertext at a time. Therefore, it remains to show that each ciphertext in the CRS is of length $\mathrm{poly}(\kappa, T)$ and each ciphertext in the proof is of length $\mathrm{poly}(\kappa)$.

Recall that in the PR quasi-argument, for every three variables the CRS contains $3m$ encrypted bits and the prover and verifier homomorphically evaluate polynomials of constant individual degree δ over these bits. Therefore, in our protocol, the CRS ciphertext encrypting these $3m$ bits consist of the encoded polynomials $C_1, \ldots, C_{3m}$ as well as encoding of every product $\prod C_i^{\delta_i}$ where $\delta_1, \ldots, \delta_{3m} \in [0, \delta]$. As in the PR encryption, to get semantic security, the polynomials $C_1, \ldots, C_{3m}$ must be of degree $\mathrm{poly}(m)$ (we elaborate on this next when discussing the security of our encryption). Overall each CRS ciphertext contains

$\delta^{3m} \cdot \text{poly}(m) = \text{poly}(T)$ bilinear encodings. Since each ciphertext in the proof encrypts $O(1)$ elements, a similar argument shows that each proof ciphertext contains $\text{poly}(m)$ bilinear encodings.

Semantic security. We prove the semantic security of our arity-one ZTHE under Assumption 1.1. Next, we provide some intuition for this reduction. Recall that to encrypt a message $\mu \in \{0,1\}^\ell$ under secret key s, we sample polynomials $C_1, \ldots, C_\ell$ such that $C_i(s) = \mu_i$, compute some products of these polynomials, and encode the coefficients of these products with a bilinear map. Similarly to the scheme of PR, to get semantic security and support homomorphic evaluation of total degree δ, each polynomial C_i must be of degree at least δ.

For simplicity, we start by arguing that the $\delta+1$ encoded coefficient of the polynomial C_i alone (without the rest of the ciphertext) hides the bit μ_i. We argue this based on the following weak version of Assumption 1.1: given bilinear encodings $[s]$ and $[t]$ it is hard to distinguish between the case where $t = s^\delta$ and the case where $t \in \mathbb{F}$ is a random and independent. First observe that given $[s]$ we can sample an encoding of a random degree-δ polynomial C such that $C(s) = 0$ as follows: sample a random polynomial $R \in \mathbb{F}[X]$ of degree $\delta - 1$ and homomorphically compute an encoding of the polynomial $[C = R \cdot X - s \cdot R]$. Now, by subtracting 1 from the coefficient of X^δ and adding $t + \mu_i$ to the free coefficient we get either an encoding of a polynomial distributed like C_i if $t = s^\delta$, or an encoding of a completely random polynomial if t is random. To prove semantic security we must be able to sample the entire ciphertext and not just $[C_i]$. We do so in a similar manner using the additional encodings given in Assumption 1.1.

2.4 Organization.

The rest of the paper is organized as follows. In Section 3 we define delegation schemes for Turing machines and RAM machines. In Section 4 we formalize our limited notion of ZTHE and construct it from bilinear maps. In Section 5 we construct a quasi-argument from our ZTHE. Our bootstrapping theorem going from quasi-arguments to delegation is described in Section 6.

3 Delegation

In this section we define the notion of a publicly verifiable non-interactive delegation scheme for deterministic Turing machines and RAM machines. We show that delegation for RAM machines implies delegation for Turing machines (see Theorem 3.7). In the subsequent sections we state our results only for the notion of RAM delegation.

3.1 Turing Machine Delegation

We define delegation for a Turing machine $\mathcal{M}$. For example, $\mathcal{M}$ can be the universal Turing machine. A publicly verifiable non-interactive delegation scheme for $\mathcal{M}$ consists of algorithms $(\mathsf{Del.S}, \mathsf{Del.P}, \mathsf{Del.V})$ with the following syntax:

Setup: The randomized setup algorithm $\mathsf{Del.S}$ takes as input a security parameter $\kappa \in \mathbb{N}$, a time bound T and an input length n, and outputs a pair of public keys: a prover key pk and a verifier key vk.

Prover: The deterministic prover algorithm $\mathsf{Del.P}$ takes as input a prover key pk and an input $x \in \{0,1\}^n$. It outputs a proof Π.

Verifier: The deterministic verifier algorithm $\mathsf{Del.V}$ takes as input a verifier key vk, an input $x \in \{0,1\}^n$ and a proof Π. It outputs a bit indicating if it accepts or rejects.

In the following definition, $\mathcal{U}_\mathcal{M}$ denotes the language such that $(x,T) \in \mathcal{U}_\mathcal{M}$ if and only if $\mathcal{M}$ accepts x within at most T steps.

Definition 3.1. *A publicly verifiable non-interactive delegation scheme* $(\mathsf{Del.S}, \mathsf{Del.P}, \mathsf{Del.V})$ *for* $\mathcal{M}$ *with setup time* $T_\mathsf{S} = T_\mathsf{S}(\kappa, T)$ *and proof length* $L_\Pi = L_\Pi(\kappa, T)$ *satisfies the following requirements.*

Completeness. *For every $\kappa, T, n \in \mathbb{N}$ such that $n \leq T \leq 2^\kappa$, and $x \in \{0,1\}^n$ such that $(x,T) \in \mathcal{U}_\mathcal{M}$:*

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

Efficiency. *In the completeness experiment above:*

- *The setup algorithm runs in time T_S.*
- *The prover runs in time $\mathrm{poly}(\kappa, T)$, and outputs a proof of length L_Π.*
- *The verifier runs in time $O(L_\Pi) + n \cdot \mathrm{poly}(\kappa)$.*

Soundness. *For every PPT adversary Adv and pair of polynomials $T = T(\kappa)$ and $n = n(\kappa)$ there exists a negligible function μ such that for every $\kappa \in \mathbb{N}$:*

$$\Pr\left[\begin{array}{l}\mathsf{Del.V}(\mathsf{vk}, x, \Pi) = 1 \\ (x,T) \notin \mathcal{U}_\mathcal{M}\end{array} \;\middle|\; \begin{array}{l}(\mathsf{pk},\mathsf{vk}) \leftarrow \mathsf{Del.S}(\kappa, T, n) \\ (x, \Pi) \leftarrow \mathsf{Adv}(\mathsf{pk}, \mathsf{vk})\end{array}\right] \leq \mu(\kappa) \ .$$

Remark 3.2 (Prover specific running time). In the above definition the prover's running time grows with T even if the running time of $\mathcal{M}$ on x is much lower. We can avoid this overhead by generating several key pairs for running time 2^i for every $i \leq \log T$ and choose which pair to use depending on the running time of the specific computation.

Remark 3.3 (Linear verification time). The above definition requires that the verifier's running time is linear in the input length n and proof length L_Π. While our constructions do achieve such optimal verification time, we note that any verification time that is less than the running time of $\mathcal{M}$ is non-trivial.

3.2 RAM Delegation

A RAM machine $\mathcal{R}$ with a word size of ℓ is modeled as a deterministic machine with random access to memory of size 2^ℓ bits and a local state of size $O(\ell)$. At every step, the machine reads or writes a single memory bit and updates its state. We refer to the machine's memory and state at a given timestep as its configuration cf. For simplicity, we think of the machine as having no input or output other than its memory and state. Also, we always use the security parameter κ as the word size.

A publicly verifiable non-interactive delegation scheme for $\mathcal{R}$ consists of algorithms $(\mathsf{RDel.S}, \mathsf{RDel.D}, \mathsf{RDel.P}, \mathsf{RDel.V})$ with the following syntax:

Setup: The randomized setup algorithm $\mathsf{RDel.S}$ takes as input a security parameter $\kappa \in \mathbb{N}$, a time bound T and outputs a triplet of public keys: a prover key pk, a verifier key vk, and a digest key dk.

Digest: The deterministic digest algorithm $\mathsf{RDel.D}$ takes as input the digest key dk and a configuration $\mathsf{cf} \in \{0,1\}^*$ and outputs a digest h.

Prover: The deterministic prover algorithm $\mathsf{RDel.P}$ takes as input a prover key pk, and a pair of source and destination configurations $\mathsf{cf}, \mathsf{cf}'$. It outputs a proof Π.

Verifier: The deterministic verifier algorithm $\mathsf{RDel.V}$ takes as input a verifier key vk, a pair of digests h, h' and a proof Π. It outputs a bit indicating if it accepts or rejects.

In the following definition, $\mathcal{U}_\mathcal{R}$ denotes the language such that $(\ell, \mathsf{cf}, \mathsf{cf}', T) \in \mathcal{U}_\mathcal{R}$ if and only if the machine $\mathcal{R}$ with word size ℓ starting from configuration cf transitions to configuration cf' in T steps.

Definition 3.4. *A publicly verifiable non-interactive delegation scheme $(\mathsf{RDel.S}, \mathsf{RDel.D}, \mathsf{RDel.P}, \mathsf{RDel.V})$ for $\mathcal{R}$ with setup time $T_\mathsf{S} = T_\mathsf{S}(\kappa, T)$ and proof length $L_\Pi = L_\Pi(\kappa, T)$ satisfies the following requirements.*

Completeness. *For every $\kappa, T \in \mathbb{N}$ such that $T \leq 2^\kappa$, and $\mathsf{cf}, \mathsf{cf}' \in \{0,1\}^*$ such that $(\kappa, \mathsf{cf}, \mathsf{cf}', T) \in \mathcal{U}_\mathcal{R}$:*

$$\Pr\left[\mathsf{RDel.V}(\mathsf{vk}, \mathsf{h}, \mathsf{h}', \Pi) = 1 \;\middle|\; \begin{array}{l} (\mathsf{pk}, \mathsf{vk}, \mathsf{dk}) \leftarrow \mathsf{RDel.S}(\kappa, T) \\ \mathsf{h} \leftarrow \mathsf{RDel.D}(\mathsf{dk}, \mathsf{cf}) \\ \mathsf{h}' \leftarrow \mathsf{RDel.D}(\mathsf{dk}, \mathsf{cf}') \\ \Pi \leftarrow \mathsf{RDel.P}(\mathsf{pk}, \mathsf{cf}, \mathsf{cf}') \end{array}\right] = 1 \ .$$

Efficiency. *In the completeness experiment above:*

- *The setup algorithm runs in time T_S.*
- *The digest algorithm on cf runs in time $|\mathsf{cf}| \cdot \mathrm{poly}(\kappa)$ and outputs a digest of length κ.*
- *The prover runs in time $\mathrm{poly}(\kappa, T, |\mathsf{cf}|)$ and it outputs a proof of length L_Π.*
- *The verifier runs in time $O(L_\Pi) + \mathrm{poly}(\kappa)$.*

Collision resistance. *For every PPT adversary Adv and polynomial $T = T(\kappa)$ there exists a negligible function μ such that for every $\kappa \in \mathbb{N}$:*

$$\Pr\left[\begin{array}{l} \mathsf{cf} \neq \mathsf{cf}' \\ \mathsf{RDel.D}(\mathsf{dk}, \mathsf{cf}) = \mathsf{RDel.D}(\mathsf{dk}, \mathsf{cf}') \end{array} \;\middle|\; \begin{array}{l} (\mathsf{pk}, \mathsf{vk}, \mathsf{dk}) \leftarrow \mathsf{RDel.S}(\kappa, T) \\ (\mathsf{cf}, \mathsf{cf}') \leftarrow \mathsf{Adv}(\mathsf{pk}, \mathsf{vk}, \mathsf{dk}) \end{array}\right] \leq \mu(\kappa) \ .$$

Soundness. *For every PPT adversary Adv and polynomial $T = T(\kappa)$ there exists a negligible function μ such that for every $\kappa \in \mathbb{N}$:*

$$\Pr\left[\begin{array}{l} \mathsf{RDel.V}(\mathsf{vk}, \mathsf{h}, \mathsf{h}', \Pi) = 1 \\ (\kappa, \mathsf{cf}, \mathsf{cf}', T) \in \mathcal{U}_\mathcal{R} \\ \mathsf{h} = \mathsf{RDel.D}(\mathsf{dk}, \mathsf{cf}) \\ \mathsf{h}' \neq \mathsf{RDel.D}(\mathsf{dk}, \mathsf{cf}') \end{array} \;\middle|\; \begin{array}{l} (\mathsf{pk}, \mathsf{vk}, \mathsf{dk}) \leftarrow \mathsf{RDel.S}(\kappa, T) \\ (\mathsf{cf}, \mathsf{cf}', \mathsf{h}, \mathsf{h}', \Pi) \leftarrow \mathsf{Adv}(\mathsf{pk}, \mathsf{vk}, \mathsf{dk}) \end{array}\right] \leq \mu(\kappa) \ .$$

Remark 3.5 (Collision resistance). We note that the collision resistance of the digest does not follow from our soundness requirement and, therefore, we explicitly require collision resistance. For example, by using a constant function as the digest we can trivially satisfy soundness.

Remark 3.6 (RAM delegation in previous work). Our focus in this work is on constructing delegation for Turing machines. However, as explained in the introduction, we rely on the stronger notion of RAM delegation to prove our bootstrapping theorem. As a result, our definition of RAM delegation differs from that in previous work [KP16, BHK17] regarding both prover's efficiency and soundness.

- Previous work explicitly requires that the running time of the prover (modeled as a RAM machine) is polynomial T (the running time of $\mathcal{R}$) and does not grow with the configuration size. However, for simplicity, our definition allows the prover's running time to grow with the configuration. We note that our construction satisfies the stronger notion of prover efficiency considered in previous work.
- The soundness definition in previous work only requires that the adversary produces accepting proofs for two different statements $(\mathsf{h}, \mathsf{h}')$ and $(\mathsf{h}, \mathsf{h}'')$ with the same initial digest. In our definition, however, the adversary must explicitly output the full configurations $\mathsf{cf}, \mathsf{cf}'$. We choose this weaker definition since it allows us to achieve adaptive soundness which is required for our bootstrapping theorem. In contrast, to achieve the prior definition, previous work restricted the adversary to choose h independently of pk. We emphasize that our soundness definition suffices for delegating Turing machines.

Next we argue that RAM delegation implies delegation for Turing machines (Definition 3.1).

Theorem 3.7. *Suppose that for any RAM machine there exists a publicly verifiable non-interactive delegation scheme with setup time T'_S and proof length L'_Π. Then for any Turing machine there exists a publicly verifiable non-interactive delegation scheme with setup time T_S and proof length L_Π where $T_\mathsf{S}(\kappa, T) = T'_\mathsf{S}(\kappa, T')$, $L_\Pi(\kappa, T) = L'_\Pi(\kappa, T')$ for $T' = O(T)$.*

Proof. Fix any deterministic Turing machine $\mathcal{M}$. Consider the RAM machine $\mathcal{R}$, that given initial memory that includes the input x, emulates the Turing machine $\mathcal{M}$ on input x. If $\mathcal{M}$ reaches an accepting state, $\mathcal{R}$ moves to a special accepting configuration cf^* and remains in it. Therefore, for every $T \in \mathbb{N}$ there exists $T' = O(T)$ such that $(x, T) \in \mathcal{U}_{\mathcal{M}}$ if and only if $(\kappa, \mathsf{cf}_x, \mathsf{cf}^*, T') \in \mathcal{U}_{\mathcal{R}}$ where cf_x is the initial configuration of $\mathcal{R}$ with memory x.

Let $(\mathsf{RDel.S}, \mathsf{RDel.D}, \mathsf{RDel.P}, \mathsf{RDel.V})$ be a publicly verifiable non-interactive delegation scheme for $\mathcal{R}$ with setup time T_{S} and proof length L_{Π}. We construct a publicly verifiable non-interactive delegation scheme $(\mathsf{Del.S}, \mathsf{Del.P}, \mathsf{Del.V})$ for $\mathcal{M}$ as follows.

- The setup algorithm $\mathsf{Del.S}$ on input (κ, T, n) emulates $\mathsf{RDel.S}(\kappa, T')$ and obtains the keys $(\mathsf{pk}, \mathsf{vk}, \mathsf{dk})$. It outputs $\mathsf{pk}' = \mathsf{pk}$ and $\mathsf{vk}' = (\mathsf{vk}, \mathsf{dk})$.
- The prover algorithm $\mathsf{Del.P}$ on input (pk, x) emulates $\mathsf{RDel.P}(\mathsf{pk}, \mathsf{cf}_x, \mathsf{cf}^*)$, obtains the proof Π and outputs it.
- The verification algorithm $\mathsf{Del.V}$ on input $(\mathsf{vk}' = (\mathsf{vk}, \mathsf{dk}), x, \Pi)$ computes the digests $\mathsf{h}_x = \mathsf{RDel.D}(\mathsf{dk}, \mathsf{cf}_x)$ and $\mathsf{h}^* = \mathsf{RDel.D}(\mathsf{dk}, \mathsf{cf}^*)$, emulates $\mathsf{RDel.V}(\mathsf{vk}, \mathsf{h}_x, \mathsf{h}^*, \Pi)$ and accepts if and only if $\mathsf{RDel.D}$ accepts.

The completeness and efficiency guarantees follow directly from those of the RAM delegation scheme. For soundness, assume there exists a PPT adversary Adv breaking the soundness of the scheme. We construct a PPT adversary Adv' breaking the soundness of the RAM delegation scheme. Given keys $(\mathsf{pk}, \mathsf{vk}, \mathsf{dk})$, Adv' emulates $\mathsf{Adv}(\mathsf{pk}, (\mathsf{vk}, \mathsf{dk}))$ and obtains an input x and a proof Π. Adv' computes in time $O(T)$ the unique configuration cf'_x such that $(\kappa, \mathsf{cf}_x, \mathsf{cf}'_x, T') \in \mathcal{U}_{\mathcal{R}}$. It also computes the digests $\mathsf{h}_x, \mathsf{h}'_x, \mathsf{h}^*$ of the configurations $\mathsf{cf}_x, \mathsf{cf}'_x, \mathsf{cf}^*$ respectively. If $\mathsf{h}'_x = \mathsf{h}^*$, Adv' output $(\mathsf{cf}'_x, \mathsf{cf}^*)$. Otherwise, it output $(\mathsf{cf}_x, \mathsf{cf}'_x, \mathsf{h}_x, \mathsf{h}^*, \Pi)$.

If $(x, T) \notin \mathcal{U}_{\mathcal{M}}$ then $(\kappa, \mathsf{cf}_x, \mathsf{cf}^*, T') \notin \mathcal{U}_{\mathcal{R}}$ and, in particular, $\mathsf{cf}^* \neq \mathsf{cf}'_x$. If $\mathsf{Del.V}((\mathsf{vk}, \mathsf{dk}), x, \Pi)$ accepts, then $\mathsf{RDel.V}(\mathsf{vk}, \mathsf{h}_x, \mathsf{h}^*, \Pi)$ accepts. Therefore, Adv' breaks either the soundness or the collision resistance of the RAM delegation scheme. □

4 Zero-Testable Homomorphic Encryption

Our quasi-argument is based on a limited version of the zero-testable homomorphic encryption defined in the work of [PR17]. We start by defining our notion of encryption in Section 4.1. In Section 4.2 we construct such an encryption based on bilinear maps. The analysis is given in Section 4.3. See Section 2.3.3 for an overview, and Section 2.3.2 for a discussion on the original notion from [PR17].

4.1 Definition

We introduce our limited zero-testable homomorphic encryption in three steps. In Section 4.1.1 we define a simple somewhat homomorphic encryption. In Section 4.1.2 we introduce the zero-test procedure. In Section 4.1.3 we add support for multi-key homomorphic operations.

4.1.1 Simple homomorphic encryption.

In this section we define a simple somewhat homomorphic encryption scheme. We first describe the syntax and then highlight some important differences between this notion and standard formulations in the literature. Our homomorphic encryption scheme is parameterized by a bound $\bar{\delta} \in \mathbb{N}$ on the individual degree of homomorphic computations. The scheme is given by the algorithms $(\mathsf{ParamGen}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{Dec}, \mathsf{Val})$ with the following syntax:

Parameter generation: the probabilistic parameter generation algorithm $\mathsf{ParamGen}$ takes as input the security parameter $\kappa \in \mathbb{N}$ and a bound $\bar{\ell} \in \mathbb{N}$ on the length of messages. It outputs public parameters pp. The public parameters include a description of the field $\mathbb{F}$ that defines the plaintext space. The running time of $\mathsf{ParamGen}$ is $\mathrm{poly}(\kappa, \log \bar{\ell})$.

Key generation: the PPT key generation algorithm KeyGen takes as input the public parameters pp and outputs a secret key sk.

Encryption: the probabilistic encryption algorithm Enc takes an input a secret key sk, a message $m \in \{0,1\}^{\leq \bar{\ell}}$ and randomness $r \in \{0,1\}^{\kappa}$ (in what follows, it will be convenient to refer to the encryption randomness as an explicit input). It outputs a ciphertext c. The running time of Enc is $\mathrm{poly}(\kappa, \bar{\delta}^{|m|})$.

Homomorphic evaluation: the deterministic polynomial-time homomorphic evaluation algorithm Eval takes as input the public parameters pp, a ciphertext c (encrypting a message in $\mathbb{F}^{\ell}$) and a polynomial $P : \mathbb{Z}^{\ell} \to \mathbb{Z}$ of individual degree $\delta \leq \bar{\delta}$ represented by a list of (possibly non-distinct) monomials with coefficients in $\{1, -1\}$. It outputs an evaluated ciphertext e. The length of e is $\mathrm{poly}(\kappa) \cdot \ell \cdot \delta$.

Decryption: the deterministic polynomial-time decryption algorithm Dec takes as input a secret key sk and an evaluated ciphertext e. It either outputs a bit or a special symbol $\perp$ if the encrypted value is not in $\{0,1\}$.

Inefficient value recovery. the deterministic value recovery algorithm Val takes as input a secret key sk and an evaluated ciphertext e. It either outputs an element in $\mathbb{F}$ or a special symbol $\perp$ if the ciphertext is malformed. The value recovery algorithm is computationally unbounded.

We note some important differences between our notion and the standard formulations in the literature:

- We only support "arity-one" homomorphic evaluation. That is, the homomorphic evaluation algorithm can only operate on a single ciphertext at a time.
- The ciphertext size can grow exponentially with the message length.
- Evaluation is "one-hop". That is, we cannot continue to compute homomorphically over evaluated ciphertexts.
- While fresh ciphertexts can encrypt long messages, an evaluated ciphertext only encrypts a single element.
- The homomorphically evaluated polynomial P is represented by a list of monomials with coefficients in $\{1, -1\}$. This representation is convenient since it is well defined over any field. Note that we can also represent P, for example, as an arithmetic circuit that might be exponentially shorter than the list representation. However, the running time of the homomorphic evaluation algorithm we construct will be polynomial in the list representation size.
- While the homomorphic computations are evaluated over a field $\mathbb{F}$, encryption and decryption only support binary messages. Ciphertexts encrypting arbitrary values in $\mathbb{F}$ can be decrypted inefficiently via the value recovery algorithm. This algorithm is only used to define the encryption's correctness and is not used in constructions or in security reductions.

We proceed to define the correctness and security of the encryption scheme. We separate the correctness requirement into two properties: one for homomorphic evaluation and one for decryption of evaluated ciphertexts. In both definitions, we use the inefficient value recovery algorithm Val to recover the value encrypted by a ciphertext.

Definition 4.1 (Correctness of Evaluation)**.** *For every $\kappa, \bar{\ell} \in \mathbb{N}$, message $m \in \{0,1\}^{\ell}$ such that $\ell \leq \bar{\ell}$ and polynomial $P : \mathbb{Z}^{\ell} \to \mathbb{Z}$ of individual degree at most $\bar{\delta}$:*

$$\Pr\left[v = P(m) \;\middle|\; \begin{array}{l} \mathsf{pp} \leftarrow \mathsf{ParamGen}(\kappa, \bar{\ell}) \\ \mathsf{sk} \leftarrow \mathsf{KeyGen}(\mathsf{pp}) \\ r \leftarrow \{0,1\}^{\kappa} \\ c \leftarrow \mathsf{Enc}(\mathsf{sk}, m, r) \\ e \leftarrow \mathsf{Eval}(\mathsf{pp}, c, P) \\ v \leftarrow \mathsf{Val}(\mathsf{sk}, e) \end{array} \right] = 1 \ .$$

In the experiment above, the polynomial P is evaluated over $\mathbb{F}$ defined by pp.

The correctness of the decrypted value holds for arbitrary evaluated ciphertexts.

Definition 4.2 (Correctness of Decryption)**.** *For every $\kappa, \bar{\ell} \in \mathbb{N}$ and evaluated ciphertext e:*

$$\Pr\left[\begin{array}{l|l} v' \in \{0,1\} \;\Rightarrow\; v = v' & \mathsf{pp} \leftarrow \mathsf{ParamGen}(\kappa, \bar{\ell}) \\ v' \notin \{0,1\} \;\Rightarrow\; v = \bot & \mathsf{sk} \leftarrow \mathsf{KeyGen}(\mathsf{pp}) \\ & v \leftarrow \mathsf{Dec}(\mathsf{sk}, e) \\ & v' \leftarrow \mathsf{Val}(\mathsf{sk}, e) \end{array}\right] = 1 \ .$$

We also require one-time semantic security. The definition is restricted to the range of parameters supported by our construction.

Definition 4.3 (Semantic Security)**.** *For every function $\bar{\ell}(\kappa) = O(\log \kappa)$ and PPT adversary Adv, there exists a negligible function μ such that for every $\kappa \in \mathbb{N}$ and messages $m_0, m_1 \in \{0,1\}^{\ell}$ such that $\ell \leq \bar{\ell}(\kappa)$:*

$$\Pr\left[b' = b \;\middle|\; \begin{array}{l} b \leftarrow \{0,1\} \\ \mathsf{pp} \leftarrow \mathsf{ParamGen}(\kappa, \bar{\ell}(\kappa)) \\ \mathsf{sk} \leftarrow \mathsf{KeyGen}(\mathsf{pp}) \\ r \leftarrow \{0,1\}^{\kappa} \\ c_0 \leftarrow \mathsf{Enc}(\mathsf{sk}, m_0, r) \\ c_1 \leftarrow \mathsf{Enc}(\mathsf{sk}, m_1, r) \\ b' \leftarrow \mathsf{Adv}(\mathsf{pp}, c_b) \end{array}\right] \leq \frac{1}{2} + \mu(\kappa) \ .$$

4.1.2 Quadratic zero-test.

In addition to the requirements above, our homomorphic encryption must support a zero-test operation. We start by discussing this notion.

The zero-test takes as input an evaluated ciphertext e and indicates whether or not it decrypts to zero. We carefully define the zero-test such that it does not contradict semantic security: the test should never pass if e does not decrypt to zero. However, even if e does decrypt to zero, the test may still fail. The zero test is only required to pass if there exists an honestly generated ciphertext c encrypting a message in $\mathbb{F}^{\ell}$ and a polynomial $P : \mathbb{Z}^{\ell} \to \mathbb{Z}$ such that $P \equiv 0$ over $\mathbb{Z}$ and e is obtained by homomorphically evaluating P on c.

Two-hop homomorphism. So far, our notion of zero-testable homomorphic encryption only supports one-hop evaluation. However, to construct our quasi-argument, we require at least two hops of homomorphic evaluation before using the zero-test (one prover evaluation followed by one verifier evaluation). Using bilinear maps, we are able to construct a scheme with a limited version of two-hop homomorphism: given evaluated ciphertexts $e_1, \ldots, e_n$ (resulting from applying the homomorphic evaluation algorithm Eval on fresh ciphertexts) we can further evaluate a polynomial $Q : \mathbb{Z}^n \to \mathbb{Z}$ of total degree at most two over $e_1, \ldots, e_n$. The resulting ciphertext can be decrypted or zero-tested, but we can no longer compute on it homomorphically. We note that, unlike the first-hop homomorphic evaluation algorithm Eval that supports degree $\bar{\delta}$ evaluations but can only operate on a single ciphertext, the second-hop evaluation can operate on multiple ciphertexts but only supports degree two evaluations.

Lohighlthe differences here: we can perform ight oking ahead, in our quasi-argument the proof will consist of evaluated ciphertexts (resulting from Eval) and the verifier will perform a second-hop evaluation and zero-test the resulting ciphertext. For simplicity, since the resulting ciphertext is never decrypted, we combine the second-hop evaluation and the zero-test into a single algorithm. This *quadratic zero-test* takes as input a vector of evaluated ciphertexts $(e_1, \ldots, e_n)$ and a polynomial $Q : \mathbb{Z}^n \to \mathbb{Z}$ of total degree two. If the quadratic zero-test passes, then each ciphertext must decrypt to a message $m_i \in \mathbb{F}$ such that $Q(m_1, \ldots, m_n) = 0$. In the other direction, the quadratic zero-test is only guaranteed to pass if there exists an honestly generated ciphertext c encrypting a message in $\mathbb{F}^{\ell}$ and polynomials $P_1, \ldots, P_n : \mathbb{Z}^{\ell} \to \mathbb{Z}$ such that $Q(P_1, \ldots, P_n) \equiv 0$ over $\mathbb{Z}$ and e_i is obtained by homomorphically evaluating P_i on c.

Formally, we add to our encryption scheme an algorithm ZT with the following syntax:

Quadratic zero-test: the deterministic polynomial-time quadratic zero-test algorithm ZT takes as input the public parameters pp, a degree-2 polynomial $Q : \mathbb{Z}^n \to \mathbb{Z}$ represented by a list of (possibly non-distinct) monomials with coefficients in $\{1, -1\}$, and evaluated ciphertexts $\mathbf{e} = (e_1, \ldots, e_n)$. It outputs a bit indicating if the test passes or fails.

The zero test satisfies the following completeness and soundness requirements.

Definition 4.4 (Weak Completeness of Zero-Test)**.** *For every $\kappa, \bar{\ell} \in \mathbb{N}$, message $m \in \{0,1\}^{\ell}$ such that $\ell \leq \bar{\ell}$, polynomials $P_1, \ldots, P_n$ such that $P_i : \mathbb{Z}^{\ell} \to \mathbb{Z}$ is of individual degree at most $\bar{\delta}$, and degree-2 polynomial $Q : \mathbb{Z}^n \to \mathbb{Z}$ such that $Q(P_1, \ldots, P_n) \equiv 0$ over $\mathbb{Z}$:*

$$\Pr\left[b = 1 \;\middle|\; \begin{array}{l} \mathsf{pp} \leftarrow \mathsf{ParamGen}(\kappa, \bar{\ell}) \\ \mathsf{sk} \leftarrow \mathsf{KeyGen}(\mathsf{pp}) \\ r \leftarrow \{0,1\}^{\kappa} \\ c \leftarrow \mathsf{Enc}(\mathsf{sk}, m, r) \\ \mathbf{e} = (e_i \leftarrow \mathsf{Eval}(\mathsf{pp}, c, P_i) \;:\; i \in [n]) \\ b \leftarrow \mathsf{ZT}(\mathsf{pp}, Q, \mathbf{e}) \end{array} \right] = 1 \ .$$

Definition 4.5 (Soundness of Zero-Test)**.** *For every set of parameters $\kappa, \bar{\ell} \in \mathbb{N}$, degree-2 polynomial $Q : \mathbb{Z}^n \to \mathbb{Z}$, and vector of evaluated ciphertexts $\mathbf{e} = (e_1, \ldots, e_n)$:*

$$\Pr\left[b = 1 \;\Rightarrow\; \begin{array}{l} \forall i \in [n] : \; m_i \neq \perp \\ Q(m_1, \ldots, m_n) = 0 \end{array} \;\middle|\; \begin{array}{l} \mathsf{pp} \leftarrow \mathsf{ParamGen}(\kappa, \bar{\ell}) \\ \mathsf{sk} \leftarrow \mathsf{KeyGen}(\mathsf{pp}) \\ \forall i \in [n] : \; m_i \leftarrow \mathsf{Val}(\mathsf{sk}, e_i) \\ b \leftarrow \mathsf{ZT}(\mathsf{pp}, Q, \mathbf{e}) \end{array} \right] = 1 \ .$$

In the experiment above, the polynomial Q is evaluated over $\mathbb{F}$ defined by pp.

4.1.3 Multi-key ciphertexts.

As our final step, we extend our zero-testable homomorphic encryption to support homomorphic operations over ciphertexts encrypted under *different* keys. Since we only support arity-one homomorphic evaluation (Eval operates on a single ciphertext), we first introduce the notion of a multi-key ciphertext: a single ciphertext that encrypts multiple messages under multiple secret keys. For simplicity, we only consider ciphertexts encrypting exactly three equal length messages under three different keys.

We also define algorithms that translate between single-key and multi-key ciphertexts. First we define a *ciphertext extension* algorithm that turns a ciphertext c encrypted under a single secret key into a multi-key ciphertext that contains c as well as two additional messages encrypted under two additional secret keys. Importantly, to extend c we do not need its secret key, however, we must know the two additional secret keys.

We also define a *ciphertext restriction* algorithm that can turn a multi-key ciphertext into a ciphertext encrypted under a single key. Since, in general, such restriction would contradict semantic security, we only require the restriction algorithm to operate correctly on certain evaluated ciphertexts. In more detail, consider the following scenario: starting with a multi-key ciphertext Γ encrypting three messages $m_1, m_2, m_3 \in \{0,1\}^{\ell}$ under three secret keys $\mathsf{sk}_1, \mathsf{sk}_2, \mathsf{sk}_3$, we homomorphically evaluate a polynomial $P : \mathbb{Z}^{3\ell} \to \mathbb{Z}$ on Γ to obtain an evaluated ciphertext e. The value encrypted in e may depend on all three messages. Therefore, by semantic security, we should not be able to decrypt it without knowing all three secret keys. However, if the polynomial P depends only on its first ℓ input variables, then we require that it is possible to decrypt e given only sk_1. That is, if there exists a polynomial $P' : \mathbb{Z}^{\ell} \to \mathbb{Z}$ such that $P'(z_1) \equiv P(z_1, z_2, z_3)$, then we can restrict e to an evaluated ciphertext e' that encrypts $P'(m_1)$ under sk_1 alone.

Formally, we add to our encryption scheme algorithms ($\mathsf{MEnc}, \mathsf{Extend}, \mathsf{Restrict}$) with the following syntax:

Multi-key encryption: the probabilistic multi-key encryption algorithm MEnc takes as input a triplet $(\gamma_1, \gamma_2, \gamma_3)$. For every $i \in [3]$, γ_i is a tuple $(\mathsf{sk}_i, m_i, r_i)$ including a secret key sk_i, a message $m_i \in \{0,1\}^\ell$ for $\ell \leq \bar{\ell}$, and randomness $r_i \in \{0,1\}^\kappa$. It outputs a multi-key ciphertext Γ. The running time of MEnc is $\mathrm{poly}(\kappa, \bar{\delta}^{\bar{\ell}})$.

Ciphertext extension: the PPT ciphertext extension algorithm Extend takes as input a triplet $(\gamma_1, \gamma_2, \gamma_3)$. For every $i \in [3]$, γ_i is either a ciphertext c_i or a tuple $(\mathsf{sk}_i, m_i, r_i)$ including a secret key sk_i, a message $m_i \in \{0,1\}^\ell$ for $\ell \leq \bar{\ell}$, and randomness $r_i \in \{0,1\}^\kappa$. We require that exactly one γ_i is a ciphertext and the other two are tuples. It outputs a multi-key ciphertext Γ.

Ciphertext restriction: the deterministic polynomial-time ciphertext restriction algorithm $\mathsf{Restrict}$ takes as input an evaluated ciphertext e and an index $j \in [3]$. It outputs a restricted ciphertext e_j.

Next we define correctness of evaluation and the completeness and soundness of the zero-test for multi-key ciphertexts (since in our quasi-argument we never decrypt multi-key ciphertexts, we need not define correctness of decryption). The definitions are analogous to the definitions for single-key ciphertexts (Definitions 4.1, 4.4, and 4.5). Then we define the correctness of the extension and restriction algorithms.

Definition 4.6 (Correctness of Evaluation for Multi-key Ciphertexts)**.** *For every $\kappa, \bar{\ell} \in \mathbb{N}$, messages $m_1, m_2, m_3 \in \{0,1\}^\ell$ such that $\ell \leq \bar{\ell}$, and polynomial $P : \mathbb{Z}^{3\ell} \to \mathbb{Z}$ of individual degree at most $\bar{\delta}$:*

$$\Pr\left[v = P(m_1, m_2, m_3) \;\middle|\; \begin{array}{l} \mathsf{pp} \leftarrow \mathsf{ParamGen}(\kappa, \bar{\ell}) \\ \forall i \in [3] : \\ \quad r_i \leftarrow \{0,1\}^\kappa \\ \quad \mathsf{sk}_i \leftarrow \mathsf{KeyGen}(\mathsf{pp}) \\ \quad \gamma_i \leftarrow (\mathsf{sk}_i, m_i, r_i) \\ \Gamma \leftarrow \mathsf{MEnc}(\gamma_1, \gamma_2, \gamma_3) \\ e \leftarrow \mathsf{Eval}(\mathsf{pp}, \Gamma, P) \\ v \leftarrow \mathsf{Val}((\mathsf{sk}_1, \mathsf{sk}_2, \mathsf{sk}_3), e) \end{array} \right] = 1 \ .$$

Definition 4.7 (Weak Completeness of Zero-Test for Multi-key Ciphertexts)**.** *For every $\kappa, \bar{\ell} \in \mathbb{N}$, messages $m_1, m_3, m_3 \in \{0,1\}^\ell$ such that $\ell \leq \bar{\ell}$, polynomials $P_1, \ldots, P_n$ such that $P_j : \mathbb{Z}^{3\ell} \to \mathbb{Z}$ is of individual degree at most $\bar{\delta}$, and degree-2 polynomial $Q : \mathbb{Z}^n \to \mathbb{Z}$ such that $Q(P_1, \ldots, P_n) \equiv 0$ over $\mathbb{Z}$:*

$$\Pr\left[b = 1 \;\middle|\; \begin{array}{l} \mathsf{pp} \leftarrow \mathsf{ParamGen}(\kappa, \bar{\ell}) \\ \forall i \in [3] : \\ \quad r_i \leftarrow \{0,1\}^\kappa \\ \quad \mathsf{sk}_i \leftarrow \mathsf{KeyGen}(\mathsf{pp}) \\ \quad \gamma_i \leftarrow (\mathsf{sk}_i, m_i, r_i) \\ \Gamma \leftarrow \mathsf{MEnc}(\gamma_1, \gamma_2, \gamma_3) \\ \mathbf{e} = (e_j \leftarrow \mathsf{Eval}(\mathsf{pp}, \Gamma, P_j) \ : \ j \in [n]) \\ b \leftarrow \mathsf{ZT}(\mathsf{pp}, Q, \mathbf{e}) \end{array} \right] = 1 \ .$$

Definition 4.8 (Soundness of Zero-Test for Multi-key Ciphertexts)**.** *For every $\kappa, \bar{\ell} \in \mathbb{N}$, degree-2 polynomial $Q : \mathbb{Z}^n \to \mathbb{Z}$, and vector of evaluated ciphertexts $\mathbf{e} = (e_1, \ldots, e_n)$:*

$$\Pr\left[b = 1 \ \Rightarrow \ \begin{array}{l} \forall j \in [n] : \ m_j \neq \bot \\ Q(m_1, \ldots, m_n) = 0 \end{array} \;\middle|\; \begin{array}{l} \mathsf{pp} \leftarrow \mathsf{ParamGen}(\kappa, \bar{\ell}) \\ \forall i \in [3] : \mathsf{sk}_i \leftarrow \mathsf{KeyGen}(\mathsf{pp}) \\ \forall j \in [n] : \ m_j \leftarrow \mathsf{Val}((\mathsf{sk}_1, \mathsf{sk}_2, \mathsf{sk}_3), e_j) \\ b \leftarrow \mathsf{ZT}(\mathsf{pp}, Q, \mathbf{e}) \end{array} \right] = 1 \ .$$

We require that the multi-key ciphertext generated by first encrypting a message using Enc, and then extending the resulting ciphertext using Extend, is identical to the multi-key ciphertext generated by MEnc.

Definition 4.9 (Correctness of Extension)**.** *For every $\kappa, \bar{\ell} \in \mathbb{N}$, messages $m_1, m_2, m_3 \in \{0,1\}^{\ell}$ such that $\ell \leq \bar{\ell}$ and index $j \in [3]$:*

$$\Pr\left[\Gamma = \Gamma' \;\middle|\; \begin{array}{l} \mathsf{pp} \leftarrow \mathsf{ParamGen}(\kappa, \bar{\ell}) \\ \forall i \in [3]: \\ \quad r_i \leftarrow \{0,1\}^{\kappa} \\ \quad \mathsf{sk}_i \leftarrow \mathsf{KeyGen}(\mathsf{pp}) \\ \quad \gamma_i \leftarrow (\mathsf{sk}_i, m_i, r_i) \\ \Gamma \leftarrow \mathsf{MEnc}(\gamma_1, \gamma_2, \gamma_3) \\ \forall i \in [3], i \neq j: \\ \quad \gamma'_i \leftarrow \gamma_i \\ \gamma'_j \leftarrow \mathsf{Enc}(\mathsf{sk}_j, m_j, r_j) \\ \Gamma' \leftarrow \mathsf{Extend}(\gamma'_1, \gamma'_2, \gamma'_3) \end{array}\right] = 1 \ .$$

We require that Restrict satisfies the following correctness property.

Definition 4.10 (Correctness of Restriction)**.** *For every $\kappa, \bar{\ell} \in \mathbb{N}$, $\ell \leq \bar{\ell}$, and index $j \in [3]$, messages $m_1, m_2, m_3 \in \{0,1\}^{\ell}$, and polynomials $P : \mathbb{Z}^{3\ell} \to \mathbb{Z}$ and $P' : \mathbb{Z}^{\ell} \to \mathbb{Z}$ of individual degree at most $\bar{\delta}$ such that $\forall z_1, z_2, z_3 \in \mathbb{Z}^{\ell}: \ P(z_1, z_2, z_3) = P'(z_j)$:*

$$\Pr\left[e_j = e'_j \;\middle|\; \begin{array}{l} \mathsf{pp} \leftarrow \mathsf{ParamGen}(\kappa, \bar{\ell}) \\ \forall i \in [3]: \\ \quad r_i \leftarrow \{0,1\}^{\kappa} \\ \quad \mathsf{sk}_i \leftarrow \mathsf{KeyGen}(\mathsf{pp}) \\ \quad \gamma_i \leftarrow (\mathsf{sk}_i, m_i, r_i) \\ c_j \leftarrow \mathsf{Enc}(\mathsf{sk}_j, m_j, r_j) \\ e_j \leftarrow \mathsf{Eval}(\mathsf{pp}, c_j, P') \\ \Gamma \leftarrow \mathsf{MEnc}(\gamma_1, \gamma_2, \gamma_3) \\ e \leftarrow \mathsf{Eval}(\mathsf{pp}, \Gamma, P) \\ e'_j \leftarrow \mathsf{Restrict}(e, j) \end{array}\right] = 1 \ .$$

Putting together our three steps, we get the following definition.

Definition 4.11 (Limited Zero-testable Homomorphic Encryption)**.** *A limited zero-testable homomorphic encryption scheme* $(\mathsf{ParamGen}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{Dec}, \mathsf{Val}, \mathsf{ZT}, \mathsf{MEnc}, \mathsf{Extend}, \mathsf{Restrict})$ *with degree bound $\bar{\delta} \in \mathbb{N}$ satisfies all the requirements in Definitions 4.1 to 4.10.*

4.2 Construction

In this section we construct a limited zero-testable homomorphic encryption from bilinear maps. We begin with notation.

Notation. For a vector of polynomials $\mathbf{C} = (C_1, \ldots, C_\ell) \in (\mathbb{F}[\mathbf{x}])^{\ell}$ and a vector $\boldsymbol{\delta} = (\delta_1, \ldots, \delta_\ell) \in [0, \delta]^{\ell}$, we denote by $\mathbf{C}^{\boldsymbol{\delta}}$ the polynomial $\prod_{i \in [\ell]} C_i^{\delta_i}$.

For every security parameter $\kappa \in \mathbb{N}$, fix a group $G = G_\kappa$ of prime order $p = p(\kappa)$ with a non-degenerate bilinear map $e : G \times G \to G_T$ and let $\mathbb{F} = \mathbb{Z}_p$.

For any $t \in \mathbb{F}$ and $g \in G$, we refer to the element g^t as the *encoding* of t under g and denote it by $\langle t \rangle_g$. For any n-variate polynomial $P(\mathbf{x}) = \sum_{\boldsymbol{\delta} \in [0,\delta]^n} \alpha_{\boldsymbol{\delta}} \cdot \mathbf{x}^{\boldsymbol{\delta}} \in \mathbb{F}[\mathbf{x}]$ of individual degree δ, the encoding of P under g consists of the encodings of its coefficients $\left(\langle \alpha_{\boldsymbol{\delta}} \rangle_g\right)_{\boldsymbol{\delta} \in [0,\delta]^n}$ and we denote it by $\langle P \rangle_g$. Observe that given the encoding $\langle P \rangle_g$ and elements $\mathbf{t} \in \mathbb{F}^n$ we can homomorphically evaluate P on $\mathbf{t}$. That is, we can compute the encoding:

$$\langle P(\mathbf{t}) \rangle_g = \left\langle \sum_{\boldsymbol{\delta} \in [0,\delta]^n} \alpha_{\boldsymbol{\delta}} \cdot \mathbf{t}^{\boldsymbol{\delta}} \right\rangle_g = \prod_{\boldsymbol{\delta} \in [0,\delta]^n} \left(\langle \alpha_{\boldsymbol{\delta}} \rangle_g\right)^{\mathbf{t}^{\boldsymbol{\delta}}} \ .$$

Similarly, we can compute $\langle P(\mathbf{t})\rangle_g$ given P in the clear and the encodings $\left(\langle \mathbf{t}^{\boldsymbol{\delta}}\rangle_g\right)_{\boldsymbol{\delta}\in[0,\delta]^n}$.

Next, we describe the algorithms of our encryption scheme with degree bound $\bar{\delta}$.

The parameter generation algorithm $\mathsf{ParamGen}$**.** Given as input the security parameter κ and a bound $\bar{\ell}$ on the message length, the parameter generation algorithm samples a random group generator $g \leftarrow G$ and outputs the public parameters $\mathsf{pp} = (1^\kappa, \bar{\ell}, g)$. In what follows, let $d = \bar{\ell}\cdot\bar{\delta}$ and $d' = 2d+1$.

The key generation algorithm KeyGen**.** Given as input the public parameters pp, the key generation algorithm samples a random element $s \leftarrow \mathbb{F}$ and outputs the secret key $\mathsf{sk} = (\mathsf{pp}, s)$.

The encryption algorithm Enc**.** Given as input the secret key $\mathsf{sk} = (\mathsf{pp}, s)$, a message $m = (m_1, \ldots, m_\ell)$ and randomness r, the encryption algorithm samples a vector of polynomials $\mathbf{C} = (C_1, \ldots, C_\ell)$ such that $C_i \in \mathbb{F}[x]$ is a random polynomial of degree d' satisfying $C_i(s) = m_i$. It outputs the ciphertext:

$$c = \left(\langle \mathbf{C}^{\boldsymbol{\delta}}\rangle_g\right)_{\boldsymbol{\delta}\in[0,\bar{\delta}]^\ell} .$$

The multi-key encryption algorithm MEnc**.** Given as input a triplet $(\gamma_1, \gamma_2, \gamma_3)$ where every γ_i is a tuple containing a secret key $\mathsf{sk}_i = (\mathsf{pp}, s_i)$, a message $m_i = (m_{i,1}, \ldots, m_{i,\ell})$ and randomness r_i, for every $i \in [3]$, the multi-key encryption algorithm uses the randomness r_i to sample a vector of polynomials $\mathbf{C}_i = (C_{i,1}, \ldots, C_{i,\ell})$ such that $C_{i,j} \in \mathbb{F}[x_i]$ is a random polynomial of degree d' satisfying $C_{i,j}(s_i) = m_{i,j}$. It outputs the multi-key ciphertext:

$$\Gamma = \left(\left\langle (\mathbf{C}_1\ ,\ \mathbf{C}_2\ ,\ \mathbf{C}_3)^{\boldsymbol{\delta}} \in \mathbb{F}[x_1, x_2, x_3]\right\rangle_g\right)_{\boldsymbol{\delta}\in[0,\bar{\delta}]^{3\ell}} .$$

The homomorphic evaluation algorithm Eval**.** The homomorphic evaluation algorithm operates either on single-key or on multi-key ciphertexts. Let $\ell' = \ell$ in the single key case, and $\ell' = 3\ell$ in the multi-key case. It is given as input:

- The public parameters pp.
- A (single-key or multi-key) ciphertext $\left(\langle \mathbf{C}^{\boldsymbol{\delta}}\rangle_g\right)_{\boldsymbol{\delta}\in[0,\bar{\delta}]^{\ell'}}$.
- An ℓ'-variate polynomial P of individual degree $\delta \le \bar{\delta}$ represented by a list of monomials with coefficients in $\{1, -1\}$.

It computes the coefficients $\alpha_{\boldsymbol{\delta}} \in \mathbb{F}$ such that $P(\mathbf{x}) = \sum_{\boldsymbol{\delta}\in[0,\delta]^{\ell'}} \alpha_{\boldsymbol{\delta}}\cdot\mathbf{x}^{\boldsymbol{\delta}}$. It computes and outputs the evaluated ciphertext:

$$e = \langle P(\mathbf{C})\rangle_g = \left\langle \sum_{\boldsymbol{\delta}\in[0,\delta]^{\ell'}} \alpha_{\boldsymbol{\delta}}\cdot\mathbf{C}^{\boldsymbol{\delta}} \right\rangle_g .$$

The decryption algorithm Dec**.** Given as input the secret key $\mathsf{sk} = (\mathsf{pp}, s)$ and an evaluated ciphertext $e = \langle R\rangle_g$ the decryption algorithm first verifies that e encodes a univariate polynomial R of degree at most $\bar{\ell}\cdot\bar{\delta}\cdot d'$ and outputs $\perp$ otherwise. Next, it computes the encoding $\langle R(s)\rangle_g$. If $\langle R(s)\rangle_g = \langle b\rangle_g$ for $b \in \{0,1\}$ it outputs b. Otherwise, it outputs $\perp$.

The inefficient value recovery algorithm Val**.** The inefficient value recovery algorithm operates either on single-key or on multi-key evaluated ciphertexts. In the single-key case it is given as input the secret key $\mathsf{sk} = (\mathsf{pp}, s)$ and an evaluated ciphertext $e = \langle R \rangle_g$. In the multi-key case it is given as input the secret keys $(\mathsf{sk}_1, \mathsf{sk}_2, \mathsf{sk}_3)$ where $\mathsf{sk}_i = (\mathsf{pp}, s_i)$ and an evaluated ciphertext $e = \langle R \rangle_g$. Let $\mathbf{s} = (s)$ in the single key case and $\mathbf{s} = (s_1, s_2, s_3)$ in the multi-key case. The inefficient value recovery algorithm first verifies that e encodes a polynomial R of degree at most $\bar{\ell} \cdot \bar{\delta} \cdot d'$ that is univariate in the single-key case and trivariate in the multi-key case, and outputs $\perp$ otherwise. Next, it computes the encoding $\langle R(\mathbf{s}) \rangle_g$. It inefficiently finds $v \in \mathbb{F}$ such that $\langle R(\mathbf{s}) \rangle_g = \langle v \rangle_g$ and outputs v.

The quadratic zero-test algorithm ZT**.** The quadratic zero-test algorithm is given as input:

- The public parameters pp.
- An n-variate degree-2 polynomial Q represented by a list of monomials with coefficients in $\{1, -1\}$.
- A vector of evaluated ciphertexts $\mathbf{e} = (e_1, \ldots, e_n)$ where $e_i = \langle R_i \rangle_g$.

It first verifies that every e_i encodes a polynomial R_i of degree at most $\bar{\ell} \cdot \bar{\delta} \cdot d'$ that is univariate in the single-key case and trivariate in the multi-key case, and outputs $\perp$ otherwise. Next, it computes the coefficients $\alpha_{i,j} \in \mathbb{F}$ such that $Q(x_1, \ldots, x_n) = \sum_{i,j \in [0,n]} \alpha_{i,j} \cdot x_i x_j$. Using the bilinear map it computes the encoded polynomial:

$$\langle Q(R_1, \ldots, R_n) \rangle_{e(g,g)} = \left\langle \sum_{i,j \in [0,n]} \alpha_{i,j} \cdot R_i R_j \right\rangle_{e(g,g)} ,$$

and outputs 1 if $\langle Q(R_1, \ldots, R_n) \rangle_{e(g,g)} = \langle \mathbf{0} \rangle_{e(g,g)}$ (that is, if all the encoded coefficients are zero). Otherwise, it outputs 0.

The ciphertext extension algorithm Extend**.** Given as input a triplet $(\gamma_1, \gamma_2, \gamma_3)$, where one γ_j is a ciphertext $c_j = \left(\left\langle \mathbf{C}_j^{\boldsymbol{\delta}} \right\rangle_g \right)_{\boldsymbol{\delta} \in [0, \bar{\delta}]^\ell}$ and for $i \neq j$, γ_i is a tuple containing a secret key $\mathsf{sk}_i = (\mathsf{pp}, s_i)$, a message $m_i = (m_{i,1}, \ldots, m_{i,\ell})$ and randomness r_i, for every $i \neq j$, the ciphertext extension algorithm uses the randomness r_i to sample a vector of polynomials $\mathbf{C}_i = (C_{i,1}, \ldots, C_{i,\ell})$ such that $C_{i,k} \in \mathbb{F}[x_i]$ is a random polynomial of degree d' satisfying $C_{i,k}(s_i) = m_{i,k}$. It computes and outputs the multi-key ciphertext:

$$\Gamma = \left(\left\langle (\mathbf{C}_1 \ , \ \mathbf{C}_2 \ , \ \mathbf{C}_3)^{\boldsymbol{\delta}} \in \mathbb{F}[x_1, x_2, x_3] \right\rangle_g \right)_{\boldsymbol{\delta} \in [0, \bar{\delta}]^{3\ell}} .$$

Note that this encoded polynomial can indeed be computed given c_j since $\mathbf{C}_i$ for $i \neq j$ are not encoded.

The ciphertext restriction algorithm $\mathsf{Restrict}$**.** Given as input an evaluated ciphertext $e = \langle R \in \mathbb{F}[x_1, x_2, x_3] \rangle_g$ and an index $j \in [3]$, the ciphertext restriction algorithm computes and outputs the restricted ciphertext: $e_j = \langle R_j \rangle_g$ where $R_j \in \mathbb{F}[x_j]$ is the restriction of R to $x_i = 0$ for every $i \neq j$.

4.3 Analysis

We prove that the construction given in Section 4.2 with degree bound $\bar{\delta}$ is a limited zero-testable homomorphic encryption scheme (Definition 4.11) under the following hardness assumption parameterized by $\bar{\delta}$. For our quasi-argument we set $\bar{\delta} = 2$.

Assumption 4.12. *There exists an ensemble of groups $\{G_\kappa\}$ of prime order $p = p_\kappa$ with a non-degenerate bilinear map such that for every $d(\kappa) = O(\log \kappa)$ and PPT adversary* Adv*, there exists a negligible function*

μ *such that for every* $\kappa \in \mathbb{N}$*:*

$$\Pr\left[b' = b \;\middle|\; \begin{array}{l} b \leftarrow \{0,1\} \\ g \leftarrow G \\ s \leftarrow \mathbb{Z}_p \\ t_0 \leftarrow \mathbb{Z}_p \\ t_1 \leftarrow s^{2d+1} \\ b' \leftarrow \mathsf{Adv}\left(\left(\left\langle s^i \cdot t_b^j \right\rangle_g\right)_{i\in[0,d],j\in[0,\bar{\delta}]}\right) \end{array}\right] \leq \frac{1}{2} + \mu(\kappa) \ .$$

Theorem 4.13. *For every* $\bar{\delta} \in \mathbb{N}$*, the scheme* (ParamGen, KeyGen, Enc, Eval, Dec, Val, ZT, MEnc, Extend, Restrict) *given in Section 4.2 is a limited zero-testable homomorphic encryption scheme with degree bound* $\bar{\delta}$ *(Definition 4.11) under Assumption 4.12 with parameter* $\bar{\delta}$*.*

In the remainder of this section we prove Theorem 4.13. We focus on proving semantic security (Definition 4.3). The rest of the requirements in Definition 4.11 follow directly from the construction.

Assume towards contradiction that there exists a function $\bar{\ell}(\kappa) = O(\log \kappa)$ and PPT adversary Adv such that for infinitely many $\kappa \in \mathbb{N}$, there exists $\ell \leq \bar{\ell}(\kappa)$ and $m^0, m^1 \in \{0,1\}^\ell$ such that:

$$\Pr\left[b' = b \;\middle|\; \begin{array}{l} b \leftarrow \{0,1\} \\ \mathsf{pp} \leftarrow \mathsf{ParamGen}(\kappa, \bar{\ell}(\kappa)) \\ \mathsf{sk} \leftarrow \mathsf{KeyGen}(\mathsf{pp}) \\ r \leftarrow \{0,1\}^\kappa \\ c_0 \leftarrow \mathsf{Enc}(\mathsf{sk}, m^0, r) \\ c_1 \leftarrow \mathsf{Enc}(\mathsf{sk}, m^1, r) \\ b' \leftarrow \mathsf{Adv}(\mathsf{pp}, c_b) \end{array}\right] \geq \frac{1}{2} + \frac{1}{\mathrm{poly}(\kappa)} \ .$$

Fix such κ, ℓ and m^0, m^1. For $i \in [0,\ell]$, let $\bar{m}^i \in \{0,1\}^\ell$ be the message such that $\bar{m}^i_j = m^1_j$ for $j \leq i$ and $\bar{m}^i_j = m^0_j$ for $j > i$. Therefore, there exists $i^* \in [\ell]$ such that:

$$\Pr\left[b' = b \;\middle|\; \begin{array}{l} b \leftarrow \{0,1\} \\ \mathsf{pp} \leftarrow \mathsf{ParamGen}(\kappa, \bar{\ell}(\kappa)) \\ \mathsf{sk} \leftarrow \mathsf{KeyGen}(\mathsf{pp}) \\ r \leftarrow \{0,1\}^\kappa \\ c_0 \leftarrow \mathsf{Enc}(\mathsf{sk}, \bar{m}^{i^*-1}, r) \\ c_1 \leftarrow \mathsf{Enc}(\mathsf{sk}, \bar{m}^{i^*}, r) \\ b' \leftarrow \mathsf{Adv}(\mathsf{pp}, c_b) \end{array}\right] \geq \frac{1}{2} + \frac{1}{\mathrm{poly}(\kappa)} \ . \tag{3}$$

Fix such $i^* \in [\ell]$. Next, we construct an adversary breaking Assumption 4.12 with parameter $\bar{\delta}$ and $d = \bar{\delta} \cdot \bar{\ell}$. To this end, we first construct an adversary Adv'_m for any $m \in \{0,1\}^\ell$ such that:

$$\Pr\left[b = 1 \;\middle|\; \begin{array}{l} g \leftarrow G \\ s \leftarrow \mathbb{F} \\ t \leftarrow s^{2d+1} \\ b \leftarrow \mathsf{Adv}'_m\left(\left(\left\langle s^i \cdot t^j \right\rangle_g\right)_{\substack{i\in[0,d] \\ j\in[0,\bar{\delta}]}}\right) \end{array}\right] = \Pr\left[b = 1 \;\middle|\; \begin{array}{l} \mathsf{pp} \leftarrow \mathsf{ParamGen}(\kappa, \bar{\ell}(\kappa)) \\ \mathsf{sk} \leftarrow \mathsf{KeyGen}(\mathsf{pp}) \\ r \leftarrow \{0,1\}^\kappa \\ c \leftarrow \mathsf{Enc}(\mathsf{sk}, m, r) \\ b \leftarrow \mathsf{Adv}(\mathsf{pp}, c) \end{array}\right] \ . \tag{4}$$

Additionally:

$$\Pr_{\substack{g\leftarrow G \\ s,t\leftarrow\mathbb{F}}}\left[1 = \mathsf{Adv}'_{\bar{m}^{i^*}}\left(\left(\left\langle s^i \cdot t^j \right\rangle_g\right)_{\substack{i\in[0,d] \\ j\in[0,\bar{\delta}]}}\right)\right] = \Pr_{\substack{g\leftarrow G \\ s,t\leftarrow\mathbb{F}}}\left[1 = \mathsf{Adv}'_{\bar{m}^{i^*-1}}\left(\left(\left\langle s^i \cdot t^j \right\rangle_g\right)_{\substack{i\in[0,d] \\ j\in[0,\bar{\delta}]}}\right)\right] \ . \tag{5}$$

Then by (3), for some $m \in \left\{\bar{m}^{i^*}, \bar{m}^{i^*-1}\right\}$, Adv'_m breaks Assumption 4.12.

Given as input the encodings $\left\langle s^i \cdot t^j \right\rangle_g$ for every $i \in [0,d]$ and $j \in [0,\bar{\delta}]$, the adversary Adv'_m proceeds as follows:

1. Let $\mathsf{pp} = (1^\kappa, \bar{\ell}, g)$ and $d' = 2d+1$.
2. For every $i \in [\ell]$ sample a random polynomial $C'_i \in \mathbb{F}[x]$ of degree $d'-1$.
3. For $i \in [\ell] \setminus \{i^*\}$ let $C_i = x \cdot C'_i - s \cdot C'_i + m_i$. Observe that C_i is distributed like a random degree-d' polynomial subject to $C_i(s) = m_i$. Note that the adversary cannot compute the polynomial C_i since it is not given s in the clear.
4. Let $C_{i^*} = x \cdot C'_{i^*} - s \cdot C'_{i^*} + m_{i^*} + (x^{d'} - t)$. Observe that if $t = s^{d'}$ then C_i is distributed like a random degree-d' polynomial subject to $C_i(s) = m_{i^*}$. If t is random and independent of s then C_i is distributed like a random degree-d' polynomial.
5. Use the encodings $\left\langle s^i \cdot t^j \right\rangle_g$ for $i \in [0,d]$ and $j \in [0,\bar{\delta}]$ to compute the ciphertext:

$$c = \left(\left\langle (C_1, \ldots, C_\ell)^{\boldsymbol{\delta}} \right\rangle_g\right)_{\boldsymbol{\delta} \in [0,\bar{\delta}]^\ell} .$$

   Observe that for every $i \in [\ell]$, every coefficient of C_i depends linearly on s. Moreover, the free coefficient of C_{i^*} depends linearly on t. Therefore, every encoded polynomial in c can indeed be computed from the input.
6. Output the bit returned by $\mathsf{Adv}(\mathsf{pp}, c)$.

By construction, if $t = s^{d'}$ then the ciphertext c generated by Adv'_m is distributed exactly like an encryption of m, and hence (4) follows. If t is random and independent of s then the output of Adv'_m is independent of m_{i^*}. Since $\bar{m}^{i^*-1}$ and $\bar{m}^{i^*}$ only differ in location i^*, (5) follows.

5 Quasi-argument

This section includes our publicly verifiable non-interactive quasi-argument. We start by defining the notion of quasi-arguments. In Section 5.1 we construct a quasi-argument from the encryption scheme in Section 4. The analysis is given in Sections 5.2 and 5.3. See also Section 2.1 for a detailed discussion of the notion, and Section 2.3 for an the overview of the construction.

Syntactically, quasi-arguments are similar to the standard notion of succinct non-interactive arguments. In a quasi-argument for a 3CNF formula φ, the prover and verifier share an input x. Using a witness w such that $\varphi(x,w) = 1$ the prover produces an accepting proof. In contrast to standard arguments, quasi-arguments only satisfy a relaxed notion of soundness, called no-signaling extraction. Loosely speaking, in the no-signaling extraction requirement we consider an adaptive adversary acting as the prover that given honestly generated keys, produces an input x for φ as well as some arbitrary auxiliary input y together with a proof. For a quasi-argument with locality parameter K we require that there exists a no-signaling extractor E, that takes as input the formula φ along with a subset $\mathbf{S}$ of φ's variables of size at most K and samples inputs x, y, together with a partial assignment to the variable in $\mathbf{S}$.

We make the following requirements of the extractor E. First, for every formula φ and set $\mathbf{S}$, the distribution of inputs x, y sampled by E must be indistinguishable from the distribution samples by the adversary, conditioned on it producing an accepting proof. More formally, we consider an experiment where the adversary produces inputs x, y together with a proof, and if the proof is rejecting we replace x and y with $\perp$. We require that this is indistinguishable from x, y sampled by E. Second, we require that for every formula φ and set $\mathbf{S}$, whenever E samples $x \neq \perp$, the partial assignment sampled by E is consistent with x and it satisfies all of φ's clauses that are over the variables in $\mathbf{S}$. Finally, the output distribution of E must satisfy the no-signaling requirement.

Formally, a quasi-argument consists of algorithms $(\mathsf{QA.S}, \mathsf{QA.P}, \mathsf{QA.V})$ with the following syntax.

Setup: The randomized setup algorithm $\mathsf{QA.S}$ takes as input a security parameter κ, an M-variate 3CNF formula φ, an input length n and a locality parameter K. It outputs a prover key pk and a verifier key vk.

Prover: The deterministic prover algorithm $\mathsf{QA.P}$ takes as input a prover key pk and an assignment $\sigma : [M] \to \{0,1\}$. It outputs a proof Π.

Verifier: The deterministic verifier algorithm $\mathsf{QA.V}$ takes as input a verifier key vk, an input $x \in \{0,1\}^n$ and a proof Π. It outputs a bit indicating if it accepts or rejects.

Remark 5.1 (Formula structure). For technical reasons, we require that every clause of φ contains 3 distinct variables. This is WLOG, as we explain in Section 5.1. In what follows, we always assume that φ satisfies this property even if we do not require so explicitly.

The definition of quasi-arguments relies on the following notion of locally satisfying assignments. In what follows, we denote the M variables of the formula φ by $z_1, \ldots, z_M$ and assume WLOG that for $i \in [n]$, the i-th input variable is z_i.

Definition 5.2 (Locally Satisfying Assignment)**.** *For an M-variate 3CNF formula ϕ, and a set $\mathbf{S} \subseteq [M]$ we say that a partial assignment $\sigma : \mathbf{S} \to \{0,1\}$ locally satisfies ϕ if every clause in ϕ that only contains variables in $\mathbf{S}$ is satisfied by σ. We denote by $\phi(\sigma)$ the bit indicating whether or not σ locally satisfies ϕ.*

Definition 5.3 (Quasi-arguments)**.** *A publicly verifiable non-interactive quasi-argument* $(\mathsf{QA.S}, \mathsf{QA.P}, \mathsf{QA.V})$ *satisfies the following requirements.*

Completeness. *For every $\kappa, K, n, M \in \mathbb{N}$ such that $K, n \leq M \leq 2^\kappa$, every M-variate 3CNF formula φ, assignment $\sigma : [M] \to \{0,1\}$ satisfying φ, and for $x \in \{0,1\}^n$ such that $\forall i \in [n] : \sigma(i) = x_i$:*

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

Efficiency. *In the completeness experiment above:*

- *The setup algorithm runs in time* $\mathrm{poly}(\kappa, |\varphi|)$ *and outputs a verifier key* vk *of length* $n \cdot \mathrm{poly}(\kappa, K)$.
- *The prover runs in polynomial time and it outputs a proof* Π *of length* $\mathrm{poly}(\kappa, K)$.
- *The verifier runs in time* $n \cdot \mathrm{poly}(\kappa, K)$.

No-signaling extraction. *For every polynomials M, n, K in the security parameter, there exists a PPT oracle machine E, called the no-signaling extractor, such that for any M-variate formula $\varphi = \varphi_\kappa$ and poly-size adversary Adv the following requirements are satisfied:*

Correct distribution: *For every PPT distinguisher D there exists a negligible function μ such that for every $\kappa \in \mathbb{N}$:*

$$\left|\Pr\left[\mathsf{D}(x,y) = 1 \;\middle|\; \begin{array}{l} (\mathsf{pk}, \mathsf{vk}) \leftarrow \mathsf{QA.S}(\kappa, \varphi, n, K) \\ (x, y, \Pi) \leftarrow \mathsf{Adv}(\mathsf{pk}, \mathsf{vk}) \\ \textit{if } \mathsf{QA.V}(\mathsf{vk}, x, \Pi) = 0 : \\ \quad \textit{set } x = y = \bot \end{array}\right] - \Pr_{(x,y,\sigma) \leftarrow \mathsf{E}^{\mathsf{Adv}}(\varphi, \emptyset)}[\mathsf{D}(x,y) = 1]\right| \leq \mu(\kappa) \ .$$

Local consistency: *There exists a negligible function μ such that for every $\kappa \in \mathbb{N}$ and set $\mathbf{S} \subseteq [M]$ of size at most K:*

$$\Pr_{(x,y,\sigma) \leftarrow \mathsf{E}^{\mathsf{Adv}}(\varphi, \mathbf{S})}\left[x = \bot \quad \vee \quad \begin{array}{l} \forall i \in \mathbf{S} \cap [n] : \ \sigma(i) = x_i \\ \varphi(\sigma) = 1 \end{array}\right] \geq 1 - \mu(\kappa) \ .$$

No-signaling: *For every PPT distinguisher* D *there exists a negligible function* μ *such that for every* $\kappa \in \mathbb{N}$ *and sets* $\mathbf{S}' \subseteq \mathbf{S} \subseteq [M]$ *of size at most* K*:*

$$\left| \Pr_{(x,y,\sigma)\leftarrow \mathsf{E}^{\mathsf{Adv}}(\varphi,\mathbf{S})} [\mathsf{D}(x,y,\sigma(\mathbf{S}')) = 1] - \Pr_{(x,y,\sigma')\leftarrow \mathsf{E}^{\mathsf{Adv}}(\varphi,\mathbf{S}')} [\mathsf{D}(x,y,\sigma') = 1] \right| \leq \mu(\kappa) \ .$$

Remark 5.4 (Universal Extractor). Our definition requires that the no-signaling extractor is universal. That is, there exists a single no-signaling extractor that works for every ensemble of formulas $\{\varphi_\kappa\}$ and adversary Adv. We rely on this fact in the proof of our bootstrapping theorem in Section 6.

In the remainder of this section, we prove the following theorem.

Theorem 5.5. *Under Assumption 4.12 the construction given in Section 5.1 is a quasi-argument.*

5.1 Construction

In this section we construct a quasi-argument based on the encryption scheme in Section 4.

5.1.1 Notation.

We start by introducing notation.

The input formula. Recall that the local-consistency property of the no-signaling extractor E requires that if $\mathsf{E}(\varphi, \mathbf{S})$ samples (x, y, σ), then σ locally satisfies φ and assigns to every input variable in $\mathbf{S}$ a value consistent with x. In our construction it will be convenient to express the input constraints on the assignment σ as additional clauses in φ. For example, for every $i \in [n]$ we could add the clause $(z_i = x_i)$ to φ. Recall, however, that we assume every clause in φ contains 3 distinct variables (see Remark 5.1). We therefore add to φ clauses that express the input constraints and have the required structure. This is done as follows.

Out of the M variables $z_1, \ldots, z_M$ fix two variables z', z'' that are not input variables (we assume WLOG that $M \geq n + 2$). For every $i \in [n]$ and $b \in \{0, 1\}$ let $I_{i,b}(z_i, z', z'')$ be the following formula that is satisfied if and only if $z_i = b$:

$$I_{i,b}(z_i, z', z'') = \bigwedge_{b',b''\in\{0,1\}} (z_i = b \ \vee \ z' = b' \ \vee \ z'' = b'') \ .$$

For $x \in \{0, 1\}^n$ we define the input formula $I_x = \bigwedge_{i\in[n]} I_{i,x_i}$. We would like to argue that for any local assignment $\sigma : \mathbf{S} \to \{0, 1\}$, if $I_x(\sigma) = 1$ then every input variable in $\mathbf{S}$ is assigned consistently with x. However, this is only the case when $\mathbf{S}$ contains z', z''. Therefore, in what follows we assume that $\mathbf{S}$ contains the variables z', z''. This is WLOG since we can always:

- Run the setup algorithm $\mathsf{QA.S}$ with locality parameter $K + 2$ instead of K.
- Invoke the no-signaling extractor E on the set $\mathbf{S} \cup \{z', z''\}$ instead of $\mathbf{S}$ and ignore the additional variables in the resulting assignment.

Finally, we assume WLOG that for every $x \in \{0, 1\}^n$, I_x and φ have no clauses in common.

Formula arithmetization. We represent 3CNF formulas as multi-linear polynomials. Let ϕ be an M-variate 3CNF formula over the variables $z_1, \ldots, z_M$. We identify the variables' indices with strings in $\{0, 1\}^m$ for $m = \log(M)$.

Definition 5.6. *A multi-linear polynomial* $\tilde{\phi} : \mathbb{Z}^{3m+3} \to \mathbb{Z}$ *is an arithmetization of the 3CNF formula* ϕ *if the following holds: For every set of indices* $\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3 \in \{0, 1\}^m$ *and bits* $b_1, b_2, b_3 \in \{0, 1\}$*,* $\tilde{\phi}(\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3, b_1, b_2, b_3)$ *outputs a bit* $b \in \{0, 1\}$ *such that* $b = 1$ *if and only if* φ *contains the clause:* $(z_{\mathbf{v}_1} = b_1 \vee z_{\mathbf{v}_2} = b_2 \vee z_{\mathbf{v}_3} = b_3)$*.*

For every $j \in [n]$ and $b \in \{0, 1\}$, let $\tilde{I}_{j,b}$ be an arithmetization of the formula $I_{j,b}$. Therefore, for any $x \in \{0, 1\}^n$, the polynomial $\tilde{I}_x = \sum_{j\in[n]} \tilde{I}_{j,x_j}$ is an arithmetization of the formula I_x. Since φ and I_x do not have any common clauses, if $\tilde{\varphi}$ is an arithmetization of φ, then $\tilde{\varphi} + \tilde{I}_x$ is an arithmetization of the formula $\varphi \wedge I_x$.

Polynomials. For a fixed $\ell \in \mathbb{N}$, let ID denote the multi-linear polynomial extending the boolean equality function:

$$\mathsf{ID}(z_1, \ldots, z_\ell, z'_1, \ldots, z'_\ell) \equiv \prod_{i \in [\ell]} (z_i z'_i + (1 - z_i)(1 - z'_i)) \ .$$

For $i \in [\ell]$, $j \in \mathbb{N}$ let $Z_i^j : \mathbb{Z}^\ell \to \mathbb{Z}$ denote the monomial:

$$Z_i^j(z_1, \ldots, z_\ell) \equiv z_i^j \ .$$

For any polynomial $P : \mathbb{Z}^\ell \to \mathbb{Z}$ of individual degree δ and any $i \in [\ell]$, let $P|_{i,0}, \ldots, P|_{i,\delta} : \mathbb{Z}^\ell \to \mathbb{Z}$ be the polynomials such that:

$$P(\mathbf{z}) \equiv \sum_{j \in [0,\delta]} P|_{i,j}(\mathbf{z}) \cdot Z_i^j(\mathbf{z}) \ .$$

Namely, $P|_{i,0}, \ldots, P|_{i,\delta}$ are defined by interpreting P as a univariate polynomial in $\mathbf{z}_i$, and letting $P|_{i,j}(\mathbf{z})$ be the coefficient of $\mathbf{z}_i^j$. (In particular, each $P|_{i,j}(\mathbf{z})$ ignores the variable $\mathbf{z}_i$.)

Triplets. Let $\langle K \rangle^3$ be the set of all triplets $(q_1, q_2, q_3) \in [K]^3$ such that q_1, q_2, q_3 are pairwise distinct.

The encryption scheme. We use the encryption scheme from Section 4 with degree bound $\bar{\delta} = 2$ given by the algorithms $(\mathsf{ParamGen}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{Dec}, \mathsf{Val}, \mathsf{ZT}, \mathsf{MEnc}, \mathsf{Extend}, \mathsf{Restrict})$. (The algorithms $\mathsf{Dec}, \mathsf{Val}$ and Extend are used only in the analysis.) We use the following shorthand for the algorithms Eval and ZT:

- We denote by $[P(c)]_{\mathsf{pp}}$ the output of $\mathsf{Eval}(\mathsf{pp}, c, P)$.
- We denote by $[P(\mathbf{e}) = P'(\mathbf{e})]_{\mathsf{pp}}$ the output of $\mathsf{ZT}(\mathsf{pp}, (P - P'), \mathbf{e})$.

5.1.2 The algorithms.

Next we describe the quasi-argument's algorithms $(\mathsf{QA.S}, \mathsf{QA.P}, \mathsf{QA.V})$.

The setup algorithm $\mathsf{QA.S}$. The setup algorithm is given as input a security parameter κ, an M-variate 3CNF formula φ, an input length n and a locality parameter K.

Let $m = \log(M)$ (we assume WLOG that m is an integer). Let $\tilde{\varphi}$ be an arithmetization of the formula φ. The setup algorithm proceeds as follows:

- Sample public parameters $\mathsf{pp} \leftarrow \mathsf{ParamGen}(\kappa, \bar{\ell} = m)$.
- For every $q \in [K]$ sample $\mathsf{sk}_q \leftarrow \mathsf{KeyGen}(\mathsf{pp}), r_q \leftarrow \{0,1\}^\kappa$ and set $\gamma_q = (\mathsf{sk}_q, 0^m, r_q)$.
- For every $q \in [K]$ set $A^q \leftarrow \mathsf{Enc}(\gamma_q)$.
- For every $\mathbf{q} = (q_1, q_2, q_3) \in \langle K \rangle^3$ set $B^{\mathbf{q}} \leftarrow \mathsf{MEnc}(\gamma_{q_1}, \gamma_{q_2}, \gamma_{q_3})$.
- For every $\mathbf{q} \in \langle K \rangle^3$, $i \in [3m]$, and $\delta \in [0,2]$, set $C_{i,\delta}^{\mathbf{q}} \leftarrow \left[Z_i^\delta(B^{\mathbf{q}})\right]_{\mathsf{pp}}$.
- For every $\mathbf{b} \in \{0,1\}^3$, $\mathbf{q} \in \langle K \rangle^3$ set $D^{\mathbf{b},\mathbf{q}} \leftarrow [\tilde{\varphi}(B^{\mathbf{q}}, \mathbf{b})]_{\mathsf{pp}}$.
- For every $\mathbf{b} \in \{0,1\}^3$, $\mathbf{q} \in \langle K \rangle^3$, $j \in [n]$, and $b \in \{0,1\}$, set $E_{j,b}^{\mathbf{b},\mathbf{q}} \leftarrow \left[\tilde{I}_{j,b}(B^{\mathbf{q}}, \mathbf{b})\right]_{\mathsf{pp}}$.

- Output the prover and verifier keys:

$$\mathsf{pk} = \left(\begin{array}{lll} \mathsf{pp} & & \\ \{A^q & : & q \in [K]\} \\ \{B^{\mathbf{q}} & : & \mathbf{q} \in \langle K \rangle^3\} \end{array} \right)$$

$$\mathsf{vk} = \left(\begin{array}{lll} \mathsf{pp} & & \\ \left\{C^{\mathbf{q}}_{i,\delta} \right. & : & \left. \mathbf{q} \in \langle K \rangle^3,\ i \in [3m],\ \delta \in [0,2] \right\} \\ \left\{D^{\mathbf{b},\mathbf{q}} \right. & : & \left. \mathbf{b} \in \{0,1\}^3,\ \mathbf{q} \in \langle K \rangle^3 \right\} \\ \left\{E^{\mathbf{b},\mathbf{q}}_{j,b} \right. & : & \left. \mathbf{b} \in \{0,1\}^3,\ \mathbf{q} \in \langle K \rangle^3,\ j \in [n],\ b \in \{0,1\} \right\} \end{array} \right) .$$

The prover algorithm QA.P**.** The prover is given as input:

- A prover key:

$$\mathsf{pk} = \left(\begin{array}{lll} \mathsf{pp} & & \\ \{A^q & : & q \in [K]\} \\ \{B^{\mathbf{q}} & : & \mathbf{q} \in \langle K \rangle^3\} \end{array} \right) .$$

- An assignment $\sigma : [M] \to \{0,1\}$ satisfying φ.

The prover proceeds as follows:

- Let $\Sigma : \mathbb{Z}^m \to \mathbb{Z}$ be the multi-linear extension of the assignment σ:

$$\Sigma(\mathbf{z}) \equiv \sum_{\mathbf{v} \in \{0,1\}^m} \mathsf{ID}(\mathbf{z}, \mathbf{v}) \cdot \sigma(\mathbf{v}) \ .$$

- For every $\mathbf{b} = (b_1, b_2, b_3) \in \{0,1\}^3$ and subset $I \subseteq [3]$, let $\Sigma^{\mathbf{b}}_I : \mathbb{Z}^{3m} \to \mathbb{Z}$ be the polynomial:

$$\Sigma^{\mathbf{b}}_I(\mathbf{z}_1, \mathbf{z}_2, \mathbf{z}_3) \equiv \prod_{i \in I} (\Sigma(\mathbf{z}_i) - b_i) \ .$$

- Let $x \in \{0,1\}^n$ be the value σ assigns to φ's input variables. Note that σ satisfies $\varphi \wedge I_x$.
- For every $\mathbf{b} \in \{0,1\}^3$ let $P^{\mathbf{b}}_0 : \mathbb{Z}^{3m} \to \mathbb{Z}$ be the polynomial:

$$P^{\mathbf{b}}_0(\mathbf{z}) \equiv (\tilde{\varphi} + \tilde{I}_x)(\mathbf{z}, \mathbf{b}) \cdot \Sigma^{\mathbf{b}}_{[3]}(\mathbf{z}) \ .$$

  Note that since σ satisfies $\varphi \wedge I_x$, $P^{\mathbf{b}}_0(\mathbf{z}) = 0$ for every $\mathbf{z} \in \{0,1\}^{3m}$.
- For every $\mathbf{b} \in \{0,1\}^3$ and $i \in [3m]$, let $P^{\mathbf{b}}_i : \mathbb{Z}^{3m} \to \mathbb{Z}$ be the polynomial:

$$P^{\mathbf{b}}_i(z_1, \ldots, z_{3m}) \equiv \sum_{v_1, \ldots, v_i \in \{0,1\}} \mathsf{ID}(z_1, \ldots, z_i, v_1, \ldots, v_i) \cdot P^{\mathbf{b}}_0(v_1, \ldots, v_i, z_{i+1}, \ldots, z_{3m}) \ .$$

  Note that for every $i \in [3m]$:

$$P^{\mathbf{b}}_i(z_1, \ldots, z_{3m}) \equiv \sum_{v_i \in \{0,1\}} \mathsf{ID}(z_i, v_i) \cdot P^{\mathbf{b}}_{i-1}(z_1, \ldots, z_{i-1}, v_i, z_{i+1}, \ldots, z_{3m}) \ .$$

  Moreover, since $P^{\mathbf{b}}_0(\mathbf{z}) = 0$ for every $\mathbf{z} \in \{0,1\}^{3m}$ we have that $P^{\mathbf{b}}_{3m} \equiv 0$.
- For every $q \in [K]$ set $F^q \leftarrow [\Sigma(A^q)]_{\mathsf{pp}}$.
- For every $\mathbf{b} \in \{0,1\}^3$, $\mathbf{q} \in \langle K \rangle^3$, and non-empty $I \subseteq [3]$, set $G^{\mathbf{b},\mathbf{q}}_I \leftarrow \left[\Sigma^{\mathbf{b}}_I(B^{\mathbf{q}})\right]_{\mathsf{pp}}$.

- For every $\mathbf{b} \in \{0,1\}^3$, $\mathbf{q} \in \langle K \rangle^3$, $i \in [3m]$, and $\delta \in [0,2]$, set $H^{\mathbf{b},\mathbf{q}}_{i-1,\delta} \leftarrow \left[P^{\mathbf{b}}_{i-1}\big|_{i,\delta}(B^{\mathbf{q}}) \right]_{\mathsf{pp}}$.
- Output the proof:

$$\Pi = \left(\begin{array}{ll} \{F^q & : \ q \in [K]\} \\ \left\{G^{\mathbf{b},\mathbf{q}}_I & : \ \mathbf{b} \in \{0,1\}^3, \ \mathbf{q} \in \langle K \rangle^3, \ I \subseteq [3] \right\} \\ \left\{H^{\mathbf{b},\mathbf{q}}_{i-1,\delta} & : \ \mathbf{b} \in \{0,1\}^3, \ \mathbf{q} \in \langle K \rangle^3, \ i \in [3m], \ \delta \in [0,2] \right\} \end{array} \right) .$$

The verifier algorithm $\mathsf{QA.V}$. The verifier is given as input:

- A verifier key:

$$\mathsf{vk} = \left(\begin{array}{ll} \mathsf{pp} & \\ \left\{C^{\mathbf{q}}_{i,\delta} & : \ \mathbf{q} \in \langle K \rangle^3, \ i \in [3m], \ \delta \in [0,2] \right\} \\ \left\{D^{\mathbf{b},\mathbf{q}} & : \ \mathbf{b} \in \{0,1\}^3, \ \mathbf{q} \in \langle K \rangle^3 \right\} \\ \left\{E^{\mathbf{b},\mathbf{q}}_{j,b} & : \ \mathbf{b} \in \{0,1\}^3, \ \mathbf{q} \in \langle K \rangle^3, \ j \in [n], \ b \in \{0,1\} \right\} \end{array} \right) .$$

- An input string $x \in \{0,1\}^n$.
- A proof:

$$\Pi = \left(\begin{array}{ll} \{F^q & : \ q \in [K]\} \\ \left\{G^{\mathbf{b},\mathbf{q}}_I & : \ \mathbf{b} \in \{0,1\}^3, \ \mathbf{q} \in \langle K \rangle^3, \ I \subseteq [3] \right\} \\ \left\{H^{\mathbf{b},\mathbf{q}}_{i-1,\delta} & : \ \mathbf{b} \in \{0,1\}^3, \ \mathbf{q} \in \langle K \rangle^3, \ i \in [3m], \ \delta \in [0,2] \right\} \end{array} \right) .$$

For every $\mathbf{b} = (b_1, b_2, b_3) \in \{0,1\}^3$ and $\mathbf{q} = (q_1, q_2, q_3) \in \langle K \rangle^3$, denote:

$$C^{\delta}_i = C^{\mathbf{q}}_{i,\delta}, \quad D = D^{\mathbf{b},\mathbf{q}}, \quad E_{j,b} = E^{\mathbf{b},\mathbf{q}}_{j,b}, \quad G_I = G^{\mathbf{b},\mathbf{q}}_I, \quad H_{i-1,\delta} = H^{\mathbf{b},\mathbf{q}}_{i-1,\delta} \ ,$$

and perform the following zero-tests.

Validity of G: For every pair of non-empty disjoint subsets $I, I' \subseteq [3]$ test that:

$$\left[G_I \cdot G_{I'} = G_{I \cup I'} \right]_{\mathsf{pp}} .$$

Consistency of G and F: For every $i \in [3]$ test that:

$$\left[\mathsf{Restrict}(G_{\{i\}}, i) = F^{q_i} - b_i \right]_{\mathsf{pp}} .$$

Consistency of G and H_0: Test that:

$$\left[\sum_{\delta \in [0,2]} H_{0,\delta} \cdot C^{\delta}_1 = \left(D + \sum_{j \in [n]} E_{j,x_j} \right) \cdot G_{[3]} \right]_{\mathsf{pp}} .$$

Consistency of H_{i-1} and H_i: For every $i \in [3m-1]$ test that:

$$\left[\sum_{v \in \{0,1\}} \mathsf{ID}(C^1_i, v) \cdot \sum_{\delta \in [0,2]} H_{i-1,\delta} \cdot v^{\delta} = \sum_{\delta \in [0,2]} H_{i,\delta} \cdot C^{\delta}_{i+1} \right]_{\mathsf{pp}} .$$

Validity of H_{3m-1}: Test that:

$$\left[\sum_{v \in \{0,1\}} \mathsf{ID}(C^1_{3m}, v) \cdot \sum_{\delta \in [0,2]} H_{3m-1,\delta} \cdot v^{\delta} = 0 \right]_{\mathsf{pp}} .$$

5.2 Completeness

Fix $\kappa \in \mathbb{N}$, $K, n \leq M \leq 2^\kappa$, an input $x \in \{0,1\}^n$, an M-variate 3CNF formula φ and an assignment $\sigma : \{0,1\}^m \to \{0,1\}$ satisfying $\varphi \wedge I_x$. Consider the experiment:

$$\begin{aligned}&(\mathsf{pk}, \mathsf{vk}) \leftarrow \mathsf{QA.S}(\kappa, \varphi, n, K)\\ &\Pi \leftarrow \mathsf{QA.P}(\mathsf{pk}, \sigma)\\ &\mathsf{QA.V}(\mathsf{vk}, x, \Pi)\end{aligned}$$

We need to prove that in this experiment, all of the verifier's tests pass and it accepts with probability 1. Fix $\mathbf{b} = (b_1, b_2, b_3) \in \{0,1\}^3$ and $\mathbf{q} = (q_1, q_2, q_3) \in \langle K \rangle^3$ and denote:

$$B = B^{\mathbf{q}}, \quad C_i^\delta = C_{i,\delta}^{\mathbf{q}}, \quad C_i^\delta = C_{i,\delta}^{\mathbf{q}}, \quad D = D^{\mathbf{b},\mathbf{q}}, \quad E_{j,b} = E_{j,b}^{\mathbf{b},\mathbf{q}}, \quad G_I = G_I^{\mathbf{b},\mathbf{q}}, \quad H_{i-1,\delta} = H_{i-1,\delta}^{\mathbf{b},\mathbf{q}} \ .$$

By construction, the following ciphertexts are computed by evaluating the following polynomials on the ciphertext B:

$$C_i^\delta = \left[Z_i^\delta(B)\right]_{\mathsf{pp}} \ , \tag{6}$$

$$D = \left[\tilde{\varphi}(B, \mathbf{b})\right]_{\mathsf{pp}} \ , \tag{7}$$

$$E_{j,b} = \left[\tilde{I}_{j,b}(B, \mathbf{b})\right]_{\mathsf{pp}} \ , \tag{8}$$

$$G_I = \left[\Sigma_I^{\mathbf{b}}(B)\right]_{\mathsf{pp}} \ , \tag{9}$$

$$H_{i-1,\delta} = \left[P_{i-1}^{\mathbf{b}}\big|_{i,\delta}(B)\right]_{\mathsf{pp}} \ . \tag{10}$$

We argue that all the verfier's tests for $\mathbf{b}$ and $\mathbf{q}$ pass.

Validity of G. For every pair of non-empty disjoint subsets $I, I' \subseteq [3]$ the verifier tests that:

$$\left[G_I \cdot G_{I'} = G_{I \cup I'}\right]_{\mathsf{pp}} \ .$$

By the definition of the polynomial $\Sigma_I^{\mathbf{b}}$ we have:

$$\begin{aligned}\Sigma_I^{\mathbf{b}}(\mathbf{z}_1, \mathbf{z}_2, \mathbf{z}_3) \cdot \Sigma_{I'}^{\mathbf{b}}(\mathbf{z}_1, \mathbf{z}_2, \mathbf{z}_3) &\equiv \prod_{i \in I} (\Sigma(\mathbf{z}_i) - b_i) \cdot \prod_{i \in I'} (\Sigma(\mathbf{z}_i) - b_i)\\ &\equiv \prod_{i \in I \cup I'} (\Sigma(\mathbf{z}_i) - b_i)\\ &\equiv \Sigma_{I \cup I'}^{\mathbf{b}}(\mathbf{z}_1, \mathbf{z}_2, \mathbf{z}_3) \ .\end{aligned}$$

Therefore, by (9) and by the weak completeness of the zero test (Definition 4.4) the test passes.

Consistency of G and F. For every $i \in [3]$ the verifier tests that:

$$\left[\mathsf{Restrict}(G_{\{i\}}, i) = F^{q_i} - b_i\right]_{\mathsf{pp}} \ .$$

By construction:

$$A^{q_i} = \mathsf{Enc}(\gamma_{q_i}) \quad , \quad B^{\mathbf{q}} = \mathsf{MEnc}(\gamma_{q_1}, \gamma_{q_2}, \gamma_{q_3}) \ .$$

By the definition of the polynomial $\Sigma_i^{\mathbf{b}}$, for every $z_1, z_2, z_3 \in \mathbb{Z}^m$:

$$\Sigma_i^{\mathbf{b}}(z_1, z_2, z_3) = \Sigma(\mathbf{z}_i) - b_i \ .$$

Therefore, by (9) and by the correctness of the restriction operation (Definition 4.10):

$$\mathsf{Restrict}(G_{\{i\}}, i) = \left[\Sigma(A^{q_i}) - b_i\right]_{\mathsf{pp}} \ .$$

We, therefore, need to show that the following test passes:

$$[\Sigma(A^{q_i}) - b_i = F^{q_i} - b_i]_{\mathsf{pp}} \ .$$

By construction $F^q = [\Sigma(A^q)]_{\mathsf{pp}}$. Therefore, by the weak completeness of the zero test (Definition 4.4) the test passes.

Consistency of G and H_0. The verifier tests that:

$$\left[\sum_{\delta\in[0,2]} H_{0,\delta}\cdot C_1^\delta = \left(D + \sum_{j\in[n]} E_{j,x_j}\right)\cdot G_{[3]}\right]_{\mathsf{pp}} \ .$$

By the definition of the polynomial $P_0^{\mathbf{b}}$ we have:

$$\begin{aligned}\sum_{\delta\in[0,2]} P_0^{\mathbf{b}}\big|_{1,\delta}(\mathbf{z})\cdot Z_i^\delta(\mathbf{z}) &\equiv P_0^{\mathbf{b}}(\mathbf{z})\\ &\equiv (\tilde{\varphi} + \tilde{I}_x)(\mathbf{z},\mathbf{b})\cdot\Sigma_{[3]}^{\mathbf{b}}(\mathbf{z})\\ &\equiv (\tilde{\varphi} + \sum_{j\in[n]}\tilde{I}_{j,x_j})(\mathbf{z},\mathbf{b})\cdot\Sigma_{[3]}^{\mathbf{b}}(\mathbf{z})\end{aligned}$$

Therefore, by (6) through (10) and by the weak completeness of the zero test (Definition 4.4) the test passes.

Consistency of H_{i-1} and H_i. For every $i\in[3m-1]$ the verifier tests that:

$$\left[\sum_{v\in\{0,1\}} \mathsf{ID}(C_i^1, v)\cdot\sum_{\delta\in[0,2]} H_{i-1,\delta}\cdot v^\delta = \sum_{\delta\in[0,2]} H_{i,\delta}\cdot C_{i+1}^\delta\right]_{\mathsf{pp}} \ .$$

By the definition of the polynomial $P_i^{\mathbf{b}}$ for $\mathbf{z} = (z_1,\ldots,z_{3m})$ we have:

$$\begin{aligned}&\sum_{v\in\{0,1\}} \mathsf{ID}(Z_i^1(\mathbf{z}), v)\cdot\sum_{\delta\in[0,2]} P_{i-1}^{\mathbf{b}}\big|_{i,\delta}(\mathbf{z})\cdot v^\delta\\ \equiv &\sum_{v\in\{0,1\}} \mathsf{ID}(z_i, v)\cdot P_{i-1}^{\mathbf{b}}(z_1,\ldots,z_{i-1},v,z_{i+1},\ldots,z_{3m})\\ \equiv &\sum_{v_1,\ldots,v_i\in\{0,1\}} \mathsf{ID}(z_1,\ldots,z_i,v_1,\ldots,v_i)\cdot P_0^{\mathbf{b}}(v_1,\ldots,v_i,z_{i+1},\ldots,z_{3m})\\ \equiv &\ P_i^{\mathbf{b}}(\mathbf{z})\\ \equiv &\sum_{\delta\in[0,2]} P_i^{\mathbf{b}}\big|_{i+1,\delta}(\mathbf{z})\cdot Z_{i+1}^\delta(\mathbf{z}) \ .\end{aligned}$$

Therefore, by (6) and (10) and by the weak completeness of the zero test (Definition 4.4) the test passes.

Validity of H_{3m-1}. The verifier tests that:

$$\left[\sum_{v\in\{0,1\}} \mathsf{ID}(C_{3m}^1, v)\cdot\sum_{\delta\in[0,2]} H_{3m-1,\delta}\cdot v^\delta = 0\right]_{\mathsf{pp}} \ .$$

By the definition of the polynomial $P_i^{\mathbf{b}}$ for $\mathbf{z} = (z_1, \ldots, z_{3m})$ we have:

$$\begin{aligned}
&\sum_{v\in\{0,1\}} \mathsf{ID}(Z_{3m}^1(\mathbf{z}), v) \cdot \sum_{\delta\in[0,2]} P_{3m-1}^{\mathbf{b}}\big|_{3m,\delta}(\mathbf{z}) \cdot v^\delta \\
\equiv &\sum_{v\in\{0,1\}} \mathsf{ID}(z_{3m}, v) \cdot P_{3m-1}^{\mathbf{b}}(z_1, \ldots, z_{3m-1}, v) \\
\equiv &\sum_{\mathbf{v}\in\{0,1\}^{3m}} \mathsf{ID}(\mathbf{z}, \mathbf{v}) \cdot P_0^{\mathbf{b}}(\mathbf{v}) \ .
\end{aligned}$$

By construction, since the assignment σ satisfies the formula $\varphi \wedge I_x$ we have that $P_0^{\mathbf{b}}(\mathbf{v}) = 0$ for every $\mathbf{v} \in \{0,1\}^{3m}$, and therefore:

$$\begin{aligned}
&\sum_{v\in\{0,1\}} \mathsf{ID}(Z_{3m}^1(\mathbf{z}), v) \cdot \sum_{\delta\in[0,2]} P_{3m-1}^{\mathbf{b}}\big|_{3m,\delta}(\mathbf{z}) \cdot v^\delta \\
\equiv &\sum_{\mathbf{v}\in\{0,1\}^{3m}} \mathsf{ID}(\mathbf{z}, \mathbf{v}) \cdot P_0^{\mathbf{b}}(\mathbf{v}) \equiv 0 \ .
\end{aligned}$$

Therefore, by (6) and (10) and by the weak completeness of the zero test (Definition 4.4) the test passes.

5.3 No-signaling Extraction

Let M, n, K be polynomials in the security parameter. We construct a PPT no-signaling extractor E, and show that it satisfies the requirements in Definition 5.3.

Fix an ensemble of M-variate formulas $\{\varphi_\kappa\}$ and a poly-size adversary Adv acting as a malicious prover in the quasi-argument. The extractor E is given oracle access to Adv and it is given as input the formula $\varphi = \varphi_\kappa$ and a set $\mathbf{S} \subseteq \{0,1\}^m$ of size at most K. We assume WLOG that $\mathbf{S}$ contains the indices of the variables z', z'' used in the input formula (see Section 5.1.1). E proceeds as follows:

- Fix K distinct indexes $\mathbf{u}_1, \ldots, \mathbf{u}_K \in \{0,1\}^m$ such that $\mathbf{S} \subseteq \{\mathbf{u}_1, \ldots, \mathbf{u}_K\}$.
- Emulate the algorithm $\mathsf{QA.S}(\kappa, \varphi, n, K)$ with the following modification: instead of setting $\gamma_q = (\mathsf{sk}_q, 0^m, r_q)$, set $\gamma_q = (\mathsf{sk}_q, \mathbf{u}_q, r_q)$. Obtain the prover and verifier keys pk, vk.
- Query Adv with the keys pk, vk and obtain inputs x, y and a proof Π including the ciphertexts $F^1, \ldots, F^K$.
- If $\mathsf{QA.V}(\mathsf{vk}, x, \Pi) = 0$, then set $x = y = \perp$.
- Fix an assignment $\sigma : \mathbf{S} \to \{0,1\}$ as follows: For every $\mathbf{u}_q \in \mathbf{S}$ set $b \leftarrow \mathsf{Dec}(\mathsf{sk}_q, F^q)$. If $b \in \{0,1\}$ set $\sigma(\mathbf{u}_q) = b$. Otherwise, if $b = \perp$ set $\sigma(\mathbf{u}_q)$ to an arbitrary bit in $\{0,1\}$.
- Output x, y and σ.

Next we argue that E satisfies the requirements in Definition 5.3. To argue that E samples the correct distribution we need to show that for every PPT distinguisher D and $\kappa \in \mathbb{N}$:

$$\left| \Pr\left[\mathsf{D}(x,y) = 1 \;\middle|\; \begin{array}{l} (\mathsf{pk}, \mathsf{vk}) \leftarrow \mathsf{QA.S}(\kappa, \varphi, n, K) \\ (x, y, \Pi) \leftarrow \mathsf{Adv}(\mathsf{pk}, \mathsf{vk}) \\ \text{if } \mathsf{QA.V}(\mathsf{vk}, x, \Pi) = 0 : \\ \quad \text{set } x = y = \perp \end{array} \right] - \Pr_{(x,y,\sigma)\leftarrow \mathsf{E}^{\mathsf{Adv}}(\varphi,\emptyset)} [\mathsf{D}(x,y) = 1] \right| \leq \mathrm{negl}(\kappa) \ .$$

This follows from the construction of E and the semantic security of the encryption (Definition 4.3) by a standard hybrid argument. The proof is omitted. We proceed to show that E satisfies local consistency and no-signaling.

5.3.1 Local consistency.

Fix $\kappa \in \mathbb{N}$ and a set $\mathbf{S} \subseteq \{0,1\}^m$ of size at most K containing the indices of the variables z', z''. We show that:

$$\Pr_{(x,y,\sigma)\leftarrow \mathsf{E}(\varphi,\mathbf{S})}\left[x = \perp \quad \vee \quad \begin{array}{l} \forall i \in \mathbf{S} \cap [n] : \ \sigma(i) = x_i \\ \varphi(\sigma) = 1 \end{array}\right] = 1 \ .$$

By the construction of the input formula I_x and since $\mathbf{S}$ contains the indices of the variables z', z'', it is sufficient to prove:

$$\Pr_{(x,y,\sigma)\leftarrow \mathsf{E}(\varphi,\mathbf{S})}\left[\begin{array}{l} x \neq \perp \\ (\varphi \wedge I_x)(\sigma) = 0 \end{array}\right] = 0 \ .$$

Since E outputs $x = \perp$ whenever $\mathsf{QA.V}(\mathsf{vk}, x, \Pi)$ rejects, it is sufficient to show that:

$$\Pr_{(x,y,\sigma)\leftarrow \mathsf{E}^{\mathsf{Adv}}(\varphi,\mathbf{S})}\left[\begin{array}{l} x \neq \perp \\ 1 = \mathsf{QA.V}(\mathsf{vk}, x, \Pi) \\ (\varphi \wedge I_x)(\sigma) = 0 \end{array}\right] = 0 \ . \tag{11}$$

The proof of local consistency proceeds as follows. We first define a computationally unbounded verifier $\tilde{\mathsf{V}}$. Loosely speaking, $\tilde{\mathsf{V}}$ emulates the verifier $\mathsf{QA.V}$ "in the clear". That is, $\tilde{\mathsf{V}}$ first uses the inefficient algorithm Val to decrypt the ciphertexts in the verifier key vk and in the proof Π and then performs the same tests as $\mathsf{QA.V}$ except on the plaintexts instead of the ciphertexts. To prove (11) we argue that with overwhelming probability:

- Whenever $\mathsf{QA.V}$ accepts $\tilde{\mathsf{V}}$ also accepts. This follows from the soundness property of the zero-test operation (Definition 4.5).
- Whenever $\tilde{\mathsf{V}}$ accepts σ locally satisfies $\varphi \wedge I_x$.

The computationally unbounded verifier $\tilde{\mathsf{V}}$ is given as input:

- A verifier key:

$$\mathsf{vk} = \left(\begin{array}{ll} \mathsf{pp} & \\ \left\{C^{\mathbf{q}}_{i,\delta}\right. & \left. : \ \mathbf{q} \in \langle K\rangle^3, \ i \in [3m], \ \delta \in [0,2]\right\} \\ \left\{D^{\mathbf{b},\mathbf{q}}\right. & \left. : \ \mathbf{b} \in \{0,1\}^3, \ \mathbf{q} \in \langle K\rangle^3\right\} \\ \left\{E^{\mathbf{b},\mathbf{q}}_{j,b}\right. & \left. : \ \mathbf{b} \in \{0,1\}^3, \ \mathbf{q} \in \langle K\rangle^3, \ j \in [n], \ b \in \{0,1\}\right\} \end{array}\right) \ .$$

- An input string $x \in \{0,1\}^n$.
- A proof:

$$\Pi = \left(\begin{array}{ll} \{F^q & : \ q \in [K]\} \\ \left\{G^{\mathbf{b},\mathbf{q}}_I\right. & \left. : \ \mathbf{b} \in \{0,1\}^3, \ \mathbf{q} \in \langle K\rangle^3, \ I \subseteq [3]\right\} \\ \left\{H^{\mathbf{b},\mathbf{q}}_{i-1,\delta}\right. & \left. : \ \mathbf{b} \in \{0,1\}^3, \ \mathbf{q} \in \langle K\rangle^3, \ i \in [3m], \ \delta \in [0,2]\right\} \end{array}\right) \ .$$

- Secret keys $\mathsf{sk}_1, \ldots, \mathsf{sk}_K$ (not given to the real verifier $\mathsf{QA.V}$).

For every $q \in [K]$ set $\tilde{F}^q \leftarrow \mathsf{Val}(\mathsf{sk}_q, F^q)$. For every $\mathbf{b} = (b_1, b_2, b_3) \in \{0,1\}^3$ and $\mathbf{q} = (q_1, q_2, q_3) \in \langle K\rangle^3$, let $\mathbf{sk} = (\mathsf{sk}_{q_1}, \mathsf{sk}_{q_2}, \mathsf{sk}_{q_3})$, and set:

$$\begin{aligned} \tilde{C}^{\delta}_i &\leftarrow \mathsf{Val}(\mathbf{sk}, C^{\mathbf{q}}_{i,\delta}) \ , \\ \tilde{D} &\leftarrow \mathsf{Val}(\mathbf{sk}, D^{\mathbf{b},\mathbf{q}}) \ , \\ \tilde{E}_{j,b} &\leftarrow \mathsf{Val}(\mathbf{sk}, E^{\mathbf{b},\mathbf{q}}_{j,b}) \ , \\ \tilde{G}_I &\leftarrow \mathsf{Val}(\mathbf{sk}, G^{\mathbf{b},\mathbf{q}}_I) \ , \\ \tilde{H}_{i-1,\delta} &\leftarrow \mathsf{Val}(\mathbf{sk}, H^{\mathbf{b},\mathbf{q}}_{i-1,\delta}) \ . \end{aligned}$$

If any of these executions of Val output $\perp$, then $\tilde{\mathsf{V}}$ rejects. Otherwise, the values defined above are in the field $\mathbb{F}$ defined by pp and $\tilde{\mathsf{V}}$ performs the following tests on these field elements.

Validity of $\tilde{G}$: For every pair of non-empty disjoint subsets $I, I' \subseteq [3]$ test that:

$$\tilde{G}_I \cdot \tilde{G}_{I'} = \tilde{G}_{I \cup I'} \ .$$

Consistency of $\tilde{G}$ and $\tilde{F}$: For every $i \in [3]$ test that:

$$\tilde{G}_{\{i\}} = \tilde{F}^{q_i} - b_i \ .$$

Consistency of $\tilde{G}$ and $\tilde{H}_0$: Test that:

$$\sum_{\delta \in [0,2]} \tilde{H}_{0,\delta} \cdot \tilde{C}_1^{\delta} = \left(\tilde{D} + \sum_{j \in [n]} \tilde{E}_{j,x_j} \right) \cdot \tilde{G}_{[3]} \ .$$

Consistency of $\tilde{H}_{i-1}$ and $\tilde{H}_i$: For every $i \in [3m-1]$ test that:

$$\sum_{v \in \{0,1\}} \mathsf{ID}(\tilde{C}_i^1, v) \cdot \sum_{\delta \in [0,2]} \tilde{H}_{i-1,\delta} \cdot v^{\delta} = \sum_{\delta \in [0,2]} \tilde{H}_{i,\delta} \cdot \tilde{C}_{i+1}^{\delta} \ .$$

Validity of $\tilde{H}_{3m-1}$: Test that:

$$\sum_{v \in \{0,1\}} \mathsf{ID}(\tilde{C}_{3m}^1, v) \cdot \sum_{\delta \in [0,2]} \tilde{H}_{3m-1,\delta} \cdot v^{\delta} = 0 \ .$$

To show that (11) holds it is sufficient to prove the following two claims:

Claim 5.7.

$$\Pr_{(x,y,\sigma) \leftarrow \mathsf{E}^{\mathsf{Adv}}(\varphi, \mathbf{S})} \left[\begin{array}{l} 1 = \mathsf{QA.V}(\mathsf{vk}, x, \Pi) \\ 0 = \tilde{\mathsf{V}}(\mathsf{vk}, x, \Pi, (\mathsf{sk}_1, \ldots, \mathsf{sk}_K)) \end{array} \right] = 0 \ .$$

Claim 5.8.

$$\Pr_{(x,y,\sigma) \leftarrow \mathsf{E}^{\mathsf{Adv}}(\varphi, \mathbf{S})} \left[\begin{array}{l} x \neq \perp \\ 1 = \tilde{\mathsf{V}}(\mathsf{vk}, x, \Pi, (\mathsf{sk}_1, \ldots, \mathsf{sk}_K)) \\ (\varphi \wedge I_x)(\sigma) = 0 \end{array} \right] = 0 \ .$$

The proof of Claim 5.7 follows directly from the construction of the verifiers $\mathsf{QA.V}$ and $\tilde{\mathsf{V}}$ by the soundness of the zero-test operation (Definition 4.5).

Proof of Claim 5.8. We prove that:

$$x \neq \perp \quad \wedge \quad 1 = \tilde{\mathsf{V}}(\mathsf{vk}, x, \Pi, (\mathsf{sk}_1, \ldots, \mathsf{sk}_K)) \quad \Rightarrow \quad (\varphi \wedge I_x)(\sigma) = 1 \ .$$

Assuming that $x \neq \perp$ and $1 = \tilde{\mathsf{V}}(\mathsf{vk}, x, \Pi, (\mathsf{sk}_1, \ldots, \mathsf{sk}_K))$ we need to show that every clause in $\varphi \wedge I_x$ that only contains variables in $\mathbf{S}$ is satisfied by σ. Recall that $\tilde{\varphi} + \tilde{I}_x$ is an arithmetization of the formula $\varphi \wedge I_x$. Let $\mathbf{u}_1, \ldots, \mathbf{u}_K \in \{0,1\}^m$ be the distinct indexes fixed by E such that $\mathbf{S} \subseteq \{\mathbf{u}_1, \ldots, \mathbf{u}_K\}$. Since every clause in $\varphi \wedge I_x$ contain three distinct variables in $\mathbf{S}$ it is sufficient to prove that for every $\mathbf{q} = (q_1, q_2, q_3) \in \langle K \rangle^3$ and bits $\mathbf{b} = (b_1, b_2, b_3) \in \{0,1\}^3$

$$(\tilde{\varphi} + \tilde{I}_x)(\mathbf{u}_{q_1}, \mathbf{u}_{q_2}, \mathbf{u}_{q_3}, \mathbf{b}) = 1 \quad \Rightarrow \quad (\sigma(\mathbf{u}_{q_1}) = b_1 \ \vee \ \sigma(\mathbf{u}_{q_2}) = b_2 \ \vee \ \sigma(\mathbf{u}_{q_3}) = b_3) \ .$$

Let $\mathbf{u} = (u_1, \ldots, u_{3m}) \in \{0,1\}^{3m}$ be the vector $\mathbf{u} = (\mathbf{u}_{q_1}, \mathbf{u}_{q_2}, \mathbf{u}_{q_3})$. By the constructions of E and $\tilde{\mathsf{V}}$, and by the correctness of evaluation property (Definition 4.1):

$$\tilde{C}_i^{\delta} = u_i^{\delta} \ , \tag{12}$$

$$\tilde{D} = \tilde{\varphi}(\mathbf{u}, \mathbf{b}) \ , \tag{13}$$

$$\tilde{E}_{j,b} = \tilde{I}_{j,b}(\mathbf{u}, \mathbf{b}) \ . \tag{14}$$

Recall that for $x \neq \perp$, $\tilde{I}_x = \sum_{j \in [n]} \tilde{I}_{j,x_j}$. And, therefore, by (13) and (14):

$$(\tilde{\varphi} + \tilde{I}_x)(\mathbf{u}, \mathbf{b}) = \tilde{D} + \sum_{j \in [n]} \tilde{E}_{j,x_j} \ . \tag{15}$$

Whenever $\tilde{\mathsf{V}}$ accepts, the validity test of $\tilde{G}$ passes and, therefore:

$$\tilde{G}_{[3]} = \tilde{G}_{\{1\}} \cdot \tilde{G}_{\{2\}} \cdot \tilde{G}_{\{3\}} \ . \tag{16}$$

Whenever $\tilde{\mathsf{V}}$ accepts, the consistency test of $\tilde{G}$ and $\tilde{F}$ passes and, therefore:

$$\tilde{G}_{\{i\}} = \tilde{F}^{q_i} - b_i \ . \tag{17}$$

Combining (16) and (17) we have:

$$\tilde{G}_{[3]} = \prod_{i \in [3]} \left(\tilde{F}^{q_i} - b_i \right) \ . \tag{18}$$

Whenever $\tilde{\mathsf{V}}$ accepts, the consistency test of $\tilde{G}$ and $\tilde{H}_0$ passes and, therefore:

$$\sum_{\delta \in [0,2]} \tilde{H}_{0,\delta} \cdot \tilde{C}_1^{\delta} = \left(\tilde{D} + \sum_{j \in [n]} \tilde{E}_{j,x_j} \right) \cdot \tilde{G}_{[3]} \ .$$

Combining the above with (12),(15) and (18) gives:

$$\sum_{\delta \in [0,2]} \tilde{H}_{0,\delta} \cdot u_1^{\delta} = (\tilde{\varphi} + \tilde{I}_x)(\mathbf{u}, \mathbf{b}) \cdot \prod_{i \in [3]} \left(\tilde{F}^{q_i} - b_i \right) \ . \tag{19}$$

Whenever $\tilde{\mathsf{V}}$ accepts, for every $i \in [3m-1]$ the consistency test of $\tilde{H}_{i-1}$ and $\tilde{H}_i$ passes and, therefore:

$$\sum_{v \in \{0,1\}} \mathsf{ID}(\tilde{C}_i^1, v) \cdot \sum_{\delta \in [0,2]} \tilde{H}_{i-1,\delta} \cdot v^{\delta} = \sum_{\delta \in [0,2]} \tilde{H}_{i,\delta} \cdot \tilde{C}_{i+1}^{\delta} \ .$$

Combining the above with (12), and by the fact that $u_i \in \{0,1\}$ we have:

$$\sum_{\delta \in [0,2]} \tilde{H}_{i-1,\delta} \cdot u_i^{\delta} = \sum_{v \in \{0,1\}} \mathsf{ID}(u_i, v) \cdot \sum_{\delta \in [0,2]} \tilde{H}_{i-1,\delta} \cdot v^{\delta} = \sum_{\delta \in [0,2]} \tilde{H}_{i,\delta} \cdot u_{i+1}^{\delta} \ .$$

Since the above holds for every $i \in [3m-1]$ it follows that:

$$\sum_{\delta \in [0,2]} \tilde{H}_{0,\delta} \cdot u_1^{\delta} = \sum_{\delta \in [0,2]} \tilde{H}_{3m-1,\delta} \cdot u_{3m}^{\delta} \ . \tag{20}$$

Whenever $\tilde{\mathsf{V}}$ accepts, the validity test of $\tilde{H}_{3m-1}$ passes and, therefore:

$$\sum_{v \in \{0,1\}} \mathsf{ID}(\tilde{C}_{3m}^1, v) \cdot \sum_{\delta \in [0,2]} \tilde{H}_{3m-1,\delta} \cdot v^{\delta} = 0 \ .$$

Combining the above with (12), and by the fact that $u_{3m} \in \{0,1\}$ we have:

$$\sum_{\delta\in[0,2]} \tilde{H}_{3m-1,\delta} \cdot u_{3m}^{\delta} = \sum_{v\in\{0,1\}} \mathsf{ID}(u_{3m}, v) \cdot \sum_{\delta\in[0,2]} \tilde{H}_{3m-1,\delta} \cdot v^{\delta} = 0 \ . \tag{21}$$

Combining (19), (20) and (21) we have:

$$(\tilde{\varphi} + \tilde{I}_x)(\mathbf{u}, \mathbf{b}) \cdot \prod_{i\in[3]} \left(\tilde{F}^{q_i} - b_i\right) = 0 \ .$$

Therefore, if $(\tilde{\varphi} + \tilde{I}_x)(\mathbf{u}, \mathbf{b}) = 1$, then there must exist some $i \in [3]$ such that $\tilde{F}^{q_i} = b_i \in \{0,1\}$. By the construction of E and $\tilde{\mathsf{V}}$ and by the correctness of decryption property (Definition 4.2) it follows that:

$$\sigma(\mathbf{u}_{q_i}) = \mathsf{Dec}(\mathsf{sk}_{q_i}, F^{q_i}) = \mathsf{Val}(\mathsf{sk}_{q_i}, F^{q_i}) = \tilde{F}^{q_i} = b_i \ .$$

Therefore, as required, it holds that:

$$(\sigma(\mathbf{u}_{q_1}) = b_1 \ \vee \ \sigma(\mathbf{u}_{q_2}) = b_2 \ \vee \ \sigma(\mathbf{u}_{q_3}) = b_3) \ .$$

□

5.3.2 No-signaling.

Fix a PPT distinguisher D. Assume towards contradiction that for infinitely many values of $\kappa \in \mathbb{N}$ there exist sets $\mathbf{S}^0 \subseteq \mathbf{S}^1 \subseteq \{0,1\}^m$ of size at most K such that:

$$\left| \Pr_{(x,y,\sigma)\leftarrow \mathsf{E}^{\mathsf{Adv}}(\varphi,\mathbf{S}^1)} \left[\mathsf{D}(x,y,\sigma(\mathbf{S}^0)) = 1\right] - \Pr_{(x,y,\sigma)\leftarrow \mathsf{E}^{\mathsf{Adv}}(\varphi,\mathbf{S}^0)} \left[\mathsf{D}(x,y,\sigma) = 1\right] \right| \geq \frac{1}{\mathrm{poly}(\kappa)} \ .$$

For every $b \in \{0,1\}$ let $\mathbf{u}_1^b, \ldots, \mathbf{u}_K^b \in \{0,1\}^m$ be the distinct indexes fixed by $\mathsf{E}^{\mathsf{Adv}}(\varphi, \mathbf{S}^b)$. Recall that $\mathbf{S}^b \subseteq \left\{\mathbf{u}_1^b, \ldots, \mathbf{u}_q^b\right\}$. We assume that for every $q_0, q_1 \in [K]$:

$$\mathbf{u}_{q_0}^0 = \mathbf{u}_{q_1}^1 \Rightarrow q_0 = q_1 \ . \tag{22}$$

This is WLOG since the output distribution of E is invariant with respect to the order of the indexes $\mathbf{u}_1^b, \ldots, \mathbf{u}_K^b$.

For $q \in [0, K]$ let Exp_q be the experiment where we sample (x, y, σ) by executing $\mathsf{E}^{\mathsf{Adv}}$ with φ and with fixed $\mathbf{u}_1, \ldots, \mathbf{u}_K$ such that $\mathbf{u}_{q'} = \mathbf{u}_{q'}^0$ for all $q' \leq q$ and $\mathbf{u}_{q'} = \mathbf{u}_{q'}^1$ for all $q' > q$. Note that Exp_0 corresponds to an execution of $\mathsf{E}^{\mathsf{Adv}}(\varphi, \mathbf{S}^1)$, and Exp_K corresponds to an execution of $\mathsf{E}^{\mathsf{Adv}}(\varphi, \mathbf{S}^0)$. Therefore, there exists $q^* \in [K]$ such that:

$$\left| \Pr_{\mathsf{Exp}_{q^*-1}} \left[\mathsf{D}(x,y,\sigma(\mathbf{S}^0)) = 1\right] - \Pr_{\mathsf{Exp}_{q^*}} \left[\mathsf{D}(x,y,\sigma(\mathbf{S}^0)) = 1\right] \right| \geq \frac{1}{\mathrm{poly}(\kappa)} \ . \tag{23}$$

Claim 5.9. $\mathbf{u}_{q^*}^0 \notin \mathbf{S}^0$.

Proof. By (22), $\mathbf{u}_{q^*}^0 \in \mathbf{S}^0 \subseteq \mathbf{S}^1$ implies $\mathbf{u}_{q^*}^0 = \mathbf{u}_{q^*}^1$. In this case the views of the experiments Exp_{q^*-1} and Exp_{q^*} are identical. □

We construct an adversary D' that breaks the semantic security of the encryption (Definition 4.3). Specifically, we show that:

$$\Pr\left[b'=b \;\middle|\; \begin{array}{l} b \leftarrow \{0,1\} \\ \mathsf{pp} \leftarrow \mathsf{ParamGen}(\kappa, m, 2) \\ \mathsf{sk} \leftarrow \mathsf{KeyGen}(\mathsf{pp}) \\ r \leftarrow \{0,1\}^\kappa \\ c_0 \leftarrow \mathsf{Enc}(\mathsf{sk}, \mathbf{u}^0_{q^*}, r) \\ c_1 \leftarrow \mathsf{Enc}(\mathsf{sk}, \mathbf{u}^1_{q^*}, r) \\ b' \leftarrow \mathsf{D}'(\mathsf{pp}, c_b) \end{array}\right] \geq \frac{1}{2} + \frac{1}{\mathrm{poly}(\kappa)} \ .$$

The adversary D' is given pp and a ciphertext c and it proceeds as follows.

- For every $q < q^*$ sample $\mathsf{sk}_q \leftarrow \mathsf{KeyGen}(\mathsf{pp}), r_q \leftarrow \{0,1\}^\kappa$ and set $\gamma_q = (\mathsf{sk}_q, \mathbf{u}^0_q, r_q)$.
- For every $q > q^*$ sample $\mathsf{sk}_q \leftarrow \mathsf{KeyGen}(\mathsf{pp}), r_q \leftarrow \{0,1\}^\kappa$ and set $\gamma_q = (\mathsf{sk}_q, \mathbf{u}^1_q, r_q)$.
- Set $\gamma_{q^*} = c$.
- For every $q \in [K] \setminus \{q^*\}$ set $A^q \leftarrow \mathsf{Enc}(\gamma_q)$. Set $A^{q^*} = c$.
- For every $\mathbf{q} = (q_1, q_2, q_3) \in \langle K \rangle^3$ such that $q^* \notin \{q_1, q_2, q_3\}$ set $B^{\mathbf{q}} \leftarrow \mathsf{MEnc}(\gamma_{q_1}, \gamma_{q_2}, \gamma_{q_3})$.
- For every $\mathbf{q} = (q_1, q_2, q_3) \in \langle K \rangle^3$ such that $q^* \in \{q_1, q_2, q_3\}$ set $B^{\mathbf{q}} \leftarrow \mathsf{Extend}(\gamma_{q_1}, \gamma_{q_2}, \gamma_{q_3})$.
- Emulate the remainder of the setup algorithm $\mathsf{QA.S}$ and obtain the prover and verifier keys pk, vk.
- Query Adv with the keys pk, vk and obtain an input x, auxiliary input y and proof Π which includes the ciphertexts $F^1, \ldots, F^K$.
- If $\mathsf{QA.V}(\mathsf{vk}, x, \Pi) = 0$, then set $x = y = \bot$.
- Fix an assignment $\sigma(\mathbf{S}^0) : \mathbf{S}^0 \to \{0,1\}$ as follows: For every $\mathbf{u}_q \in \mathbf{S}^0$ set $b \leftarrow \mathsf{Dec}(\mathsf{sk}_q, F^q)$. Note that by Claim 5.9, if $\mathbf{u}_q \in \mathbf{S}^0$ then $q \neq q^*$ and therefore sk_q is indeed defined. If $b \in \{0,1\}$ set $\sigma(\mathbf{u}_q) = b$. Otherwise, if $b = \bot$ set $\sigma(\mathbf{u}_q)$ to be an arbitrary bit in $\{0,1\}$.
- Output $\mathsf{D}(x, y, \sigma(\mathbf{S}^0))$.

The following claim together with (23) imply that D' breaks semantic security.

Claim 5.10.

$$\Pr_{\mathsf{Exp}_{q^*-1}}\left[\mathsf{D}(x,y,\sigma(\mathbf{S}^0))=1\right] = \Pr\left[\mathsf{D}'(\mathsf{pp},c)=1 \;\middle|\; \begin{array}{l} \mathsf{pp} \leftarrow \mathsf{ParamGen}(\kappa, m, 2) \\ \mathsf{sk} \leftarrow \mathsf{KeyGen}(\mathsf{pp}) \\ r \leftarrow \{0,1\}^\kappa \\ c \leftarrow \mathsf{Enc}(\mathsf{sk}, \mathbf{u}^1_{q^*}, r) \end{array}\right]$$

$$\Pr_{\mathsf{Exp}_{q^*}}\left[\mathsf{D}(x,y,\sigma(\mathbf{S}^0))=1\right] = \Pr\left[\mathsf{D}'(\mathsf{pp},c)=1 \;\middle|\; \begin{array}{l} \mathsf{pp} \leftarrow \mathsf{ParamGen}(\kappa, m, 2) \\ \mathsf{sk} \leftarrow \mathsf{KeyGen}(\mathsf{pp}) \\ r \leftarrow \{0,1\}^\kappa \\ c \leftarrow \mathsf{Enc}(\mathsf{sk}, \mathbf{u}^0_{q^*}, r) \end{array}\right]$$

Proof. By the construction of D' and by the correctness of extension property (Definition 4.9) we have that when c is an encryption of $\mathbf{u}^1_{q^*}$, the distribution of $\mathsf{D}(x, y, \sigma(\mathbf{S}^0))$ sampled by D' is identical to that sampled in the experiment Exp_{q^*-1}. Similarly, when c is an encryption of $\mathbf{u}^0_{q^*}$, the distribution of $\mathsf{D}(x, y, \sigma(\mathbf{S}^0))$ sampled by D' is identical to that sampled in the experiment Exp_{q^*}. □

6 Bootstrapping Theorem

In this section we prove our bootstrapping theorem. See Section 2.2 for an overview.

Theorem 6.1 (Bootstrapping)**.** *If there exists a family of collision-resistant hash functions and a publicly verifiable non-interactive quasi-argument (Definition 5.3), then there exists a publicly verifiable non-interactive delegation scheme for any RAM machine (Definition 3.4) where the setup time T_{S} and proof length L_{Π} are both $T^{O(1/\log_2 \log_\kappa T)}$.*

The proof of the theorem is by induction. The base case is proven in Section 6.1 and the inductive step is proven in Section 6.2

6.1 The Base Case

In the base of the induction we construct a RAM delegation scheme with a long setup time.

Theorem 6.2. *If there exists a family of collision-resistant hash functions and a publicly verifiable non-interactive quasi-argument (Definition 5.3), then there exists a publicly verifiable non-interactive delegation scheme for any RAM machine with setup time $T_{\mathsf{S}} = \mathrm{poly}(T, \kappa)$ and proof length $L_{\Pi} = \mathrm{poly}(\kappa)$ (Definition 3.4).*

6.1.1 Hash tree.

Before proving Theorem 6.2 we define the notion of hash trees. A hash-tree scheme consists of polynomial time algorithms:

$$(\mathsf{HT.Gen}, \mathsf{HT.Hash}, \mathsf{HT.Read}, \mathsf{HT.Write}, \mathsf{HT.VerRead}, \mathsf{HT.VerWrite}) \ ,$$

with the following syntax:

- $\mathsf{HT.Gen}$ is a randomized algorithm that takes as input the security parameter 1^κ and outputs a hash key dk.
- $\mathsf{HT.Hash}$ is a deterministic algorithm that takes as input a key dk and memory string D. It outputs a hash tree tree and a root rt.
- $\mathsf{HT.Read}$ is a deterministic algorithm that takes as input a tree tree and a memory location ℓ. It outputs a bit b and a proof Π.
- $\mathsf{HT.Write}$ is a deterministic algorithm that takes as input a tree tree, a memory location ℓ and a bit b. It outputs a new tree tree', a new root rt' and a proof Π.
- $\mathsf{HT.VerRead}$ is a deterministic algorithm that takes as input a key dk, a root rt, a memory location ℓ, a bit b and a proof Π. It outputs an acceptance bit.
- $\mathsf{HT.VerWrite}$ is a deterministic algorithm that takes as input a key dk, a root rt, a memory location ℓ, a bit b, a new root rt' and a proof Π. It outputs an acceptance bit.

Definition 6.3 (Hash-Tree)**.** *A hash-tree scheme*

$$(\mathsf{HT.Gen}, \mathsf{HT.Hash}, \mathsf{HT.Read}, \mathsf{HT.Write}, \mathsf{HT.VerRead}, \mathsf{HT.VerWrite}) \ ,$$

satisfies the following properties.

Completeness of Read. *For every $\kappa \in \mathbb{N}$, $D \in \{0,1\}^L$ such that $L \leq 2^\kappa$, and $\ell \in [L]$:*

$$\Pr\left[\begin{array}{l|l} 1 = \mathsf{HT.VerRead}(\mathsf{dk}, \mathsf{rt}, \ell, b, \Pi) & \mathsf{dk} \leftarrow \mathsf{HT.Gen}(1^\kappa) \\ D[\ell] = b & (\mathsf{tree}, \mathsf{rt}) \leftarrow \mathsf{HT.Hash}(\mathsf{dk}, D) \\ & (b, \Pi) \leftarrow \mathsf{HT.Read}(\mathsf{tree}, \ell) \end{array}\right] = 1 \ .$$

Completeness of Write. *For every $\kappa \in \mathbb{N}$, $D \in \{0,1\}^L$ such that $L \leq 2^\kappa$, $\ell \in [L]$ and $b \in \{0,1\}$ let D' be the string D with its ℓ-th location set to b. We have that:*

$$\Pr\left[\begin{array}{l|l} 1 = \mathsf{HT.VerWrite}(\mathsf{dk},\mathsf{rt},\ell,b,\mathsf{rt}',\Pi) & \mathsf{dk} \leftarrow \mathsf{HT.Gen}(1^\kappa) \\ (\mathsf{tree}',\mathsf{rt}') = \mathsf{HT.Hash}(\mathsf{dk},D') & (\mathsf{tree},\mathsf{rt}) \leftarrow \mathsf{HT.Hash}(\mathsf{dk},D) \\ & (\mathsf{tree}',\mathsf{rt}',\Pi) \leftarrow \mathsf{HT.Write}(\mathsf{tree},\ell,b) \end{array}\right] = 1 \ .$$

Efficiency. *In the completness experiments above, the running time of* $\mathsf{HT.Hash}$ *is* $|D| \cdot \mathrm{poly}(\kappa)$. *The length of the root produced by* $\mathsf{HT.Hash}$ *and the length of the proofs produced by* $\mathsf{HT.Read}$ *and* $\mathsf{HT.Write}$ *are* $\mathrm{poly}(\kappa)$.

Soundness of Read. *For every polynomial size adversary* Adv *there exists a negligible function μ such that for every $\kappa \in \mathbb{N}$:*

$$\Pr\left[\begin{array}{l|l} b_1 \neq b_2 & \mathsf{dk} \leftarrow \mathsf{HT.Gen}(1^\kappa) \\ 1 = \mathsf{HT.VerRead}(\mathsf{dk},\mathsf{rt},\ell,b_1,\Pi_1) & (\mathsf{rt},\ell,b_1,\Pi_1,b_2,\Pi_2) \leftarrow \mathsf{Adv}(\mathsf{dk}) \\ 1 = \mathsf{HT.VerRead}(\mathsf{dk},\mathsf{rt},\ell,b_2,\Pi_2) & \end{array}\right] \leq \mu(\kappa) \ .$$

Soundness of Write. *For every polynomial size adversary* Adv *there exists a negligible function μ such that for every $\kappa \in \mathbb{N}$:*

$$\Pr\left[\begin{array}{l|l} \mathsf{rt}_1 \neq \mathsf{rt}_2 & \mathsf{dk} \leftarrow \mathsf{HT.Gen}(1^\kappa) \\ 1 = \mathsf{HT.VerWrite}(\mathsf{dk},\mathsf{rt},\ell,b,\mathsf{rt}_1,\Pi_1) & (\mathsf{rt},\ell,b,\mathsf{rt}_1,\Pi_1,\mathsf{rt}_2,\Pi_2) \leftarrow \mathsf{Adv}(\mathsf{dk}) \\ 1 = \mathsf{HT.VerWrite}(\mathsf{dk},\mathsf{rt},\ell,b,\mathsf{rt}_2,\Pi_2) & \end{array}\right] \leq \mu(\kappa) \ .$$

Theorem 6.4 ([Mer87])**.** *A hash-tree scheme can be constructed from any family of collision-resistant hash functions.*

6.1.2 Construction.

Let $(\mathsf{HT.Gen}, \mathsf{HT.Hash}, \mathsf{HT.Read}, \mathsf{HT.Write}, \mathsf{HT.VerRead}, \mathsf{HT.VerWrite})$ be a hash-tree scheme (Definition 6.3) and let $(\mathsf{QA.S}, \mathsf{QA.P}, \mathsf{QA.V})$ be a publicly verifiable non-interactive quasi-argument (Definition 5.3). We construct a delegation scheme $(\mathsf{RDel.S}, \mathsf{RDel.D}, \mathsf{RDel.P}, \mathsf{RDel.V})$ for any RAM machine with setup time $T_\mathsf{S} = \mathrm{poly}(T,\kappa)$ and proof length $L_\Pi = \mathrm{poly}(\kappa)$ (Definition 3.4).

The RAM machine. Fix a RAM machine $\mathcal{R}$. We assume for simplicity and WLOG that in every step $\mathcal{R}$ reads from one memory location and then writes to one memory location. Let $(\mathsf{StepR}, \mathsf{StepW})$ be the following deterministic polynomial-time algorithms:

- Given a local state st of $\mathcal{R}$, StepR outputs the memory location ℓ such that $\mathcal{R}$ in state st reads from location ℓ.
- Given a local state st and a bit b, StepW outputs a bit b', a memory location ℓ' and a state st', such that $\mathcal{R}$ in state st, after reading the bit b, writes the bit b' to location ℓ' and transitions to state st'.

The setup algorithm $\mathsf{RDel.S}$**.** Given as input κ and T, the setup algorithm first samples a hash key $\mathsf{dk} \leftarrow \mathsf{HT.Gen}(1^\kappa)$. Next, it emulates the quasi-argument setup algorithm $\mathsf{QA.S}(\kappa,\phi,n,K)$ with the formula ϕ, locality parameter K, and input length n, defined below. It obtains the keys $(\mathsf{QA.pk}, \mathsf{QA.vk})$ and outputs the keys $(\mathsf{pk} = (\mathsf{QA.pk}, \mathsf{dk}) \ , \ \mathsf{vk} = \mathsf{QA.vk} \ , \ \mathsf{dk})$.

Let φ be a 3CNF formula corresponding to one step of $\mathcal{R}$. That is, for every pair of digests $\mathsf{h} = (\mathsf{st}, \mathsf{rt}), \mathsf{h}' = (\mathsf{st}', \mathsf{rt}')$, bit b, and proofs Π, Π' there exists w such that $\varphi(\mathsf{h}, \mathsf{h}', b, \Pi, \Pi', w) = 1$ if and only if:

$$\begin{aligned} \ell &\leftarrow \mathsf{StepR}(\mathsf{st}) \\ (b', \ell', \mathsf{st}'') &\leftarrow \mathsf{StepW}(\mathsf{st}, b) \\ \mathsf{st}' &= \mathsf{st}'' \\ 1 &= \mathsf{HT.VerRead}(\mathsf{dk}, \mathsf{rt}, \ell, b, \Pi) \\ 1 &= \mathsf{HT.VerWrite}(\mathsf{dk}, \mathsf{rt}, \ell', b', \mathsf{rt}', \Pi') \end{aligned}$$

Moreover, such w can be efficiently computed given $(\mathsf{h}, \mathsf{h}', b, \Pi, \Pi')$. By the efficiency of the hash-tree scheme, there exists such a formula φ with $K = \mathrm{poly}(\kappa)$ variables.

Let ϕ be the following formula over $M = O(K \cdot T)$ variables:

$$\phi\left(\mathsf{h}_0, (b_i, \Pi_i, \Pi'_i, w_i, \mathsf{h}_i)_{i\in[T]}\right) = \bigwedge_{i\in[T]} \varphi(\mathsf{h}_{i-1}, \mathsf{h}_i, b_i, \Pi_i, \Pi'_i, w_i) \ .$$

Let the variables describing $\mathsf{h}_0, \mathsf{h}_T$ be the n input variables of ϕ.

The digest algorithm $\mathsf{RDel.D}$**.** Given as input dk and a configuration cf that consists of state st and memory D, the digest algorithm computs the hash tree $(\mathsf{tree}, \mathsf{rt}) = \mathsf{HT.Hash}(\mathsf{dk}, D)$. It outputs the pair $(\mathsf{st}, \mathsf{rt})$ as the digest.

The prover algorithm $\mathsf{RDel.P}$**.** Given as input $\mathsf{pk} = (\mathsf{QA.pk}, \mathsf{dk})$ and a pair of configurations $\mathsf{cf}, \mathsf{cf}'$ such that $(\kappa, \mathsf{cf}, \mathsf{cf}', T) \in \mathcal{U}_{\mathcal{R}}$ the prover emulates $\mathcal{R}$ and obtains a satisfying assignment for ϕ as follows: let $\mathsf{cf} = (\mathsf{st}_0, D_0)$, let $(\mathsf{tree}_0, \mathsf{rt}_0) = \mathsf{HT.Hash}(\mathsf{dk}, D_0)$ and let $\mathsf{h}_0 = (\mathsf{st}_0, \mathsf{rt}_0)$. For every $i \in [T]$ let:

$$\begin{aligned} \ell_i &\leftarrow \mathsf{StepR}(\mathsf{st}_{i-1}) \ , \\ (b_i, \Pi_i) &\leftarrow \mathsf{HT.Read}(\mathsf{tree}_{i-1}, \ell_i) \ , \\ (b'_i, \ell'_i, \mathsf{st}_i) &\leftarrow \mathsf{StepW}(\mathsf{st}_{i-1}, b_i) \ , \\ (\mathsf{tree}_i, \mathsf{rt}_i, \Pi'_i) &\leftarrow \mathsf{HT.Write}(\mathsf{tree}_{i-1}, \ell'_i, b'_i) \ , \\ \mathsf{h}_i &\leftarrow (\mathsf{st}_i, \mathsf{rt}_i) \ , \end{aligned}$$

and let w_i be such that $\varphi(\mathsf{h}_{i-1}, \mathsf{h}_i, b_i, \Pi_i, \Pi'_i, w_i) = 1$ (such w_i can be efficiently computed).

The prover now holds an accepting assignment σ for ϕ:

$$\phi(\mathsf{h}_0, (b_i, \Pi_i, \Pi'_i, w_i, \mathsf{h}_i)_{i\in[T]}) = \bigwedge_{i\in[T]} \varphi(\mathsf{h}_{i-1}, \mathsf{h}_i, b_i, \Pi_i, \Pi'_i, w_i) = 1 \ .$$

It emulates the quasi-argument prover and outputs the proof $\Pi = \mathsf{QA.P}(\mathsf{QA.pk}, \sigma)$.

The verifier algorithm $\mathsf{RDel.V}$**.** Given as input vk, a pair of digests h, h' and a proof Π, the verifier emulates the quasi-argument verifier $\mathsf{QA.V}(\mathsf{vk}, (\mathsf{h}, \mathsf{h}'), \Pi)$ and accepts if and only if $\mathsf{QA.V}$ accepts.

6.1.3 Analysis.

The completeness and efficiency requirements of the construction follow immediately from those of the quasi-argument and the hash-tree scheme. We focus on proving soundness.

Fix a PPT adversary Adv and a polynomial $T = T(\kappa)$. Assume towards contradiction that there exists a polynomial p such that for infinitely many $\kappa \in \mathbb{N}$:

$$\Pr\left[\begin{array}{l|l} \mathsf{RDel.V}(\mathsf{vk}, \mathsf{h}, \mathsf{h}', \Pi) = 1 & \\ (\kappa, \mathsf{cf}, \mathsf{cf}', T) \in \mathcal{U}_{\mathcal{R}} & (\mathsf{pk}, \mathsf{vk}, \mathsf{dk}) \leftarrow \mathsf{RDel.S}(\kappa, T) \\ \mathsf{h} = \mathsf{RDel.D}(\mathsf{dk}, \mathsf{cf}) & (\mathsf{cf}, \mathsf{cf}', \mathsf{h}, \mathsf{h}', \Pi) \leftarrow \mathsf{Adv}(\mathsf{pk}, \mathsf{vk}, \mathsf{dk}) \\ \mathsf{h}' \neq \mathsf{RDel.D}(\mathsf{dk}, \mathsf{cf}') & \end{array}\right] \geq \frac{1}{p(\kappa)} \ . \tag{24}$$

Fix κ such that (24) holds. We say that a hash key dk is bad if conditioned on $\mathsf{RDel.S}$ sampling dk, the probability in (24) is at least $1/2p(\kappa)$. Therefore, a $1/2p(\kappa)$ fraction of hash keys are bad. Fix a bad dk and let ϕ be the formula defined by dk. By the construction, $\mathsf{RDel.V}$ accepts if and only if $\mathsf{QA.V}$ accepts. Therefore:

$$\Pr\left[\begin{array}{l|l} \mathsf{QA.V}(\mathsf{QA.vk},(\mathsf{h},\mathsf{h}'),\Pi)=1 & (\mathsf{QA.pk},\mathsf{QA.vk}) \leftarrow \mathsf{QA.S}(\kappa,\phi,n,K) \\ (\kappa,\mathsf{cf},\mathsf{cf}',T)\in\mathcal{U}_{\mathcal{R}} & (\mathsf{cf},\mathsf{cf}',\mathsf{h},\mathsf{h}',\Pi)\leftarrow \mathsf{Adv}(\mathsf{pk},\mathsf{vk},\mathsf{dk}) \\ \mathsf{h}=\mathsf{RDel.D}(\mathsf{dk},\mathsf{cf}) & \\ \mathsf{h}'\neq\mathsf{RDel.D}(\mathsf{dk},\mathsf{cf}') & \end{array}\right] \geq \frac{1}{2p(\kappa)} \ . \tag{25}$$

Let $\mathsf{QA.Adv}$ be the following adversary for the quasi-argument: given $(\mathsf{QA.pk},\mathsf{QA.vk})$ it uses the hash key dk and obtains $(\mathsf{pk},\mathsf{vk})$. It emulates Adv and obtains an input $x=(\mathsf{h},\mathsf{h}')$ to ϕ, an auxiliary input $y=(\mathsf{cf},\mathsf{cf}')$ and a proof Π. Let E be the no-signaling extractor of the quasi-argument. By the correct-distribution property of E for every PPT distinguisher D:

$$\left|\begin{array}{l} \Pr\left[\mathsf{D}(x,y)=1 \;\middle|\; \begin{array}{l}(\mathsf{QA.pk},\mathsf{QA.vk})\leftarrow\mathsf{QA.S}(\kappa,\phi,n,K)\\ (x,y,\Pi)\leftarrow\mathsf{QA.Adv}(\mathsf{QA.pk},\mathsf{QA.vk})\\ \text{if } \mathsf{QA.V}(\mathsf{QA.vk},x,\Pi)=0: \\ \quad \text{set } x=y=\perp\end{array}\right] \\ \quad -\Pr\left[\mathsf{D}(x,y)=1 \;\middle|\; (x,y,\sigma)\leftarrow\mathsf{E}^{\mathsf{QA.Adv}}(\phi,\emptyset)\right] \end{array}\right| \leq \mathrm{negl}(\kappa) \ . \tag{26}$$

In the experiments above, if $x,y\neq\perp$, we parse x as (h,h') and y as $(\mathsf{cf},\mathsf{cf}')$. Given the configuration cf, we define for every $i\in[0,T]$ the configuration $\overline{\mathsf{cf}}_i$ that follows i steps after cf. That is, $\overline{\mathsf{cf}}_i$ is the unique configuration such that $(\kappa,\mathsf{cf},\overline{\mathsf{cf}}_i,i)\in\mathcal{U}_{\mathcal{R}}$. We also define the digest $\overline{\mathsf{h}}_i=\mathsf{RDel.D}(\mathsf{dk},\overline{\mathsf{cf}}_i)$. Note that $\overline{\mathsf{h}}_i$ can be efficiently computed given cf.

Let CHEAT be the event that $x,y\neq\perp$ and $\mathsf{h}=\overline{\mathsf{h}}_0$, but $\mathsf{h}'\neq\overline{\mathsf{h}}_T$. By (25) and (26) we have:

$$\Pr_{(x,y,\sigma)\leftarrow\mathsf{E}^{\mathsf{QA.Adv}}(\phi,\emptyset)}[\mathsf{CHEAT}] \geq \frac{1}{\mathrm{poly}(\kappa)} \ .$$

For $i\in[T]$ let $\mathbf{S}_i\subseteq[M]$ be the set of ϕ's variables describing $\mathsf{h}_{i-1},\mathsf{h}_i,b_i,\Pi_i,\Pi'_i,w_i$. Note that $|\mathbf{S}_i|=K$. Let Exp_i be the experiment where we sample (x,y,σ) from $\mathsf{E}^{\mathsf{QA.Adv}}(\phi,\mathbf{S}_i)$.

By the no-signaling property of E it follows that:

$$\Pr_{\mathsf{Exp}_i}[\mathsf{CHEAT}] \geq \frac{1}{\mathrm{poly}(\kappa)} \ . \tag{27}$$

Since CHEAT implies $x\neq\perp$, by the local-consistency property of E:

$$\Pr_{\mathsf{Exp}_i}\left[\mathsf{CHEAT}\Rightarrow \begin{array}{l}\forall j\in\mathbf{S}_i\cap[n]:\ \sigma(j)=x_j\\ \phi(\sigma)=1\end{array}\right] \geq 1-\mathrm{negl}(\kappa) \ .$$

Let $\mathsf{h}_{i-1},\mathsf{h}_i,b_i,\Pi_i,\Pi'_i,w_i$ be the values assigned by σ to the variables in $\mathbf{S}_i$. If σ locally satisfies ϕ, then we have that $\varphi(\mathsf{h}_{i-1},\mathsf{h}_i,b_i,\Pi_i,\Pi'_i,w_i)=1$. Therefore:

$$\Pr_{\mathsf{Exp}_i}\left[\begin{array}{l}\mathsf{CHEAT}\\ \varphi(\mathsf{h}_{i-1},\mathsf{h}_i,b_i,\Pi_i,\Pi'_i,w_i)=0\end{array}\right] \leq \mathrm{negl}(\kappa) \ . \tag{28}$$

Additionally, in Exp_1, if σ is consistent with the input $x=(\mathsf{h},\mathsf{h}')$ then $\mathsf{h}_0=\mathsf{h}$. Therefore:

$$\Pr_{\mathsf{Exp}_1}\left[\begin{array}{l}\mathsf{CHEAT}\\ \mathsf{h}_0\neq\overline{\mathsf{h}}_0\end{array}\right] \leq \mathrm{negl}(\kappa) \ . \tag{29}$$

Similarly, in Exp_T, if σ is consistent with the input $x=(\mathsf{h},\mathsf{h}')$ then $\mathsf{h}_T=\mathsf{h}'$. Therefore:

$$\Pr_{\mathsf{Exp}_T}\left[\begin{array}{l}\mathsf{CHEAT}\\ \mathsf{h}_T=\overline{\mathsf{h}}_T\end{array}\right] \leq \mathrm{negl}(\kappa) \ . \tag{30}$$

By the no-signaling property of E, for every $i \in [T-1]$:

$$\left| \Pr_{\mathsf{Exp}_i} \left[\begin{array}{l} \mathsf{CHEAT} \\ \mathsf{h}_i = \overline{\mathsf{h}}_i \end{array} \right] - \Pr_{\mathsf{Exp}_{i+1}} \left[\begin{array}{l} \mathsf{CHEAT} \\ \mathsf{h}_i = \overline{\mathsf{h}}_i \end{array} \right] \right| \leq \mathrm{negl}(\kappa) \ . \tag{31}$$

By combining (27) to (31) it follows that there exists $i \in [T]$ such that:

$$\Pr_{\mathsf{Exp}_i} \left[\begin{array}{l} \mathsf{CHEAT} \\ \mathsf{h}_{i-1} = \overline{\mathsf{h}}_{i-1} \\ \mathsf{h}_i \neq \overline{\mathsf{h}}_i \\ \varphi(\mathsf{h}_{i-1}, \mathsf{h}_i, b_i, \Pi_i, \Pi'_i, w_i) = 1 \end{array} \right] \geq \frac{1}{T \cdot \mathrm{poly}(\kappa)} = \frac{1}{\mathrm{poly}(\kappa)} \ . \tag{32}$$

Fix $i \in [T]$ such that (32) holds. For $\mathsf{h}_{i-1} = (\mathsf{st}_{i-1}, \mathsf{rt}_{i-1})$ let:

$$\begin{aligned} \ell_i &\leftarrow \mathsf{StepR}(\mathsf{st}_{i-1}) \\ (b'_i, \ell'_i, \mathsf{st}_i) &\leftarrow \mathsf{StepW}(\mathsf{st}_{i-1}, b_i) \end{aligned}$$

For $\overline{\mathsf{cf}}_{i-1} = (\overline{\mathsf{st}}_{i-1}, \overline{D}_{i-1})$ let:

$$\begin{aligned} (\overline{\mathsf{tree}}_{i-1}, \overline{\mathsf{rt}}_{i-1}) &\leftarrow \mathsf{HT.Hash}(\mathsf{dk}, \overline{D}_{i-1}) \\ \overline{\ell}_i &\leftarrow \mathsf{StepR}(\overline{\mathsf{st}}_{i-1}) \ , \\ (\overline{b}_i, \overline{\Pi}_i) &\leftarrow \mathsf{HT.Read}(\overline{\mathsf{tree}}_{i-1}, \overline{\ell}_i) \ , \\ (\overline{b}'_i, \overline{\ell}'_i, \overline{\mathsf{st}}_i) &\leftarrow \mathsf{StepW}(\overline{\mathsf{st}}_{i-1}, \overline{b}_i) \ , \\ (\overline{\mathsf{tree}}_i, \overline{\mathsf{rt}}_i, \overline{\Pi}'_i) &\leftarrow \mathsf{HT.Write}(\overline{\mathsf{tree}}_{i-1}, \overline{\ell}'_i, \overline{b}'_i) \ , \end{aligned}$$

Let CHEAT_1 be the event that:

$$\begin{aligned} & b_i \neq \overline{b}_i \\ & 1 = \mathsf{HT.VerRead}(\mathsf{dk}, \mathsf{rt}_{i-1}, \ell_i, b_i, \Pi_i) \\ & 1 = \mathsf{HT.VerRead}(\mathsf{dk}, \mathsf{rt}_{i-1}, \ell_i, \overline{b}_i, \overline{\Pi}_i) \end{aligned}$$

Let CHEAT_2 be the event that:

$$\begin{aligned} & \mathsf{rt}_i \neq \overline{\mathsf{rt}}_i \\ & 1 = \mathsf{HT.VerWrite}(\mathsf{dk}, \mathsf{rt}_{i-1}, \ell'_i, b'_i, \mathsf{rt}_i, \Pi'_i) \\ & 1 = \mathsf{HT.VerWrite}(\mathsf{dk}, \mathsf{rt}_{i-1}, \ell'_i, b'_i, \overline{\mathsf{rt}}_i, \overline{\Pi}'_i) \end{aligned}$$

Claim 6.5.

$$\left[\begin{array}{l} \mathsf{CHEAT} \\ \mathsf{h}_{i-1} = \overline{\mathsf{h}}_{i-1} \\ \mathsf{h}_i \neq \overline{\mathsf{h}}_i \\ \varphi(\mathsf{h}_{i-1}, \mathsf{h}_i, b_i, \Pi_i, \Pi'_i, w_i) = 1 \end{array} \right] \quad \Rightarrow \quad \mathsf{CHEAT}_1 \vee \mathsf{CHEAT}_2 \ .$$

Before proving this claim, we use it to conclude the proof. By (32) and Claim 6.5:

$$\Pr_{\mathsf{Exp}_i} \left[\ \mathsf{CHEAT}_1 \vee \mathsf{CHEAT}_2 \ \right] \geq \frac{1}{\mathrm{poly}(\kappa)} \ . \tag{33}$$

Next, we define the adversary $\mathsf{HT.Adv}$ against the hash-tree scheme. Given the hash key dk, $\mathsf{HT.Adv}$ emulates the experiment Exp_i and:

- If CHEAT_1 holds, $\mathsf{HT.Adv}$ outputs $(\mathsf{rt}_{i-1}, \ell_i, b_i, \Pi_i, \overline{b}_i, \overline{\Pi}_i)$.

- If CHEAT_2 holds, $\mathsf{HT.Adv}$ outputs $(\mathsf{rt}_{i-1}, \ell'_i, b'_i, \mathsf{rt}_i, \Pi'_i, \overline{\mathsf{rt}}_i, \overline{\Pi}'_i)$.

Since (33) holds for every bad hash key dk and since a $1/2p(\kappa)$ fraction of hash keys are bad, it follows that $\mathsf{HT.Adv}$ breaks the hash-tree soundness of either read or write.

Proof of Claim 6.5. Assume that the LHS of the implication holds. Recall that, by construction, $\overline{\mathsf{h}}_{i-1} = (\overline{\mathsf{st}}_{i-1}, \overline{\mathsf{rt}}_{i-1})$ and $\overline{\mathsf{h}}_i = (\overline{\mathsf{st}}_i, \overline{\mathsf{rt}}_i)$. Therefore, since $\mathsf{h}_{i-1} = \overline{\mathsf{h}}_{i-1}$, it holds that $(\mathsf{st}_{i-1}, \mathsf{rt}_{i-1}) = (\overline{\mathsf{st}}_{i-1}, \overline{\mathsf{rt}}_{i-1})$ and $\ell_i = \overline{\ell}_i$. Since $\varphi(\mathsf{h}_{i-1}, \mathsf{h}_i, b_i, \Pi_i, \Pi'_i, w_i) = 1$ we have:

$$\begin{aligned}(\mathsf{st}_i, \mathsf{rt}_i) &= \mathsf{h}_i \\ 1 &= \mathsf{HT.VerRead}(\mathsf{dk}, \mathsf{rt}_{i-1}, \ell_i, b_i, \Pi_i) \\ 1 &= \mathsf{HT.VerWrite}(\mathsf{dk}, \mathsf{rt}_{i-1}, \ell'_i, b'_i, \mathsf{rt}_i, \Pi'_i)\end{aligned}$$

By the completeness of the hash-tree scheme:

$$\begin{aligned}1 &= \mathsf{HT.VerRead}(\mathsf{dk}, \overline{\mathsf{rt}}_{i-1}, \overline{\ell}_i, \overline{b}_i, \overline{\Pi}_i) \\ 1 &= \mathsf{HT.VerWrite}(\mathsf{dk}, \overline{\mathsf{rt}}_{i-1}, \overline{\ell}'_i, \overline{b}'_i, \overline{\mathsf{rt}}'_i, \overline{\Pi}'_i)\end{aligned}$$

We consider two cases. If $b_i \neq \overline{b}_i$ then CHEAT_1 holds. Otherwise, if $b_i = \overline{b}_i$ it follows that $(\ell'_i, b'_i, \mathsf{st}_i) = (\overline{\ell}'_i, \overline{b}'_i, \overline{\mathsf{st}}_i)$. However, since $\mathsf{h}_i \neq \overline{\mathsf{h}}_i$ we have that $\mathsf{rt}_i \neq \overline{\mathsf{rt}}_i$ so CHEAT_2 holds.

□

6.2 The Inductive Step

In the inductive step we transform any RAM delegation scheme into a scheme with shorter setup time.

Theorem 6.6. *If there exists a publicly verifiable non-interactive quasi-argument (Definition 5.3) and a publicly verifiable non-interactive delegation scheme for a RAM machine $\mathcal{R}$ with setup time T_{S} and proof length L_Π (Definition 3.4), then for every polynomial $B = B(\kappa, T)$ there exists a publicly verifiable non-interactive delegation scheme for $\mathcal{R}$ with setup time T'_{S} and proof length L'_Π where:*

$$T'_{\mathsf{S}}(\kappa, T') = T_{\mathsf{S}}(\kappa, T) + \mathrm{poly}(\kappa, B, L_\Pi(\kappa, T)) \quad , \quad L'_\Pi(\kappa, T') = \mathrm{poly}(\kappa, L_\Pi(\kappa, T)) \quad , \quad T = T'/B \ .$$

6.2.1 Construction.

Let $(\mathsf{QA.S}, \mathsf{QA.P}, \mathsf{QA.V})$ be a publicly verifiable non-interactive quasi-argument (Definition 5.3) and let $(\mathsf{RDel.S}, \mathsf{RDel.D}, \mathsf{RDel.P}, \mathsf{RDel.V})$ be a publicly verifiable non-interactive delegation scheme for $\mathcal{R}$ with setup time T_{S} and proof length L_Π (Definition 3.4). We construct a new publicly verifiable non-interactive delegation scheme $(\mathsf{RDel.S}', \mathsf{RDel.D}', \mathsf{RDel.P}', \mathsf{RDel.V}')$ for $\mathcal{R}$ with setup time T'_{S} and proof length L'_Π.

The setup algorithm $\mathsf{RDel.S}'$. Given as input the parameters κ and T', the setup algorithm first emulates the original setup algorithm $\mathsf{RDel.S}(\kappa, T)$ with $T = T'/B$ and obtains the keys $(\mathsf{pk}, \mathsf{vk}, \mathsf{dk})$. Next, it emulates the quasi-argument setup algorithm $\mathsf{QA.S}(\kappa, \phi, n, K)$ with the formula ϕ defined below and obtains the keys $(\mathsf{QA.pk}, \mathsf{QA.vk})$. It outputs the keys $\big(\mathsf{pk}' = (\mathsf{pk}, \mathsf{QA.pk}, \mathsf{vk}) \ , \ \mathsf{vk}' = (\mathsf{vk}, \mathsf{QA.vk}) \ , \ \mathsf{dk}' = \mathsf{dk}\big)$.

Let φ be a 3CNF formula corresponding to the verification algorithm $\mathsf{RDel.V}$ with the key vk. That is, for every pair of digests h, h' and proof Π there exists w such that $\varphi(\mathsf{h}, \mathsf{h}', \Pi, w) = 1$ if and only if $\mathsf{RDel.V}(\mathsf{vk}, \mathsf{h}, \mathsf{h}', \Pi) = 1$. Moreover, such w can be efficiently computed given $(\mathsf{vk}, \mathsf{h}, \mathsf{h}', \Pi)$. Since $\mathsf{RDel.V}$ runs in time $O(L_\Pi(\kappa, T)) + \mathrm{poly}(\kappa)$, there exists such a formula φ with $K = \mathrm{poly}(L_\Pi(\kappa, T), \kappa)$ variables.

Let ϕ be the following formula over $M = O(K \cdot B)$ variables:

$$\phi\left(\mathsf{h}_0, (\Pi_i, w_i, \mathsf{h}_i)_{i \in [B]}\right) = \bigwedge_{i \in [B]} \varphi(\mathsf{h}_{i-1}, \mathsf{h}_i, \Pi_i, w_i) \ .$$

Let the variables describing $\mathsf{h}_0, \mathsf{h}_B$ be the n input variables of ϕ.

The digest algorithm $\mathsf{RDel.D'}$**.** The digest algorithm is identical to the digest algorithm $\mathsf{RDel.D}$.

The prover algorithm $\mathsf{RDel.P'}$**.** Given the prover key $\mathsf{pk'} = (\mathsf{pk}, \mathsf{QA.pk})$ and a pair of source and destination configurations $\mathsf{cf}, \mathsf{cf'}$ such that $(\kappa, \mathsf{cf}, \mathsf{cf'}, T') \in \mathcal{U}_{\mathcal{R}}$ the prover algorithm first emulates $\mathcal{R}$ starting from configuration $\mathsf{cf}_0 = \mathsf{cf}$ and obtains, for every $i \in [B]$, the configurations cf_i such that $(\kappa, \mathsf{cf}_{i-1}, \mathsf{cf}_i, T) \in \mathcal{U}_{\mathcal{R}}$ (note that $\mathsf{cf}_B = \mathsf{cf'}$) and a proof $\Pi_i = \mathsf{RDel.P}(\mathsf{pk}, \mathsf{cf}_{i-1}, \mathsf{cf}_i)$. It also computes $\mathsf{h}_i = \mathsf{RDel.D}(\mathsf{dk}, \mathsf{cf}_i)$ and w_i such that $\varphi(\mathsf{h}_{i-1}, \mathsf{h}_i, \Pi_i, w_i) = 1$. It, therefore, holds an accepting assignment σ for ϕ:

$$\phi\left(\mathsf{h}_0, (\Pi_i, w_i, \mathsf{h}_i)_{i\in[B]}\right) = \bigwedge_{i\in[B]} \varphi(\mathsf{h}_{i-1}, \mathsf{h}_i, \Pi_i, w_i) = 1 \ .$$

It emulates the quasi-argument prover and outputs the proof $\Pi = \mathsf{QA.P}(\mathsf{QA.pk}, \sigma)$.

The verifier algorithm $\mathsf{RDel.V'}$**.** given the verifier key $\mathsf{vk'} = (\mathsf{vk}, \mathsf{QA.vk})$, a pair of digests $\mathsf{h}, \mathsf{h'}$ and a proof Π the verifier algorithm emulates the original verifier algorithm $\mathsf{QA.V}(\mathsf{QA.vk}, (\mathsf{h}, \mathsf{h'}), \Pi)$ and accepts if and only if $\mathsf{QA.V}$ accepts.

6.2.2 Analysis.

The completeness and efficiency requirements of the construction follow immediately from those of the quasi-argument and the original RAM delegation scheme. We focus on proving soundness. Fix a PPT adversary Adv and a polynomial $T' = T'(\kappa)$. Assume towards contradiction that there exists a polynomial p such that for infinitely many $\kappa \in \mathbb{N}$:

$$\Pr\left[\begin{array}{l|l} \mathsf{RDel.V'}(\mathsf{vk}, \mathsf{h}, \mathsf{h'}, \Pi) = 1 & \\ (\kappa, \mathsf{cf}, \mathsf{cf'}, T') \in \mathcal{U}_{\mathcal{R}} & (\mathsf{pk'}, \mathsf{vk'}, \mathsf{dk}) \leftarrow \mathsf{RDel.S'}(\kappa, T') \\ \mathsf{h} = \mathsf{RDel.D}(\mathsf{dk}, \mathsf{cf}) & (\mathsf{cf}, \mathsf{cf'}, \mathsf{h}, \mathsf{h'}, \Pi) \leftarrow \mathsf{Adv}(\mathsf{pk'}, \mathsf{vk'}, \mathsf{dk}) \\ \mathsf{h'} \neq \mathsf{RDel.D}(\mathsf{dk}, \mathsf{cf'}) & \end{array}\right] \geq \frac{1}{p(\kappa)} \ . \tag{34}$$

Fix κ such that (34) holds. We say that a random tape r for $\mathsf{RDel.S}$ is bad if, when fixing r, the probability in (34) is at least $1/2p(\kappa)$. Therefore, a $1/2p(\kappa)$ fraction of r's are bad. Fix a bad r, let $(\mathsf{pk}, \mathsf{vk}, \mathsf{dk})$ be the keys defined by r and let ϕ be the formula defined by vk. By the construction, $\mathsf{RDel.V'}$ accepts if and only if $\mathsf{QA.V}$ accepts. Therefore:

$$\Pr\left[\begin{array}{l|l} \mathsf{QA.V}(\mathsf{QA.vk}, (\mathsf{h}, \mathsf{h'}), \Pi) = 1 & \\ (\kappa, \mathsf{cf}, \mathsf{cf'}, T') \in \mathcal{U}_{\mathcal{R}} & (\mathsf{QA.pk}, \mathsf{QA.vk}) \leftarrow \mathsf{QA.S}(\kappa, \phi, n, K) \\ \mathsf{h} = \mathsf{RDel.D}(\mathsf{dk}, \mathsf{cf}) & (\mathsf{cf}, \mathsf{cf'}, \mathsf{h}, \mathsf{h'}, \Pi) \leftarrow \mathsf{Adv}(\mathsf{pk'}, \mathsf{vk'}, \mathsf{dk}) \\ \mathsf{h'} \neq \mathsf{RDel.D}(\mathsf{dk}, \mathsf{cf'}) & \end{array}\right] \geq \frac{1}{2p(\kappa)} \ . \tag{35}$$

Let $\mathsf{QA.Adv}$ be the following adversary for the quasi-argument: given $(\mathsf{QA.pk}, \mathsf{QA.vk})$ it uses the keys $(\mathsf{pk}, \mathsf{vk}, \mathsf{dk})$ defined by r to obtain $(\mathsf{pk'}, \mathsf{vk'})$. It emulates Adv and obtains an input $x = (\mathsf{h}, \mathsf{h'})$ to ϕ, an auxiliary input $y = (\mathsf{cf}, \mathsf{cf'})$ and a proof Π. Let E be the no-signaling extractor of the quasi-argument. By the correct-distribution property of E for every PPT distinguisher D and $\kappa \in \mathbb{N}$:

$$\left|\begin{array}{c} \Pr\left[\mathsf{D}(x, y) = 1 \;\middle|\; \begin{array}{l} (\mathsf{QA.pk}, \mathsf{QA.vk}) \leftarrow \mathsf{QA.S}(\kappa, \phi, n, K) \\ (x, y, \Pi) \leftarrow \mathsf{QA.Adv}(\mathsf{QA.pk}, \mathsf{QA.vk}) \\ \text{if } \mathsf{QA.V}(\mathsf{QA.vk}, x, \Pi) = 0 : \\ \quad \text{set } x = y = \bot \end{array}\right] \\ -\Pr\left[\mathsf{D}(x, y) = 1 \;\middle|\; (x, y, \sigma) \leftarrow \mathsf{E}^{\mathsf{QA.Adv}}(\phi, \emptyset)\right] \end{array}\right| \leq \mathrm{negl}(\kappa) \ . \tag{36}$$

In the experiments above, if $x, y \neq \bot$, we parse x as $(\mathsf{h}, \mathsf{h'})$ and y as $(\mathsf{cf}, \mathsf{cf'})$. Given the configuration cf, we define for every $i \in [0, B]$ the configuration $\overline{\mathsf{cf}}_i$ that follows $T \cdot i$ steps after cf. That is, $\overline{\mathsf{cf}}_i$ is the unique

configuration such that $(\kappa, \mathsf{cf}, \overline{\mathsf{cf}}_i, T \cdot i) \in \mathcal{U}_{\mathcal{R}}$. We also define the digest $\overline{\mathsf{h}}_i = \mathsf{RDel.D}(\mathsf{dk}, \overline{\mathsf{cf}}_i)$. Note that $\overline{\mathsf{h}}_i$ can be efficiently computed given cf.

Let CHEAT be the event that $x, y \neq \bot$ and $\mathsf{h} = \overline{\mathsf{h}}_0$, but $\mathsf{h}' \neq \overline{\mathsf{h}}_B$. By (35) and (36) we have:

$$\Pr_{(x,y,\sigma) \leftarrow \mathsf{E}^{\mathsf{QA.Adv}}(\phi, \emptyset)} [\mathsf{CHEAT}] \geq \frac{1}{\mathrm{poly}(\kappa)} \ .$$

For $i \in [B]$ let $\mathbf{S}_i \subseteq [M]$ be the set of ϕ's variables describing $\mathsf{h}_{i-1}, \mathsf{h}_i, \Pi_i, w_i$. Note that $|\mathbf{S}_i| = K$. Let Exp_i be the experiment where we sample (x, y, σ) from $\mathsf{E}^{\mathsf{QA.Adv}}(\phi, \mathbf{S}_i)$.

By the no-signaling property of E it follows that:

$$\Pr_{\mathsf{Exp}_i} [\mathsf{CHEAT}] \geq \frac{1}{\mathrm{poly}(\kappa)} \ . \tag{37}$$

Since CHEAT implies $x \neq \bot$, by the local-consistency property of E it follows that:

$$\Pr_{\mathsf{Exp}_i} \left[\mathsf{CHEAT} \Rightarrow \begin{array}{l} \forall j \in \mathbf{S}_i \cap [n] : \ \sigma(j) = x_j \\ \phi(\sigma) = 1 \end{array} \right] \geq 1 - \mathrm{negl}(\kappa) \ .$$

Let $\mathsf{h}_{i-1}, \mathsf{h}_i, \Pi_i, w_i$ be the values assigned by σ to the variables in $\mathbf{S}_i$. If σ locally satisfies ϕ, then we have that $\varphi(\mathsf{h}_{i-1}, \mathsf{h}_i, \Pi_i, w_i) = 1$ and hence $\mathsf{RDel.V}(\mathsf{vk}, \mathsf{h}_{i-1}, \mathsf{h}_i, \Pi_i) = 1$:

$$\Pr_{\mathsf{Exp}_i} \left[\begin{array}{l} \mathsf{CHEAT} \\ \mathsf{RDel.V}(\mathsf{vk}, \mathsf{h}_{i-1}, \mathsf{h}_i, \Pi_i) = 0 \end{array} \right] \leq \mathrm{negl}(\kappa) \ . \tag{38}$$

Additionally, in Exp_1, if σ is consistent with the input $x = (\mathsf{h}, \mathsf{h}')$ then $\mathsf{h}_0 = \mathsf{h}$:

$$\Pr_{\mathsf{Exp}_1} \left[\begin{array}{l} \mathsf{CHEAT} \\ \mathsf{h}_0 \neq \overline{\mathsf{h}}_0 \end{array} \right] \leq \mathrm{negl}(\kappa) \ . \tag{39}$$

Similarly, in Exp_B, if σ is consistent with the input $x = (\mathsf{h}, \mathsf{h}')$ then $\mathsf{h}_B = \mathsf{h}'$:

$$\Pr_{\mathsf{Exp}_B} \left[\begin{array}{l} \mathsf{CHEAT} \\ \mathsf{h}_B = \overline{\mathsf{h}}_B \end{array} \right] \leq \mathrm{negl}(\kappa) \ . \tag{40}$$

By the no-signaling property of E, for every $i \in [B-1]$:

$$\left| \Pr_{\mathsf{Exp}_i} \left[\begin{array}{l} \mathsf{CHEAT} \\ \mathsf{h}_i = \overline{\mathsf{h}}_i \end{array} \right] - \Pr_{\mathsf{Exp}_{i+1}} \left[\begin{array}{l} \mathsf{CHEAT} \\ \mathsf{h}_i = \overline{\mathsf{h}}_i \end{array} \right] \right| \leq \mathrm{negl}(\kappa) \ . \tag{41}$$

By combining (37) to (41) it follows that there exists $i \in [B]$ such that:

$$\Pr_{\mathsf{Exp}_i} \left[\begin{array}{l} \mathsf{CHEAT} \\ \mathsf{h}_{i-1} = \overline{\mathsf{h}}_{i-1} \\ \mathsf{h}_i \neq \overline{\mathsf{h}}_i \\ \mathsf{RDel.V}(\mathsf{vk}, \mathsf{h}_{i-1}, \mathsf{h}_i, \Pi_i) = 1 \end{array} \right] \geq \frac{1}{B \cdot \mathrm{poly}(\kappa)} = \frac{1}{\mathrm{poly}(\kappa)} \ . \tag{42}$$

Next, we define the adversary $\mathsf{RDel.Adv}$ for the original RAM delegation scheme that given the keys $(\mathsf{pk}, \mathsf{vk}, \mathsf{dk})$ emulates the experiment Exp_i and outputs $(\overline{\mathsf{cf}}_{i-1}, \overline{\mathsf{cf}}_i, \mathsf{h}_{i-1}, \mathsf{h}_i, \Pi_i)$. Since (42) holds for every set of fixed keys sampled by $\mathsf{RDel.S}$ with a bad random tape and since a $1/2p(\kappa)$ fraction of random tapes are bad:

$$\Pr \left[\begin{array}{l} \mathsf{RDel.V}(\mathsf{vk}, \mathsf{h}_{i-1}, \mathsf{h}_i, \Pi_i) = 1 \\ (\kappa, \overline{\mathsf{cf}}_{i-1}, \overline{\mathsf{cf}}_i, T) \in \mathcal{U}_{\mathcal{R}} \\ \mathsf{h}_{i-1} = \mathsf{RDel.D}(\mathsf{dk}, \overline{\mathsf{cf}}_{i-1}) \\ \mathsf{h}_i \neq \mathsf{RDel.D}(\mathsf{dk}, \overline{\mathsf{cf}}_i) \end{array} \middle| \begin{array}{l} (\mathsf{pk}, \mathsf{vk}, \mathsf{dk}) \leftarrow \mathsf{RDel.S}(\kappa, T) \\ (\overline{\mathsf{cf}}_{i-1}, \overline{\mathsf{cf}}_i, \mathsf{h}_{i-1}, \mathsf{h}_i, \Pi_i) \leftarrow \mathsf{RDel.Adv}(\mathsf{pk}, \mathsf{vk}, \mathsf{dk}) \end{array} \right] \geq \frac{1}{\mathrm{poly}(\kappa)} \ .$$

This contradicts the soundness of the original RAM delegation scheme.

6.3 Proof of Theorem 6.1

In this section we put together the base case (Theorem 6.2) and the inductive step (Theorem 6.6) to prove our bootstrapping theorem (Theorem 6.1). Assuming a family of collision-resistant hash functions and a publicly verifiable non-interactive quasi-argument (Definition 5.3), we construct a publicly verifiable non-interactive delegation scheme RDel' for any RAM machine $\mathcal{R}$ where the setup time T_{S} and proof length L_{Π} are both $T^{O(1/\log_2 \log_\kappa T)}$.

We start with the delegation scheme RDel^0 for $\mathcal{R}$ given by Theorem 6.2 (the base case) with setup time $T^0_{\mathsf{S}} = \mathrm{poly}(T, \kappa)$ and proof length $L^0_{\Pi} = \mathrm{poly}(\kappa)$. For any $d \in \mathbb{N}$ let RDel^d be the delegation scheme obtained from RDel^0 after d applications of the transformation in Theorem 6.6 (the inductive step) with $B = T^{1/d}$. Then the setup time T^d_{S} and proof length L^d_{Π} of RDel^d are:

$$T^d_{\mathsf{S}}(\kappa, T) = d \cdot T^{O(1/d)} \cdot \kappa^{2^{O(d)}} \quad , \quad L^d_{\Pi}(\kappa, T) = \kappa^{2^{O(d)}} \ .$$

Our final scheme RDel' emulates the scheme RDel^d for $d = O(\log_2 \log_\kappa T)$ where κ, T are the parameters given to the setup algorithm. Note that since d is not a constant, the fact that RDel' is a delegation scheme does not follow directly from Theorems 6.2 and 6.6. Nevertheless, by following the proof of Theorem 6.6 it is straightforward to verify that RDel' indeed satisfies the completeness and efficiency requirements. As for soundness, for every polynomial $T = T(\kappa)$ the scheme RDel' is simply emulating RDel^d for some constant d and therefore soundness follows from Theorems 6.2 and 6.6.

References

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

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

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

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

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

[BCI+13] Nir Bitansky, Alessandro Chiesa, Yuval Ishai, Rafail Ostrovsky, and Omer Paneth. Succinct non-interactive arguments via linear interactive proofs. In *TCC*, pages 315–333, 2013.

[BEG+94] Manuel Blum, William S. Evans, Peter Gemmell, Sampath Kannan, and Moni Naor. Checking the correctness of memories. *Algorithmica*, 12(2/3):225–244, 1994.

[BFLS91] László Babai, Lance Fortnow, Leonid A. Levin, and Mario Szegedy. Checking computations in polylogarithmic time. In *Proceedings of the 23rd Annual ACM Symposium on Theory of Computing, May 5-8, 1991, New Orleans, Louisiana, USA*, pages 21–31, 1991.

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

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

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

[BK18] Zvika Brakerski and Yael Tauman Kalai. Monotone batch np-delegation with applications to access control. *IACR Cryptology ePrint Archive*, 2018:375, 2018.

[BKK+18] Saikrishna Badrinarayanan, Yael Tauman Kalai, Dakshita Khurana, Amit Sahai, and Daniel Wichs. Succinct delegation for low-space non-deterministic computation. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, Los Angeles, CA, USA, June 25-29, 2018*, pages 709–721, 2018.

[BMW99] Ingrid Biehl, Bernd Meyer, and Susanne Wetzel. Ensuring the integrity of agent-based computations by short proofs. In *Proceedings of the Second International Workshop on Mobile Agents*, MA '98, pages 183–194, London, UK, UK, 1999. Springer-Verlag.

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

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

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

[CCH+18] Ran Canetti, Yilei Chen, Justin Holmgren, Alex Lombardi, Guy N. Rothblum, and Ron D. Rothblum. Fiat-shamir from simpler assumptions. Cryptology ePrint Archive, Report 2018/1004, 2018. `https://eprint.iacr.org/2018/1004`.

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

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

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

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

[DFH12] Ivan Damgrard, Sebastian Faust, and Carmit Hazay. Secure two-party computation with low communication. In *Theory of Cryptography - 9th Theory of Cryptography Conference, TCC 2012, Taormina, Sicily, Italy, March 19-21, 2012. Proceedings*, pages 54–74, 2012.

[DFK+92] Cynthia Dwork, Uriel Feige, Joe Kilian, Moni Naor, and Shmuel Safra. Low communication 2-prover zero-knowledge proofs for NP. In *Advances in Cryptology - CRYPTO '92, 12th Annual International Cryptology Conference, Santa Barbara, California, USA, August 16-20, 1992, Proceedings*, pages 215–227, 1992.

[DLN+04] Cynthia Dwork, Michael Langberg, Moni Naor, Kobbi Nissim, and Omer Reingold. Succinct proofs for NP and spooky interactions. Unpublished manuscript, available at `http://www.cs.bgu.ac.il/~kobbi/papers/spooky_sub_crypto.pdf`, 2004.

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

[GGPR13] Rosario Gennaro, Craig Gentry, Bryan Parno, and Mariana Raykova. Quadratic span programs and succinct nizks without pcps. In *Advances in Cryptology - EUROCRYPT 2013, 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Athens, Greece, May 26-30, 2013. Proceedings*, pages 626–645, 2013.

[GH98] Oded Goldreich and Johan Hrastad. On the complexity of interactive proofs with bounded communication. *Inf. Process. Lett.*, 67(4):205–214, 1998.

[GKR08] Shafi Goldwasser, Yael Tauman Kalai, and Guy N. Rothblum. Delegating computation: interactive proofs for muggles. In Cynthia Dwork, editor, *STOC*, pages 113–122. ACM, 2008.

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

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

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

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

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

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

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

[KP15] Yael Tauman Kalai and Omer Paneth. Delegating RAM computations. *IACR Cryptology ePrint Archive*, 2015:957, 2015.

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

[KPY18] Yael Kalai, Omer Paneth, and Lisa Yang. On publicly verifiable delegation from standard assumptions. *IACR Cryptology ePrint Archive*, 2018:776, 2018.

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

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

[KT85] Leonid A. Khalfin and Boris S. Tsirelson. Quantum and quasi-classical analogs of Bell inequalities. In *In Symposium on the Foundations of Modern Physics*, pages 441–460, 1985.

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

[Mer87] Ralph C. Merkle. A digital signature based on a conventional encryption function. In *CRYPTO*, volume 293 of *Lecture Notes in Computer Science*, pages 369–378. Springer, 1987.

[Mic94] Silvio Micali. CS proofs (extended abstracts). In *35th Annual Symposium on Foundations of Computer Science, Santa Fe, New Mexico, USA, 20-22 November 1994*, pages 436–453. IEEE Computer Society, 1994. Full version in [Mic00].

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

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

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

[PRV12] Bryan Parno, Mariana Raykova, and Vinod Vaikuntanathan. How to delegate and verify in public: Verifiable computation from attribute-based encryption. In *Theory of Cryptography - 9th Theory of Cryptography Conference, TCC 2012, Taormina, Sicily, Italy, March 19-21, 2012. Proceedings*, pages 422–439, 2012.

[Ras85] Peter Rastall. Locality, Bell's theorem, and quantum mechanics. *Foundations of Physics*, 15(9):963–972, 1985.

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

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

[Val08] Paul Valiant. Incrementally verifiable computation or proofs of knowledge imply time/space efficiency. In *TCC*, pages 1–18, 2008.