Electronic Colloquium on Computational Complexity, Report No. 77 (2016)

Non-Interactive RAM and Batch NP Delegation from any PIR

Zvika Brakerski* Justin Holmgren† Yael Kalai‡

Abstract

We present an *adaptive* and *non-interactive* protocol for verifying arbitrary efficient computations in *fixed* polynomial time. Our protocol is computationally sound and can be based on any computational PIR scheme, which in turn can be based on standard polynomial-time cryptographic assumptions (e.g. the worst case hardness of polynomial-factor approximation of short-vector lattice problems). In our protocol, the prover and the verifier do not need to interact at all: The verifier sets up a public key ahead of time, and this key can be used by any prover to prove arbitrary statements in a completely adaptive manner. Verification is done using a secret verification key, and soundness relies on this key not being known to the prover. Our protocol further allows to prove statements about computations of arbitrary RAM machines.

Previous works either relied on knowledge assumptions, or could only offer non-adaptive two-message protocols (where the first message could not be re-used), and required either obfuscation-based assumptions or super-polynomial hardness assumptions.

We show that our techniques can also be applied to construct a new type of (non-adaptive) 2-message delegation protocols for *batch* NP-*statements*. Specifically, we can simultaneously prove the membership of multiple instances in a given NP language, with communication complexity proportional to the length of a *single* witness.

*Weizmann Institute of Science, zvika.brakerski@weizmann.ac.il. Supported by the Israel Science Foundation (Grant No. 468/14), the Alon Young Faculty Fellowship, Binational Science Foundation (Grant No. 712307) and Google Faculty Research Award.

†MIT, holmgren@mit.edu.

‡Microsoft Research, yael@microsoft.com.

ISSN 1433-8092

Contents

1 Introduction

Efficient verification of computation is one of the most fundamental tasks in computer science, and in particular it lies at the heart of the P vs. NP question. In the most basic setting, a prover P proves to a verifier V that a certain instance x is in some language L. In a *delegation protocol* the resources for verification should be much lower than those required to compute whether x is indeed in L. The most common flavor of this question, which will be the focus of this work, is to verify computations of time complexity T, with verification complexity $\mathrm{polylog}(T)$.[1] However, our results are not restricted to deterministic computations, and we also obtain non-deterministic (batch) delegation (in Section 6).

If soundness is sought against computationally unbounded provers, then a sound delegation protocol would have unlikely computational complexity consequences, in particular letting $L \in \mathrm{EXP}$, we get an interactive protocol with communication and computational complexity $\mathrm{polylog}(T) = \mathsf{poly}(n)$, where n is the input length, which would imply that $\mathrm{EXP} \subseteq \mathrm{PSPACE}$. We therefore consider *computational soundness*, i.e. we require that soundness holds only against computationally bounded provers (such proof systems are also known in the literature as *arguments*). Of course this will only make sense if an *honest* prover (aiming to prove a true statement) can produce a proof efficiently as well. In this context, we would like to understand which classes of computation adhere to succinct and efficient delegation schemes, and what amount of communication and interaction are required in order to accomplish this task.

Aside from its foundational value, the question of efficient delegation carries significance to applications as well. In many cases, computation today is becoming asymmetric, with lightweight local computation, and with large computational tasks performed off-site (e.g. by a cloud server). The ability to verify that the computation is carried out correctly without investing significant computational resources is obviously useful in this situation.

Probabilistically checkable proofs (PCPs) [AS98, ALM$^+$98] have proven to be an invaluable tool for the task of constructing efficient arguments, since they allow verification with very little communication. In the PCP model, the prover writes a proof π for the statement $x \in L$, but the verifier only needs to read very few locations in that proof (chosen according to a specific distribution) in order to be convinced of the correctness of the statement. Kilian [Kil92] translated this into an argument system using *collision resistant hash functions* (CRH), which allow to produce a succinct computationally binding commitment to a long string (the PCP proof string π), and also to succinctly reveal specific locations of the committed string, via a technique known as Merkle Trees. Kilian's protocol requires to exchange 4 messages between the prover and the verifier. Micali [Mic94] showed that in the random oracle model this can be done without interaction at all, but relied heavily on "programming" the random oracle.

Aiello et al. [ABOR00] proposed an approach to reduce the message complexity to 2, i.e. challenge-response, using a computational private information retrieval (PIR) scheme. For simplicity, in what follows, we explain their approach using a fully homomorphic encryption scheme (FHE), which is a stronger object than a PIR scheme.[2] Loosely speaking, they suggest to encrypt the PCP queries using an FHE scheme. In such a scheme, computation can be performed under the encryption without compromising security, so the transition $\mathsf{Enc}_{\mathsf{pk}}(x) \to \mathsf{Enc}_{\mathsf{pk}}(f(x))$ can be performed publicly with computational complexity which is comparable to that of f. The intuition is that if each PCP query is encrypted under an independently generated public key, a cheating prover should not be able to correlate the answers and will therefore respond as if according to a predetermined string π. Unfortunately, it was later shown in [DLN$^+$01], that this approach might not be sound, i.e., there may exist encryption schemes and PCP schemes for which this approach fails to produce a sound delegation scheme. Recently in [DHRW16], specific counter examples were presented. Intuitively, the problem is that this method does not necessarily force the adversary to answer each query "locally" without looking at the other encrypted queries; rather, it only guarantees that the *marginal* distribution of answers under any subset of public keys does not depend on the queries encrypted under other public keys. It had been established that the two are not the same, with the notable example being "spooky interactions" in quantum information theory.

We note that the approaches outlined above (as well as the approach we use in this work) relies on PCPs. This should not come as a surprise, given the work of Rothblum and Vadhan [RV10], which, loosely speaking, shows that any delegation scheme whose security is based on a standard (falsifiable [Nao03]) assumption, inherently uses a PCP (implicitly or explicitly).[3]

[1] Some subtleties arise when formalizing this problem, e.g. the time it takes to read the input x. These will be discussed below.

[2] In the technical sections we explain the Aiello et al. [ABOR00] proposal using the PIR terminology.

[3] An exception are delegation schemes in the preprocessing model, where the verifier must run a long pre-processing phase in order to verify

The work of Goldwasser, Kalai and Rothblum [GKR08], combined with the work of Kalai and Raz [KR09], give a 2-message delegation scheme for the class of logspace-uniform NC computations, using a super-polynomially hard PIR/FHE scheme. More specifically, [GKR08] construct an interactive proof for such languages (where the prover is efficient), and [KR09] show how to compile the interactive proof of [GKR08] (and interactive proofs in general) into a 2-message delegation scheme using a super-polynomially hard PIR/FHE scheme. Very recently, and in parallel to our work, Dwork, Naor and Rothblum [DNR16] showed that in certain situations (which include the [GKR08] scheme) it is possible to improve the compiler of [KR09] and only rely on polynomial hardness of the PIR/FHE scheme. To date, this (along with our work) is the only 2-message delegation scheme that is proven secure under what's known as a *falsifiable* (or a *complexity*) assumption [Nao03, GK16].

Kalai, Raz and Rothblum [KRR13] showed that the approach in [ABOR00] is sound, so long as the underlying PCP is sound against (statistical) *no-signaling* provers. A no-signaling prover does not have to respond according to a prescribed string π, rather for any query set Q it answers according to some distribution A. However, for every two query sets Q, Q', the marginal distribution of answers on the intersection $Q \cap Q'$ are identically distributed, whether Q or Q' had been asked. A statistical no-signaling prover is allowed to deviate, but only by a small statistical distance. It was shown in [KRR13] that a no-signaling PCP allows to apply the [ABOR00] approach to achieve a sound 2-message delegation, and in [KRR14] such a statistical no-signaling proof system had been presented, thus achieving a 2-message delegation scheme for all $\mathsf{TIME}(T)$ computation. The underlying computational assumption (to achieve the best parameters) was again super-polynomially hard PIR/FHE.

Kalai and Paneth [KP15] extended this result to the RAM computational model. They showed that time T RAM computations can be verified using proofs of length $\mathrm{polylog}(T)$, under the same computational assumptions. RAM machines more accurately describe real-world computations and in addition this model allows for a clean formal treatment of the input length and the non-uniformity of the computation. It is obvious that verification cannot succeed unless the verifier reads the input and knows what is the computation that is being verified (both can be treated in a unified way by considering a universal machine). In the RAM model, these can be encoded in the initial database (= memory array) of the machine. In the [KP15] model, the initial database can be preprocessed ahead of time, and a short digest is produced. The verifier only needs to know this digest in order to verify the computation. Thus, one can consider the case where the verifier constructs the digest himself, which is equivalent to the previous model where the verifier's complexity can depend on the input. Most importantly, in the RAM model, the verifier can use the same digest to verify many different computations. Moreover, the computations may not only read from the memory, but also write to the memory, in which case a new digest is produced, and the verifier can efficiently verify the validity of both the output of the computation and the new digest. Finally, we note that RAM delegation allows us to consider cases where the digest is produced separately, or even maliciously and provide more extensive security guarantees.

We note that this model of RAM delegation generalizes the model of memory delegation [CKLR11], where the verifier also saves a short digest of the memory and verifies computations using only the digest. However, in RAM delegation the prover's runtime is proportional to the RAM runtime of the computation M, whereas in memory delegation the prover's runtime is proportional to the size of the circuit corresponding to the RAM machine M. The former can be independent of the total size of the database (= memory) and thus be significantly shorter than the latter, which must be at least linear in the memory size. Further, the security guarantee we provide is stronger than the notion suggested for memory delegation.

Adaptivity and the Hunt for a Completely Non-Interactive Protocol. As explained above, several works in recent years established that delegation is achievable in as little as two messages, *dare we even hope for anything better?* Indeed, several works showed that it is possible to delegate even non-deterministic computations [DFH12, BCC^{+}14], and furthermore, the work of [BCCT12] even constructs a *non-interactive* delegation scheme for non-deterministic computations, assuming only a *common setup*. However, all these works rely on knowledge assumptions. Such assumptions, are not only not falsifiable [Nao03], but are of a very different nature than standard hardness assumptions. Gentry and Wichs [GW11] showed that for such languages it will be hard to achieve a non-interactive protocol under polynomial-time standard assumptions.

One would hope that the techniques of [GKR08, KRR14, KP15] could be adapted to imply a non-interactive protocol (with setup). Indeed, the first message of these delegation schemes relies only superficially on the input and

proofs. Such schemes do not need to use PCPs, as demonstrated by the works of [GGP10, CKV10, AIK10].

on the function being computed, essentially only (an upper bound on) the time complexity of the computation needs to be known in order to produce the first message. One would hope that the first message can be generated once and for all as setup, and would allow for verification of arbitrary statements.

However, the proof of security is inherently *non-adaptive*. Namely, even though the input is not required in order to produce the first message, security is not guaranteed when the adversary gets to choose the instance after seeing the first message. Technically, this is because the proof of security relies on extracting an entire computation table from the adversary, and in order to do this, the adversary is rewound and made to respond on different queries with regards to the same input. Obviously this technique is not applicable in an *adaptive* setting.

Note that proving adaptive security will make the delegation scheme *non-interactive* (with setup), since adaptive security allows to generate the first message during setup. However, verification will still require the secret randomness that was used to produce the first message. Therefore, the resulting scheme will not enjoy public verifiability as the aforementioned extractability-based scheme [BCCT12]. Moreover, such a scheme will only be secure in a model where malicious provers do not learn whether their maliciously crafted proofs are accepted or not; or alternatively, where the setup is generated anew if such leakage occurs. Our construction will inherit this limitation.

We note that one could try to get adaptive soundness by a "brute force" approach using *complexity leveraging*: The observation is that we can always guess the instance that the adversary will choose with probability 2^n. Therefore, if we pick the parameters of the scheme such that we are guaranteed soundness even with this slowdown, then adaptive security will follow. However, this method, aside from being exorbitant in terms of communication, computation and underlying hardness assumption, requires that there is a predetermined polynomial upper bound on the length of the input. This method is therefore completely inapplicable if we intend to support arbitrary polynomial time computations on arbitrary polynomial length inputs.

1.1 Our Results

We present a 2-message *adaptively* secure delegation scheme for RAM computations, under standard *polynomial* hardness assumptions. In particular, our scheme is *non-interactive* assuming each verifier is associated with a public key which is generated ahead of time (e.g. via a public-key infrastructure). The prover generates a proof respective to the public key of the verifier it wants to convince.

Our scheme works in the aforementioned RAM model: Given the public key of the verifier, the prover can prove the correctness of the computation of any RAM machine M on any database D. More specifically, the prover can produce a proof that M^D outputs a value y after T computational steps, for all $T = \mathsf{poly}(\lambda)$, with λ denoting the *security parameter*.[4] To verify a proof efficiently, the database D needs to be preprocessed via an efficient deterministic procedure to create a digest d of length λ. This can be done ahead of time and does not depend on the machine M.[5] Then the verification process can be performed as a polynomial time procedure taking as input the digest d, the machine M, the output y and the proof.

We note that a RAM computation M^D may alter the database D. Therefore the prover, along with the output y, will also output a digest d_{new} of the new updated database, and will prove that both y and d_{new} are correct.

In order to be able to support *any* polynomial running time, we allow any $T \leq 2^\lambda$. The running time of the prover is $\mathsf{poly}(T)$, and the length of the proof is $p(\lambda) \cdot \mathrm{polylog}(T) \leq p'(\lambda)$ for *fixed* polynomials p, p'. Namely, the length of the proof is a priori bounded by $p'(\lambda)$ and does not depend on the computation. It follows that the verification complexity is independent of T, and there is a fixed polynomial (in λ) upper bound on the running time of the online phase (assuming that the description of M does not grow asymptotically, say M is a universal machine). In the formal description of the model in Section 3, it is convenient for us to consider the proof as containing three pieces: The output of the machine y, the new digest of the database at the end of the computation d_{new}, and the proof string itself pf.

Our notion of security is an adaptive version of the definition of [KP15]. Their definition goes beyond just requiring that an efficient adversary cannot prove a false statement on any actual computation, i.e. on any machine M and database D, but rather requires that the prover cannot prove contradicting statements on any *digest*, even one that is

[4] The security parameter represents the asymptotics of efficient computations since the input size is irrelevant here. An efficient procedure is one that runs in time $\mathsf{poly}(\lambda)$.

[5] Our digest, as in [KP15], is simply a tree commitment of the database D.

generated maliciously and does not correspond to any database. Namely, a prover cannot find a digest d, machine M, and two accepting proofs for two different outputs of M^D. This means that even if the verifier has been cheated and given a malicious digest, there is still a well defined "functionality" associated with this digest for any machine M. This makes the separation between the offline and online stages even more substantial.

The soundness of our scheme relies on the *polynomial* hardness of a 2-message private information retrieval (PIR) scheme, or a homomorphic encryption scheme for a sufficiently rich class of functions (the latter implies the former). Unlike previous works, we do not require super-polynomial hardness of our underlying primitive. We note that in the independent work of [DNR16], they present a delegation scheme under polynomial hardness assumptions, but only for NC circuits and only in the non-adaptive setting.

We mention again that our delegation scheme is not publicly verifiable (similarly to previous works [GKR08, KRR13, KRR14, KP15] and the concurrent work of [DNR16]). There is a recent line of work that constructs publicly verifiable delegation schemes for deterministic computations [PR14, KLW15], as well as for RAM computations [BGL$^+$15, CHJV15, CCHR15, ACC$^+$15, CH16, CCC$^+$16]. However, all these works rely on indistinguishability obfuscation, except the work of [PR14] which relies on new assumptions on multi-linear maps. It is not clear, however, whether secure indistinguishability obfuscators actually exist, as current proposals of multi-linear maps, as well as obfuscation candidates, appear to have flaws. Nonetheless, even under these very strong assumptions, these aforementioned schemes require two messages, and it is not known how to achieve a non-interactive scheme as we do in this work.

As a main tool in our constructions, we define and construct *computationally* no-signaling PCPs in Section 4. Recalling that in statistical no-signaling, given sets of queries Q, Q', the marginal distribution of prover answers on $Q \cap Q'$ had to be statistically close whether the prover was asked on Q or on Q', now we only require that these distributions are *computationally indistinguishable*. This captures a potentially much larger set of cheating provers. Furthermore, our PCPs are adaptively secure, allowing the prover to first see the PCP query and then choose the instance to be proven, so long as computational no-signaling (CNS) holds even conditioned on the chosen instance. Whereas many parts in the construction of the computational no-signaling PCP mirror ones from previous works, and especially [KRR14], we are able to abstract the proof in a new way, and provide a somewhat shorter and more modular proof. The transition from PCPs to a delegation scheme in Section 5 is similar to [KP15], but the proof is quite different. Along the way, we are able to simplify a significant component that is inherited all the way back from [GKR08] (specifically, we do away with delegating the construction of the low degree extension to the prover). We view these as additional contributions of our work, deepening our understanding of the interplay of the different components in the proof. See more details in Section 1.2 below.

Batch delegation for NP. Our new techniques enable us to make progress on the task of delegating NP computations as well, but unfortunately only in the 2-message non-adaptive setting. We construct a 2-message (non-adaptive) batch delegation for NP which allows a prover to prove that $x_1, \ldots, x_k$ are all in $L \in \mathsf{NP}$, where the communication complexity is $m \cdot \mathsf{poly}(\lambda)$ with m being the length of a *single* witness. The security of this scheme is also based on the same assumption as our RAM delegation scheme, i.e., polynomially hardness of the underlying PIR scheme. Previously this was only achievable under knowledge assumptions (or in the Random Oracle Model). This contribution is described in Section 6.

1.2 Our Techniques

We make novel technical contributions in two aspects: First, in constructing a PCP scheme that is secure against *adaptive* and *computational* no-signaling cheating provers, as opposed to non-adaptive and statistical no-signaling provers as in previous work. Second, in converting such a PCP into protocols for adaptively secure RAM delegation or batched NP delegation. Previous proof techniques were insufficient for this, even given the enhanced PCPs. We will start by describing the delegation protocols, assuming we have a PCP at hand. Then, once we realize what properties of the PCP are actually required for the construction, we will describe the former contribution.

We will consider PCP systems for 3SAT with a standard completeness property, and will be interested in their performance against computational no-signaling (CNS) adaptive provers. A CNS adaptive prover P is a possibly randomized function, that takes a query set Q as input, and outputs a 3CNF formula φ as well as a set of answers A.

We require that for every two query sets Q, Q', and for (φ, A) distributed according to $P(Q)$ and (φ', A') distributed according to $P(Q')$, it holds that $(\varphi, A|_{Q\cap Q'}) \approx (\varphi', A'|_{Q\cap Q'})$, where "$\approx$" denotes computational indistinguishability. Clearly, the honest prover that answers according to a prescribed proof string has this property (where "$\approx$" can be replaced with equality). In the non-adaptive setting, which has been considered in previous works, φ is known ahead of time to both the prover and verifier and is not generated adaptively.

We start with recalling the aforementioned [ABOR00] delegation approach: generate a set of PCP queries $\{q_i\}$ according to the PCP verifier distribution. Then encrypt the queries using a homomorphic encryption scheme, each using its own key, and send the encrypted queries $\{\mathsf{Enc}_{\mathsf{pk}_i}(q_i)\}$ as the verifier's message. An honest prover will generate a PCP proof string π proving that φ is satisfiable, and use the homomorphism to compute $\{\mathsf{Enc}_{\mathsf{pk}_i}(\pi|_{q_i})\}$. The verifier can then decrypt and check that the PCP indeed accepts. We note that the generality of homomorphic encryption is not needed here, and all we need is to be able to obliviously select an entry out of a string, which is exactly the functionality provided by private information retrieval.

As for soundness, the important observation of [KRR13] is that any efficient prover, when converted to a PCP prover, must be CNS, since a non CNS prover would necessarily violate the security of the encryption scheme: If the distributions $A_1 = A|_{Q\cap Q'}$ and $A_2 = A'|_{Q\cap Q'}$ are distinguishable, then we can also distinguish between an encryption of $Q_1 = Q \setminus Q'$ and an encryption of $Q_2 = Q' \setminus Q$, which would violate the FHE security. This is done by taking the encryptions of the elements of Q_i (where the secret key is unknown), and appending the encryptions of $Q \cap Q'$ under a known secret key. Then we run the prover on the set of encrypted queries, and examine the prover's response using the distinguisher. If the distinguisher says A_1, this means we probably started with Q_1, and vice versa.

The [KRR14] approach to delegation was to construct a PCP that has soundness against no-signaling provers (statistical, in their case, but the distinction will not concern us at this point). Consider a computation, and consider a 3CNF formula φ that represents the evolution of this computation. Namely, consider the complete computation tableau, and assign a variable to each of the bits in it. A proof for a correct computation thus becomes a matter of proving that φ is satisfiable, when the variables that correspond to the initial state and the output are fixed to the respective input and output of the computation. Note that if φ is satisfiable then there is exactly one satisfying assignment which corresponds to the correct evolution of the computation.

The formula φ described above corresponds to a translation of the computation into a circuit. Thus, the prover complexity suffers accordingly (indeed the protocol of [KRR14] was designed for circuits and not for RAM machines). This was improved by [KP15], who considered RAM computations. Their approach was as follows: Rather than considering the state of the RAM memory database at every point in time, they only considered a *computationally binding commitment* to the memory contents. By using locally updatable commitments (Merkle Trees), it was possible to construct a formula φ that describes the evolution of the computation, and in particular the evolution of the committed memory database, in a compact manner, only requiring a fixed polynomial number of variables per time step, regardless of the length of the database. Since the commitments are only computationally binding, it was no longer true that φ only has one satisfying assignment. The guarantee that they get, however, is that finding two different satisfying assignments will break the underlying cryptographic primitive (a collision resistant hash function), and therefore cannot be done efficiently. In order to use this guarantee, [KP15] considered a strong notion of soundness for their PCP (which was already used as a stepping stone in [KRR14]): they showed that any successful no-signaling prover against their PCP can be efficiently converted into a *local assignment generator*, which is an algorithm that takes any sufficiently small subset of the variables of φ and outputs an assignment for these variables that does not violate any of the clauses in which they appear. The generator is allowed to give different responses for the same variable, depending on the other elements in the set, but the assignments have to be no-signaling in an analogous sense to above: intersecting sets of variables should induce the same *distribution* on the values (up to negligible statistical or computational distance). Combining this with computational binding, they get a way to extract an entire satisfying assignment for the formula, by starting from the initial state, and filling up the table sequentially and locally, each time asking about the next variable and the previous variables that determine its value. The no-signaling property guarantees that the distribution on each and every variable is independent of the set it belongs to, and computational binding guarantees that this distribution is in fact *constant*, otherwise the binding of the commitment is broken. Thus, assuming that they have a cheating prover that successfully proves two contradicting statements about the final state of M^D, this translates into two formulae φ_1, φ_2 that correspond to computations with the same initial state but different final state. Using the assignment generator they can find the point of divergence in the two computations which will necessarily imply that they can open a commitment in two different ways, thus breaking the underlying commitment.

Adaptivity, At Last. How can we generalize this approach to the adaptive setting? We now consider a PCP whose query distribution is independent of the instance in question (most known PCPs have this property), and we allow a cheating prover to choose the formula φ, for which it wants to provide a proof *adaptively*, based on the query set Q. This means that a prover who wants to make our lives harder will never even output the same formula twice. Indeed, we can generalize the previous technique, and construct a local assignment generator, but it will be severely crippled: Upon receiving a set of variables as input, the assignment generator will output an assignment for these variables *together with a formula* φ to which the assignment relates. Whereas all output formulae still relate to some RAM computation, we lose our anchor for constructing the computation tableau, namely the uniqueness of the initial state. The only ray of light in this situation is that the no-signaling property is still preserved for this assignment generator. Namely the marginal distributions on subsets of variables need to be computationally indistinguishable, regardless of the sets in which they are contained, and this indistinguishability holds even in the presence of the respective φ's (in particular, the φ distribution itself should be computationally indistinguishable for different values of query sets Q).

To see how to use these properties, let us recall our notion of security. We consider a prover that is able to adaptively output M, d (where d is an alleged digest, or commitment to the initial state), together with two possible alleged outputs for the computation M^D, and respective computationally no-signaling (CNS) PCP responses. This means that even though the adversary might change the formulae φ_1, φ_2, it is always the case that it outputs two formulae with the same initial state and different final state. Therefore, there has to exist a step in the computation where the two RAM computations diverge. Perhaps surprisingly, we can find this point of divergence even if the adversary tries to shift it around! The idea is to extend the local assignment generator into one that outputs φ_1, φ_2 at the same time, and allows to make local tests on variables of both formulae at the same time. This means that we can define a predicate of the sort "φ_1 agrees with φ_2 at step i", denote this predicate by θ_i, and test θ_i or even $(\theta_i \wedge \neg\theta_{i+1})$ using our assignment generator, since these predicates only depend on a small number of variables of the original φ_1, φ_2. The critical observation is that CNS should hold even with respect to the θ predicates, since they are efficiently computable from the variables of φ_1, φ_2. We know that θ_0 is true and θ_T is false, therefore from the CNS property, there must exist an i such that $(\theta_i \wedge \neg\theta_{i+1})$. Having found this i, we can look at the specific assignment of the φ_1, φ_2 variables to find the inconsistency and thus break the binding of the commitment scheme. This allows us to achieve an adaptive delegation scheme in spite of an elusive prover.

Efficiently Computing Low-Degree Extensions. In the verification process, it is necessary to compute a the low-degree extension (LDE) of φ. This is the case in our scheme, as well as in previous schemes [GKR08, KRR13, KRR14, KP15]. A low degree extension of a function is a multivariate low degree polynomial over a finite field whose restriction to a specific subset of the domain is exactly the original function. Usually the cardinality of the field is polylogarithmic in the input length. In our setting, we seek an LDE of the indicator function ϕ, which takes labels of 3 literals and checks whether the clause they define appears in φ. Recall that the verifier only has M, d, which implicitly defines φ, and in particular it can compute ϕ at any point, but computing the LDE requires a global view of ϕ that cannot necessarily be computed with the resources of the verifier. Note that the verifier cannot run in time $|\varphi|$ since this is proportional to the runtime of the RAM computation.

In previous works [KRR13, KRR14, KP15] there was an additional step to the delegation process, where the computation of the LDE itself had been delegated to the prover. A fairly complicated proof was required in order to show that the delegation of the LDE can be composed with the delegation of the computation. We are able to get rid of this step completely, and show that the LDE can be computed in polylogarithmic time given M, d. The idea is quite simple, and uses the uniformity (or symmetry) of RAM computations. See details in Section 5.1 (and in particular in Lemma 3).

Batch NP Delegation. We show that similar methods also allow us to achieve 2-message non-adaptive delegation protocols for batch NP verification. Specifically, to prove that all $x_1, \ldots, x_k \in L$, for some $L \in \mathsf{NP}$, we only require $m \cdot \mathsf{poly}(\lambda)$ bits of communication, where m is the length of a *single* witness for L (note that since we are in the non-adaptive setting, we can assume that m is known ahead of time).

To see how this is done, let us first consider the degenerate case where $k = 1$. In this case the solution is clear: The prover sends the witness w, together with a short proof of the NP verification process of (x, w) via the former delegation protocol. We emphasize that we must require adaptive security of the original protocol even to achieve selective security here, since the computation that is actually being verified depends on w which is adversarially chosen. We note that this protocol still has advantages over the trivial verification of (x, w) since its computational complexity is independent of the specific NP relation. Moreover, we note that prior to this work, such an NP delegation protocol, even for $k = 1$, was known only in the random oracle model or under knowledge assumptions.

Next, consider a slight variation of this protocol, where instead of sending w in the clear, the adversary encodes it into a RAM database, and sends a digest of this database along with the proof. Proving such a protocol is already beyond the abilities of our proof methods, since our assignment generator is local and therefore can only "see" a (possibly small) part of the witness at the same time. Since the actual value of the witness is not "anchored" in any way, we cannot go from local to global consistency. This can be fixed, however, if we extend our assignment generator to allow reading sets of $> m$ variables in one go, which will allow to read the witness value itself and thus bring us back to a situation similar to the previous adaptive delegation setting with the witness playing the role of the input. Doing this will require to make more PCP queries so as to allow assignments to additional variables, which makes the communication complexity proportional to m. However, these additional PCP queries can be "reused" to verify additional instances, which will allow the batch functionality as explained next.

Indeed this is the intuition behind the k-instance setting. Loosely speaking, the idea is that the prover, rather than sending all the witnesses in the clear (which is too long), will send a digest of the witnesses, along with a proof that these are accepting witnesses. From a similar argument to the one above, if the number of PCP queries is more than m, then we can can locally check that each witness is valid. We note that this intuition is an oversimplification, and there are several technical hurdles that we need to overcome to make this idea work. We refer the reader to Section 6 for details.

Constructing Adaptive PCPs. Finally, we attend to the construction of the adaptive PCP required for our protocols as described above. In fact, our PCP is exactly the same as the one used in [KRR13, KP15]. We had to prove soundness in the adaptive and computational no-signaling setting. Namely, we show that adaptive assignment generators (of the type we want) can be constructed given any computational no-signaling adaptive prover (as opposed to non-adaptive statistical no-signaling). The overall proof outline remains the same, and the repetitive parts of the proof are deferred to the appendix. Along the way we were able to simplify parts of the proof.

The PCP of [KRR14] is in fact a repetition of a more simple PCP. The repetition is required in order to amplify soundness. In this work we prove the soundness amplification in a modular way. Specifically, we present an amplification lemma that applies in general to no-signaling PCPs, and shows that a prover with noticeable success probability against a repeated PCP can be transformed into a prover with success probability almost 1 against a relaxed PCP verifier that only checks that some of the copies of the repetition verify correctly. This part is completely self contained (appears in Section 4.3), and is used in a black-box way in the remainder of the proof.

2 Preliminaries

2.1 Low Degree Extensions

Let $\mathbb{F}$ be a finite field, and let H be a subset of $\mathbb{F}$, and let m be some integer. If f is a function mapping $H^m \to \{0, 1\}$, then there exists a unique extension of f into a function $\hat{f} : \mathbb{F}^m \to \mathbb{F}$ (which agrees with f on H^m; i.e., $\hat{f}|_{H^m} \equiv f$), such that $\hat{f}$ is an m-variate polynomial of degree at most $(|H| - 1)$ in each variable. This function $\hat{f}$ is called the *low degree extension* of f.

Low-Degree Extensions of Strings. If x is a binary string of length n, we pick $H \subseteq \mathbb{F}$ such that $|H| = \log n$ and $|\mathbb{F}| = \mathrm{polylog}(n)$, and m such that $m = \log n / \log\log n$. By mapping $[n]$ into H^m (in lexicographical order), we can view x as a function mapping $H^m \to \{0, 1\}$, and we define x's low-degree extension accordingly.

Low-Degree Extensions of 3-CNF Formulas. If φ is a 3-CNF over a set V of n variables, then (with some ordering of V) an assignment to these variables is an n-bit string. As above, such an assignment can be represented by a function mapping $H^m \to \{0,1\}$, where each variable of φ is associated with an element of H^m. We similarly write a "clause indicator function" $\phi : H^{3m} \times \{0,1\}^3 \to \{0,1\}$ for φ:

$$\phi(v_1, v_2, v_3, b_1, b_2, b_3) = \begin{cases} 1 & \text{if the clause } v_1 = b_1 \vee v_2 = b_2 \vee v_3 = b_3 \text{ is in } \varphi \\ 0 & \text{otherwise} \end{cases}$$

In fact, we view ϕ as a function from $H^{3m+3} \to \{0,1\}$ by defining the output of ϕ as 0 if b_1, b_2, or b_3 is not in $\{0,1\}$. When we refer to the *low-degree extension of a formula* φ, we mean the low-degree extension of this clause indicator function ϕ.

In the following we assume that all algorithms have access to m, the set H and the field $\mathbb{F}$, and we assume that the elementary field operations over $\mathbb{F}$ have unit cost.

2.2 Cryptographic Primitives

We start with a standard definition of computational indistinguishability.

Definition 1 (Computational Indistinguishability)**.** *Two ensembles of distributions (or of random variables)* $X = \{X_\lambda\}_{\lambda\in\mathbb{N}}, Y = \{Y_\lambda\}_{\lambda\in\mathbb{N}}$ *are computationally indistinguishable, denoted* $X \approx Y$ *if for every* PPT *algorithm* $\mathcal{A}$ *it holds that*

$$\left|\Pr[\mathcal{A}(1^\lambda, X_\lambda) = 1] - \Pr[\mathcal{A}(1^\lambda, Y_\lambda) = 1]\right| = \mathsf{negl}(\lambda) \ .$$

We define length halving collision resistant hashing (we will not require more elaborate variants).

Definition 2. *A family of collision resistant hash functions is an ensemble of efficiently sampleable distributions* $\{\mathcal{H}_\lambda\}_{\lambda\in\mathbb{N}}$ *(i.e. given* 1^λ *one can efficiently sample from* $\mathcal{H}_\lambda$*) such that* $\mathcal{H}_\lambda \subseteq \left(\{0,1\}^{2\lambda} \to \{0,1\}^\lambda\right)$*, and for every probabilistic polynomial time algorithm* $\mathcal{A}$*:*

$$\Pr_{h\leftarrow\mathcal{H}_\lambda}\left[(x \neq x') \wedge (h(x) = h(x')) \;\middle|\; (x, x') \leftarrow \mathcal{A}(1^\lambda, h)\right] = \mathsf{negl}(\lambda) \ .$$

We define the notion of computational polylogarithmic PIR scheme below, and then proceed to derive a variant that will be useful for our construction.

Definition 3. *A* 2*-message private information retrieval (PIR) scheme over alphabet* $\Sigma = \Sigma_\lambda$*, with* $\log|\Sigma| \leq \mathsf{poly}(\lambda)$*, is a tuple of* PPT *algorithms* ($\mathsf{PIR.Send}$, $\mathsf{PIR.Respond}$, $\mathsf{PIR.Decode}$)*, where:*

- $(q, s) \leftarrow \mathsf{PIR.Send}(1^\lambda, i, N)$*: given* 1^λ *and some* $i, N \in \mathbb{N}$ *such that* $i \leq N$*, outputs a query string* q *and a secret state* s*.*
- $a \leftarrow \mathsf{PIR.Respond}(q, D)$*: given a query string* q *and a database* $D \in \Sigma^N$*, outputs a response string* a*.*
- $x \leftarrow \mathsf{PIR.Decode}(s, a)$*: given an answer* a *and state* s*, outputs an element* $x \in \Sigma$*.*

We say that the scheme is a polylogarithmic PIR if $|a| = \mathsf{poly}(\lambda, \log(N))$*. We say that it is (perfectly) correct if for all* $i \leq N \leq 2^\lambda$ *and* $D \in \Sigma^N$ *it holds that when setting* $(q, s) \leftarrow \mathsf{PIR.Send}(1^\lambda, i, N)$*,* $a \leftarrow \mathsf{PIR.Respond}(1^\lambda, q, D)$*, and* $x \leftarrow \mathsf{PIR.Decode}(1^\lambda, s, a)$*, then* $x = D[i]$ *with probability* 1*.*

We say that the scheme is secure if for any sequence of $N_\lambda = \mathsf{poly}(\lambda)$*,* $i_\lambda, i'_\lambda \leq N_\lambda$ *it holds that* $q \approx q'$*, where:* $(q, s) \leftarrow \mathsf{PIR.Send}(1^\lambda, i, N)$*,* $(q', s') \leftarrow \mathsf{PIR.Send}(1^\lambda, i', N)$*.*

It had been shown in [IKO05] that a succinct 2-message PIR implies the existence of a family of collision resistant hash functions. Succinct PIR can be constructed based on the ϕ-hiding assumption [CMS99] or based on the existence of leveled fully homomorphic encryption scheme [Gen09], which in turn can be based on the (polynomial) hardness of approximating short-vector lattice problems (SIVP and GapSVP) to within a polynomial factor in *worst case* lattices [BV14].

We define a variant that we call "succinct" PIR, where only a bound on the size of the database N needs to be known. This notion will be useful for our construction and can be derived from the standard notion of PIR as we explain below.

Definition 4. *A* succinct 2*-message private information retrieval (PIR) scheme over alphabet* $\Sigma = \Sigma_\lambda$*, with* $\log|\Sigma| \leq \mathsf{poly}(\lambda)$*, is a tuple of* PPT *algorithms* $(\mathsf{ScPIR.Send}, \mathsf{ScPIR.Respond}, \mathsf{ScPIR.Decode})$*, where:*

- $(q, s) \leftarrow \mathsf{ScPIR.Send}(1^\lambda, i)$*: given* 1^λ *and some* $i \leq 2^\lambda$*, outputs a query string* q *and a secret state* s*.*
- $a \leftarrow \mathsf{ScPIR.Respond}(q, D)$*: given a query string* q *and a database* $D \in \Sigma^{\leq 2^\lambda}$*, outputs a response string* a*.*
- $x \leftarrow \mathsf{ScPIR.Decode}(s, a)$*: given an answer* a *and state* s*, outputs an element* $x \in \Sigma$*.*

We say that the scheme is succinct PIR if $|a| = \mathsf{poly}(\lambda)$*. We say that it is (perfectly) correct if for all* $i \leq 2^\lambda$ *and* $D \in \Sigma^{\leq 2^\lambda}$ *with* $|D| \geq i$ *it holds that when setting* $(q, s) \leftarrow \mathsf{ScPIR.Send}(1^\lambda, i)$*,* $a \leftarrow \mathsf{ScPIR.Respond}(1^\lambda, q, D)$*, and* $x \leftarrow \mathsf{ScPIR.Decode}(1^\lambda, s, a)$*, then* $x = D[i]$ *with probability* 1*.*

We say that the scheme is secure if for any sequences i_λ, i'_λ *it holds that* $q \approx q'$*, where:* $(q, s) \leftarrow \mathsf{ScPIR.Send}(1^\lambda, i)$*,* $(q', s') \leftarrow \mathsf{ScPIR.Send}(1^\lambda, i')$*.*

Whereas the definition here is syntactically stronger (since N does not need to be known), succinct PIR can be constructed from any polylogarithmic PIR scheme as in Definition 3. This can be done by considering all databases of size 2^t, for $t = 1, \dots, \lambda$, and running the query algorithm on all of them in parallel. This will incur an overhead of λ which does not change the polynomial dependence on λ. Furthermore, specific constructions of PIR, e.g. from leveled homomorphic encryption, can be adapted to the new definition without overhead at all.

3 Our Model: Public-Key Delegation for RAM Computations

In this section we motivate and formally define the model for adaptive non-interactive RAM delegation.

RAM Computation. We consider the standard model of RAM computation where a program M can access an initial database $D \in \{0,1\}^n$. We denote by M^D an execution of the program M with initial database D. For a bit $y \in \{0,1\}$ and for a string $D_{\mathsf{new}} \in \{0,1\}^n$ we also use the notation $y \leftarrow M^{D \to D_{\mathsf{new}}}$ to denote that y is the output of the program M on initial database D, and D_{new} is the final database contents after the execution. For simplicity we think only of RAM programs that output a single bit.[6]

RAM Delegation. Delegation, or verifiable outsourcing, is concerned with a client that wishes to know the result of some computation without performing it himself. Thus the computation is delegated to a prover (or worker), that performs the computation and sends the output back to the client. However, the client wishes to ensure that the prover indeed sent the correct output, and to this end it sends to the prover a challenge message, and the prover responds with the output, accompanied by a proof. One can show that such a proof can only be computationally sound, i.e. the protocol relies on the computational limitations of the prover.

As for efficiency, denoting the complexity of the computation by T, we would like the client's work to only depend (poly)logarithmically on T. This goal might seem beside the point, since the client should at least be able to read the input and have some description of the function being computed. These could already be super-polylogarithmic in the T. Indeed, in almost all previous works on Turing machine delegation, the client's running time was allowed to be linear in the input size n, and in the description of the computation (which was assumed to be uniform and thus asymptotically constant).

This approach can be generalized even further when thinking about RAM computations as in [KP15]. One can think of a uniform RAM machine (w.l.o.g this can even be a universal machine), where the input and code are stored in the RAM database (a.k.a memory or array) itself. In the RAM delegation model, the initial database is preprocessed ahead of time, so as to produce a succinct digest. Given this digest, any computation on the given database can be verified in (fixed) $\mathrm{polylog}(T)$ complexity. One can consider, as above, a client that constructs the database himself (and thus runs in time that depends on n and the description of the computation), but this modeling is significantly adventagous when multiple computations are ran on the same database, or in an offline-online setting.

[6] A program that outputs multiple bits can be simulated by executing several programs in parallel, or by writing the output directly to the database.

A Public-Key Delegation Scheme. In previous works, the soundness of the protocol was only guaranteed if the input (which in the RAM setting contains the machine and the database) was known before the challenge message is sent. If a polynomial upper bound on the length of the input was known, then complexity leveraging could have been used to remove this restriction, at the expense of a significant increase in the communication complexity and significant strengthening of the underlying hardness assumption (namely, sub-exponential hardness would be needed).

In this work, we present an adaptively secure delegation protocol, where nothing about the input (machine and database) needs to be known in order to generate the client's challenge message, *not even the input size*. This amounts to a significant qualitative difference in the applicability of the protocol. In fact, the adaptive protocol can be casted as *public key scheme* for delegation of computation. In particular, a client can generate at their own leisure a pair of secret key sk and public key pk. These keys are persistent and can be used throughout the life of the system (assuming that an adversary cannot learn whether maliciously generated proofs are accepted). As usual, the public key is posted for everybody to use, and the secret key is kept private by the client. A server can then prove any statement about arbitrary computations using a client's public key, without the client's involvement. The proof can then be posted for the client to verify whenever it wants. For the verification process, the client needs the description of the computation, which is represented by the RAM machine M and the digest d of the database in the beginning of the computation. It needs to receive from the prover the final state of the computation, represented as an output y, a digest d' of the final state of the database, and the running time T, as well as the proof of correctness pf. As mentioned above, the running time of the verifier can only depend polylogarithmically on the running time of the computation T. One could consider a stronger notion where the verification is done using public information, as opposed to a secret key. This variant does not require any change in the syntax of our scheme, only in the definition of soundness (i.e. allowing our "secret key" to be known to the attacker). Unfortunately, we cannot achieve this stronger notion of security and therefore our definition only considers verification using a secret key.

For correctness, we require that if indeed the initial digest was properly generated, then the prover can produce an accepting proof using only the client's public key (and the description of the computation), and that he can do so in $\mathsf{poly}(T)$ time, i.e. the complexity of proving is comparable to the complexity of computing. The formal syntax of the primitive is given below. Note that we allow a set of public parameters to be generated ahead of time and can be used by all parties, which will allow us to strengthen the notion of soundness we can provide. We remark that in our construction, the public parameters contain only a collision resistant hash function, which can be generated using public randomness from lattice assumptions or discrete log.

The Syntax of a Public-Key Delegation Scheme. A public key delegation scheme consists of a tuple of PPT algorithms

$$(\mathsf{Setup}, \mathsf{ProcessDB}, \mathsf{KeyGen}, \mathsf{Prove}, \mathsf{Verify})$$

with the following syntax and efficiency:

- $\mathsf{Setup}(1^\lambda) \to \mathsf{pp}$: A PPT algorithm that takes as input a security parameter 1^λ, and outputs public parameters pp.

- $\mathsf{ProcessDB}(\mathsf{pp}, D) \to \mathsf{dt}, \mathsf{d}$: A deterministic algorithm running in time $|D| \cdot \mathsf{poly}(\lambda)$ that takes as input public parameters pp and database D, and outputs the processed data dt of size $|D| \cdot \mathsf{poly}(\lambda)$ and a digest d of size $\mathsf{poly}(\lambda)$. We will write $\mathsf{Digest}(\mathsf{pp}, D)$ to denote just d. The processed dt will contain a copy of D, which we will denote $\mathsf{dt}.D$.

- $\mathsf{KeyGen}(1^\lambda) \to (\mathsf{pk}, \mathsf{sk})$: A randomized polynomial-time algorithm that takes as input the security parameter 1^λ (in unary representation), and outputs a public key pk and a secret key sk.

- $\mathsf{Prove}^{\mathsf{dt}}(\mathsf{pp}, \mathsf{pk}, M) \to (y, \mathsf{d}_{\mathsf{new}}, T, \mathsf{pf})$: A deterministic algorithm, that takes as input public parameters pp, a public key pk and a RAM machine M, it runs in time $\mathsf{poly}(T, \lambda)$, where $T = \textsc{Time}(M^{\mathsf{dt}.D})$ is the runtime of the RAM machine M with database $\mathsf{dt}.D$. Prove then outputs the result y of executing $M^{\mathsf{dt}.D}$, a digest $\mathsf{d}_{\mathsf{new}}$ of the resulting database contents, the runtime T of the computation, and a proof pf of size $\mathsf{poly}(\lambda)$.

- $\mathsf{Verify}(\mathsf{sk}, \mathsf{pp}, (M, \mathsf{d}, y, \mathsf{d}_{\mathsf{new}}, T), \mathsf{pf}) \to b$: A deterministic algorithm running in time $|M| \cdot \mathsf{poly}(\lambda)$ that outputs an acceptance bit b.

Soundness. The basic requirement of soundness is that a prover cannot convince the verifier of a false statement. Our public-key structure allows us to consider *adaptive* security, where an adversary is allowed to choose the computation it wants to forge on *after* seeing the public parameters and the public key. No prior work was able to achieve such a strong notion of security under a standard cryptographic assumption (especially for unbounded input length).

The weakest flavor of adaptive soundness that we can require is one where a cheating prover attempts to generate a database D and a RAM machine M, such that it can convince the verifier of a false output of the computation M^D, providing Verify with an honestly generated digest of D. This corresponds to a setting where the client himself generated the digest ahead of time and kept it for verification time (or alternatively the digest had been generated or certified by a trusted party). However, we can consider (and achieve) an even stronger notion of soundness.

In our final soundness definition, even if the digest d upon which verification is performed is completely spoofed by the cheating prover (i.e. does not necessarily correspond to any database), the prover still cannot prove the correctness of two different outputs with respect to the same RAM machine M and digest d. That is, the cheating prover gets access to the public parameters and the public key of the verifier. It will generate a machine M and a digest d, and two different output values y_1, y_2, each alongside a value for the final digest of the computation $\mathsf{d}'_1, \mathsf{d}'_2$. It will produce proofs $\mathsf{pf}_1, \mathsf{pf}_2$, and will try to convince the verifier that there exists a time bound T such that applying M to a database whose digest is d results in output y_i and digest d'_i for both $i = 1, 2$.

We also require that the prover also outputs T in *unary* to prevent a degenerate case where the prover claims to have performed a very complex computation that the security reduction itself cannot recreate. This requirement can be removed, and the prover can be allowed to prove arbitrary time $T \leq 2^\lambda$ computations, but only if we assume stronger (super-polynomial) cryptographic hardness. A formal definition follows.

Definition 5 (RAM Delegation)**.** *A non-interactive adaptive RAM delegation scheme*

$$(\mathsf{Setup}, \mathsf{ProcessDB}, \mathsf{KeyGen}, \mathsf{Prove}, \mathsf{Verify})$$

must satisfy the following properties.

- *Correctness. For every RAM machine M and database D, such that $M^{D \to D_{\mathsf{new}}} \to y$, it holds that*

$$\Pr\left[y' = y \wedge \mathsf{d}' = \mathsf{Digest}(\mathsf{pp}, D_{\mathsf{new}})\right] = 1$$

in the probability space defined by sampling

$$\begin{array}{l} \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda) \\ (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda) \\ (\mathsf{dt}, \mathsf{d}) \leftarrow \mathsf{ProcessDB}(\mathsf{pp}, D) \\ (y', \mathsf{d}', T, \mathsf{pf}) \leftarrow \mathsf{Prove}^{\mathsf{dt}}(\mathsf{pp}, \mathsf{pk}, M) \end{array}$$

- *Completeness. For every RAM machine M and database D such that* $\mathrm{TIME}(M^D) \leq T \leq 2^\lambda$*, it holds that*

$$\Pr\left[\mathsf{Verify}(\mathsf{sk}, \mathsf{pp}, (M, \mathsf{d}, y, \mathsf{d}_{\mathsf{new}}, T), \mathsf{pf}) = 1\right] = 1$$

in the probability space defined by sampling

$$\begin{array}{l} \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda) \\ (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda) \\ (\mathsf{dt}, \mathsf{d}) \leftarrow \mathsf{ProcessDB}(\mathsf{pp}, D) \\ (y, \mathsf{d}_{\mathsf{new}}, T, \mathsf{pf}) \leftarrow \mathsf{Prove}^{\mathsf{dt}}(\mathsf{pp}, \mathsf{pk}, M) \end{array}$$

- *Soundness. For every ensemble $\mathcal{P}^* = \{P^*_\lambda\}_{\lambda \in \mathbb{N}}$ of poly-sized probabilistic circuits,*

$$\Pr\left[\begin{array}{l} \mathsf{Verify}(\mathsf{sk}, \mathsf{pp}, (M, \mathsf{d}, y_1, \mathsf{d}'_1, T), \mathsf{pf}'_1) = 1 \ \wedge \\ \mathsf{Verify}(\mathsf{sk}, \mathsf{pp}, (M, \mathsf{d}, y_2, \mathsf{d}'_2, T), \mathsf{pf}'_2) = 1 \ \wedge \\ (y_1, \mathsf{d}'_1) \neq (y_2, \mathsf{d}'_2) \end{array}\right] \leq \mathsf{negl}(\lambda)$$

in the probability space defined by sampling

$$\begin{aligned}&\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)\\&(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda)\\&(M, \mathsf{d}, y_1, \mathsf{d}'_1, \mathsf{pf}'_1, y_2, \mathsf{d}'_2, \mathsf{pf}'_2, 1^T) \leftarrow P^*_\lambda(\mathsf{pp}, \mathsf{pk})\end{aligned}$$

Stronger Notions to Consider for Future Work. As explained above, we cannot achieve (and therefore do not consider) *public verifiability* where verification does not require a secret key. Our notion of security does not even allow a "chosen proof attack" where a cheating prover gets to interact with a verifier and observe its outputs on maliciously crafted proofs. Indeed, our construction (as well as any other known construction from standard assumptions) falls short of achieving any type of security in this setting.

With regards to the definition of soundness, a desirable feature would be to forbid the cheating prover from proving inconsistent statement on related computations (as opposed to inconsistencies on the same computation as in our definition above). Namely, the prover should not be able to prove that $f(x) = 0$ and also that $g(x) = 0$ if $g := f + 1$. An even more ambitious goal would be to require that if a cheating prover successfully proved a statement with respect to a digest, then there exists a database that corresponds to this digest. The latter notion would imply an extension of our proof system beyond polynomial time computation and allow verification of non-deterministic computation.

4 Adaptive PCPs

In this section, we present a new notion of *adaptive PCPs*. These are probabilistically checkable proofs [AS98, ALM$^+$98] where a cheating prover is allowed to first see all of the PCP queries, and only then *adaptively* choose both the (non-accepting) input on which it wants to generate a cheating proof, as well as the proof string itself. Naturally, it is impossible to achieve security against such adaptive adversaries. Therefore, we limit the cheating provers to be *computationally no-signaling* (CNS).

Adaptive CNS provers must satisfy the property that for any two sets of PCP queries $Q' \subseteq Q$ (such that Q is not too large), the distributions of answers on Q' are computationally indistinguishable, even given the respective adaptively selected inputs. Namely, an adaptive CNS cheating prover, must satisfy that for every such Q, Q' it holds that

$$(x, A|_{Q'}) \approx (x', A'),$$

where $(x, A) \leftarrow \mathsf{Prover}^*(Q)$ and $(x', A') \leftarrow \mathsf{Prover}^*(Q')$. We note that this means that unlike the conventional definition of PCP, the PCP queries are chosen independently of the instance. This might seem weird at first glance, but in fact known PCP constructions usually have this property. All that is required is some upper bound on the length of the instance, which we take to be 2^λ, where λ is our security parameter.

For technical reasons, we need to consider cheating provers that are somewhat more involved. As explained in Section 3, to prove soundness of our RAM delegation scheme, we need to rule out the adversary's ability to come up with a RAM computation M, a digest d, two outputs y_1, y_2, with corresponding new digests, $\mathsf{d}_1, \mathsf{d}_2$, together with two valid proofs. As we show in Section 5, this translates, via our construction, to a PCP prover Prover^* that given a set of queries Q, produces two such instances $(M, \mathsf{d}, y_1, \mathsf{d}_1)$ and $(M, \mathsf{d}, y_2, \mathsf{d}_2)$ (which we will think of as two 3CNFs), together with answers A_1 and A_2, respectively, corresponding to the PCP queries Q.

We thus consider any PCP cheating prover, Prover^*, that for any two sets of PCP queries $Q' \subseteq Q$ (such that Q is not too large), outputs $(\varphi_1, \varphi_2, A_1, A_2) \leftarrow \mathsf{Prover}^*(Q)$ and $(\varphi'_1, \varphi'_2, A'_1, A'_2) \leftarrow \mathsf{Prover}^*(Q')$, and we require that

$$(\varphi_1, \varphi_2, (A_1)|_{Q'}, (A_2)|_{Q'}) \approx (\varphi'_1, \varphi'_2, A'_1, A'_2)$$

We note that since the prover is adaptive, working on each of the instances separately is insufficient: the prover might output pairs of inconsistent instances, but if we look at the marginal distribution of each element in the pair, these distributions might even be identical. Therefore, our soundness crucially relies on the cheating prover outputting two instances together with PCP answers for both.

The notion of soundness that we prove is a refinement of notions presented in previous works[7]. We prove that an adaptive CNS adversary that convinces our PCP verifier with noticeable probability implies a *local assignment generator*. A local assignment generator is an algorithm that takes as input a set of variables W for a 3SAT formula, and outputs a pair of formulae φ_1, φ_2, together with assignments for the variables W in both φ_1, φ_2 that does not refute the formulae. Furthermore, the distribution of the formulae should be computationally indistinguishable from the one output by the cheating prover, and the local assignment needs to be (computational) no-signaling (otherwise the task may be trivial).

To understand why this is a meaningful notion of soundness, recall that a cheating prover outputs formulae that correspond to inconsistent statements. A local assignment generator will allow us to track down the point in the computation where the two statements diverge, and show that being able to prove divergent statements allows to break the underlying hardness assumption. See details in Section 5. The crux of our approach is that even though the adversary is allowed to adaptively change the formulae its outputting, the no-signaling property guarantees that it cannot shift around the point of divergence.

Organization of This Section. We start in Section 4.1 with the definition of adaptive PCP, and our notion of adaptive cheating provers. Then, in Section 4.2 we present Theorem 1, which states that there exists a PCP system, that considers adaptive CNS cheating provers, which has (local) soundness guarantees. The PCP construction itself, and parts of the proof, are very similar to [KRR14], and hence are deferred to the appendix. However, the part of the proof, which consists of a soundness amplification lemma, we change completely and simplify. The new proof for this is presented in Section 4.3.

4.1 Definitions

We start with a variant of the classical definition of a PCP system. The only change is that the PCP queries Q do not depend on the instance x. As we mentioned above, known constructions of PCP have this property.

Definition 6. *A $k(\cdot)$-query PCP for an NP language L is a tuple of* PPT *algorithms (V_0, V_1, P), such that:*

- *(Completeness) For all $\lambda \in \mathbb{N}$ and $x \in L$ (with witness w) such that $|x| \le 2^\lambda$,*

$$\Pr\left[V_1(\mathsf{st}, x, \pi|_Q) = 1 \;\middle|\; \begin{array}{c} (Q, \mathsf{st}) \leftarrow V_0(1^\lambda) \\ \pi \leftarrow P(1^\lambda, x, w) \end{array} \right] = 1 \ ,$$

  The PCP proof π is a string of characters over some alphabet Γ. It will be convenient for us to view this string as indexed by a set Σ (w.l.o.g think of $\Sigma = [\ell]$ where ℓ is the length of the string), and $Q \subseteq \Sigma$. Alternatively, π can be thought of as a function from Σ to Γ.

- *(Soundness) For all $\lambda \in \mathbb{N}$, $x \notin \mathcal{L}$, $|x| \le 2^\lambda$, and all proof strings π,*

$$\Pr\left[V_1(\mathsf{st}, x, \pi|_Q) = 1 \middle| (Q, \mathsf{st}) \leftarrow V_0(1^\lambda) \right] \le \frac{1}{2}$$

- *(Query Efficiency) If $(Q, \mathsf{st}) \leftarrow V_0(1^\lambda)$, then $|Q| \le k(\lambda)$ and the combined run-time of V_0 and V_1 is* $\mathsf{poly}(\lambda)$.

- *(Prover Efficiency) The prover P runs in polynomial time, where its input is $(1^\lambda, x, w)$.*

Remark. A reader may wonder, since V_0 is not given $|x|$, how P can achieve proofs with length polynomial in $|x|$, rather than polynomial in the bound on $|x|$ that V_0 knows (which is 2^λ). In a typical PCP scheme, we assume the verifier and prover are both given a bound N such that $|x| \le N$. The prover runs in time $\mathsf{poly}(N)$ and the verifier runs in time $\mathsf{poly}(\log N)$. From such a PCP, we can achieve the above "instance-specific efficiency" generically: we just use λ schemes in parallel, where in the i^{th} scheme $N = 2^i$. When given x, a prover generates a proof for the scheme in which $N/2 < x \le N$.

[7] Formally, our theorem is incomparable to that of [KRR14], but its use of computational no-signaling seems to make it more suited to cryptographic applications.

Adaptive CNS Provers. We now consider cheating PCP provers, who are adaptive and computationally no-signaling (CNS) provers, as described above.

Definition 7 (Adaptive Adversarial PCP Prover)**.** *An adaptive adversarial PCP prover for a language L is an ensemble of* PPT *algorithms $\{P^*_\lambda\}$ which takes a finite set of queries Q and outputs strings x_0 and x_1 that it claims are in L, together with assignments A_0 and $A_1 : Q \to \Gamma$.*

Definition 8 (Adaptive Computationally No-Signaling Prover)**.** *An adaptive adversarial PCP prover is said to be $k_{max}(\cdot)$-wise computationally no-signaling if for all sets $Q'_\lambda \subseteq Q_\lambda$, with $|Q_\lambda| \leq k_{max}(\lambda)$, it holds that*

$$(x_0, x_1, A_0|_{Q'_\lambda}, A_1|_{Q'_\lambda}) \approx_c (x'_0, x'_1, A'_0, A'_1) \left| \begin{array}{l} (x_0, x_1, A_0, A_1) \leftarrow P^*_\lambda(Q_\lambda) \\ (x'_0, x'_1, A'_0, A'_1) \leftarrow P^*_\lambda(Q'_\lambda) \end{array} \right.$$

Adaptive Local Soundness. As explained above, our notion of adaptive local soundness relies on the notion of local assignment generators. A formal definition follows. From this point and on, we will focus only on proofs for 3SAT (with the natural witness relation).

Definition 9 (Adaptive Local Assignment Generator)**.** *An adaptive $(\ell(\cdot), \epsilon(\cdot))$-local assignment generator* Assign *on variables $\{V_\lambda\}_{\lambda\in\mathbb{N}}$ is an algorithm that takes as input a security parameter 1^λ and a set of at most $\ell(\lambda)$ queries $W \subseteq V_\lambda$, and outputs two 3-CNFs (φ_0, φ_1) (each on variables V_λ) and assignments $A_0 : W \to \{0, 1\}$ and $A_1 : W \to \{0, 1\}$, such that the following two properties hold.*

- **Everywhere Local Consistency.** *For every λ, every set $W \subseteq V_\lambda$ with $|W| \leq \ell(\lambda)$, with probability at least $1 - \epsilon(\lambda)$ over sampling*
$$(\varphi_0, \varphi_1, A_0, A_1) \leftarrow \mathsf{Assign}(1^\lambda, W),$$
the assignments A_0 and A_1 are respectively "locally consistent" with the formulas φ_0 and φ_1. That is, every clause in φ_b whose variables v_1, v_2, v_3 are in W is satisfied by the assignment $A_b(v_1), A_b(v_2), A_b(v_3)$.

- **Computational No-Signaling.** *For every ensemble $\{W'_\lambda, W_\lambda\}_{\lambda\in\mathbb{N}}$ of subsets $W'_\lambda \subset W_\lambda \subset V_\lambda$ with $|W_\lambda| \leq \ell(\lambda)$, we have*
$$\varphi_0, \varphi_1, A_0|_{W'_\lambda}, A_1|_{W'_\lambda} \approx_c \varphi'_0, \varphi'_1, A'_0, A'_1$$
in the probability space defined by sampling
$$(\varphi_0, \varphi_1, A_0, A_1) \leftarrow \mathsf{Assign}(1^\lambda, W_\lambda)$$
$$(\varphi'_0, \varphi'_1, A'_0, A'_1) \leftarrow \mathsf{Assign}(1^\lambda, W'_\lambda)$$

We say that Assign *is an $\ell(\cdot)$-local assignment generator if it is an $(\ell(\cdot), \mathsf{negl}(\cdot))$-local assignment generator for some negligible function* negl.

Threshold Verifiers. For the purpose of our construction and proof, we consider a variant of the common sequential repetition principle, where properties of a PCP are enhanced by running the verifier several times on the same proof string. In our case, the purpose will not be to enhance soundness but rather to prove no-signaling properties. We would like to consider a case where the verifier accepts even if not all answers are satisfying, but rather only some fraction. We will require the definition of a threshold verifier as follows.

Definition 10 (Threshold Verifier)**.** *Given a PCP verifier $V = (V_0, V_1)$, we define the t-of-n threshold verifier $(V_0^{\otimes n}, V_1^{\geq t})$ (where both n and t may be functions of λ).*

$V_0^{\otimes n}$ takes a security parameter 1^λ and does the following:

1. *Compute $(Q_i, \mathsf{st}_i) \leftarrow V_0(1^\lambda)$ for $i = 1, \ldots, n$.*

2. *Output $(\cup_{i=1}^n Q_i, (Q_1, \mathsf{st}_1), \ldots, (Q_n, \mathsf{st}_n))$.*

$V_1^{\geq t}$ takes input $(((Q_1, \mathsf{st}_1), \ldots, (Q_n, \mathsf{st}_n)), x, A)$ and does the following:

1. *Compute $y_i \leftarrow V_1(\mathsf{st}_i, x, A_{Q_i})$ for $i = 1, \ldots, n$.*
2. *Output 1 if at least t of the y_i's are 1; otherwise outputs 0.*

For the special case where $t = n$, we write $V_1^{\otimes n}$ instead of $V_1^{\geq n}$.

4.2 Existence of Adaptive PCP with Local Soundness

We show that there exists an adaptive PCP with local soundness against CNS cheating provers. Relating our result to previous work, [KRR14] showed a PCP in which: if a non-adaptive statistically no-signaling prover convinces the verifier to accept a 3-CNF φ, then there is a (non-adaptive) local assignment generator for φ. We show that the same PCP provides stronger soundness guarantees. The cheating prover is now allowed to only be *computationally* no-signaling, and can be adaptive (i.e. output the instances on which they attempt to convince the verifier). Consequently, the local assignment generator, which results from a cheating prover, is computationally no-signaling and adaptive.

Theorem 1. *There is a PCP $(\tilde{V}_0, \tilde{V}_1, \tilde{P})$ for 3-SAT satisfying the following soundness property.*

For every $c > 0$ there is a PPT oracle machine Assign_c and a polynomial ℓ_0 such that for all polynomials ℓ, for all poly-sized adversarial PCP provers $\mathcal{P}^* = \{P^*_\lambda\}_{\lambda \in \mathbb{N}}$ which are $(\ell \cdot \ell_0 + \lambda^2)$-wise CNS, if for all λ in an infinite set Λ,

$$\Pr\left[\begin{array}{l} 1 \leftarrow \tilde{V}_1(\mathsf{st}, \varphi_0, A_0) \ \wedge \\ 1 \leftarrow \tilde{V}_1(\mathsf{st}, \varphi_1, A_1) \end{array} \middle| \begin{array}{l} (Q, \mathsf{st}) \leftarrow \tilde{V}_0(1^\lambda) \\ (\varphi_0, \varphi_1, A_0, A_1) \leftarrow P^*_\lambda(Q) \end{array}\right] \geq \lambda^{-c},$$

then for all $\lambda \in \Lambda$, $\mathsf{Assign}_c^{P^*_\lambda}(1^\lambda, \cdot)$ is an adaptive ℓ-local assignment generator such that for any set of variables W, sampling

$$(\varphi_0, \varphi_1, A_0, A_1) \leftarrow \mathsf{Assign}_c^{P^*_\lambda}(1^\lambda, W)$$

produces a computationally indistinguishable distribution on (φ_0, φ_1) as sampling (φ_0, φ_1) according to the conditional distribution

$$\begin{array}{l} (Q, \mathsf{st}) \leftarrow \tilde{V}_0(1^\lambda) \\ (\varphi_0, \varphi_1, A_0, A_1) \leftarrow P^*_\lambda(Q) \end{array} \middle| \begin{array}{l} \tilde{V}_1(\mathsf{st}, \varphi_0, A_0) = 1 \ \wedge \\ \tilde{V}_1(\mathsf{st}, \varphi_1, A_1) = 1. \end{array}$$

Moreover, $\tilde{V}_1$ only uses the LDEs of φ_0 and φ_1. Given oracle access to these LDEs, $\tilde{V}_0$ and $\tilde{V}_1$'s total running time is $\mathsf{poly}(\lambda)$ for some fixed (i.e. independent of $|\varphi_0|$ or $|\varphi_1|$) polynomial poly.

Proof. As explained above, our PCP is identical to the one from [KRR14]. However, our theorem requires different guarantees than previous works and we do it in a more modular manner and prove a new amplification theorem on CNS provers along the way. We start with a "base PCP" which is formally described in Appendix A. We denote this PCP by (V_0, V_1, P), and we let k (=polylog(λ)) denote the number of queries made by V_0. We define our PCP $(\tilde{V}_0, \tilde{V}_1, \tilde{P})$ to be $(V_0^{\otimes \lambda}, V_1^{\otimes \lambda}, P)$, i.e. the λ-repeated (λ-out-of-λ) version of (V_0, V_1, P), as per Definition 10. Completeness holds trivially. We thus focus on proving the adaptive computational no-signaling local soundness property.

To this end, fix any ℓ, and suppose there exists an $(\ell \cdot \ell_0 + \lambda^2)$-wise computationally no-signaling cheating prover, Prover^*, (where ℓ_0 is a polynomial that will be determined later), and there exists a constant $c \in \mathbb{N}$ such that for infinitely many λ's,

$$\Pr\left[\begin{array}{l} (Q, \mathsf{st}) \leftarrow V_0^{\otimes \lambda}(1^\lambda); \\ (\varphi_1, \varphi_2, A_1, A_2) \leftarrow \mathsf{Prover}^*(Q); \\ 1 \leftarrow \left(V_1^{\otimes \lambda}\right)^{\hat{\phi}_1, \hat{\phi}_2}(\mathsf{st}, A_1, A_2) \end{array}\right] \geq \frac{1}{\lambda^c}.$$

We show that by lowering the threshold for verification to a small yet super-logarithmic value, the success probability of a related cheating prover increases to near perfect. In particular, Lemma 1, which is stated and proven in

Section 4.3 below, implies that there exists a $(\ell \cdot \ell_0 + \lambda^2 - \lambda \cdot k)$-wise computationally no-signaling cheating prover, which will be denoted by Prover^{**}, and there exists a negligible function ϵ such that for infinitely many λ's,

$$\Pr\left[\begin{array}{l}(Q, \mathsf{st}) \leftarrow V_0^{\otimes\lambda}(1^\lambda);\\ (\varphi'_1, \varphi'_2, A_1, A_2) \leftarrow \mathsf{Prover}^{**}(Q);\\ 1 \leftarrow \left(V_1^{\geq\lambda-r}\right)^{\hat{\phi}'_1, \hat{\phi}'_2}(\mathsf{st}, A_1, A_2)\end{array}\right] \geq 1 - \epsilon,$$

for any $r = \omega(\log \lambda)$ (for concreteness, the reader can think of $r = \sqrt{\lambda}$). Moreover, the distribution of (φ'_1, φ'_2) is indistinguishable from the distribution of $(\varphi_1, \varphi_2)|(V \text{ accepts})$. Note that since $\lambda \geq k$,

$$\ell \cdot \ell_0 + \lambda^2 - \lambda \cdot k \geq \ell \cdot \ell_0.$$

In what follows we let $k_{max} = \ell \cdot \ell_0$.

The remainder of the proof is showing how to construct an assignment generator out of the amplified cheating prover. This is done using methods that are very similar to those of [KRR14]. In particular, Lemma 6 in Appendix B implies that there exists a probabilistic polynomial-time oracle machine Assign such that $\mathsf{Assign}^{\mathsf{Prover}^{**}}$ is an adaptive (ℓ, ϵ')-local assignment generator with $\epsilon' = \epsilon \cdot \mathsf{poly}(\lambda) + \mathsf{negl}(\lambda) = \mathsf{negl}(\lambda)$, which completes the proof of the theorem. □

4.3 New Soundness Amplification for No-Signaling PCPs

In the proof of Theorem 1, we use a soundness amplification lemma, which uses the notion of PCPs with "t-of-n threshold" verifiers.

Roughly speaking, in our soundness amplification lemma below, we show that if an adaptive CNS prover convinces a λ-of-λ threshold verifier with any non-negligible probability, then there is an adaptive CNS prover which convinces a corresponding $(\lambda - \omega(\log \lambda))$-of-$\lambda$ verifier with high probability. Further, the second prover produces the same distribution of (φ_0, φ_1) as the first, conditioned on the first convincing the λ-of-λ verifier.

Lemma 1 (Soundness Amplification). *For all $k(\cdot)$-query PCPs (V_0, V_1, P) for 3SAT and all $c > 0$, there is a* PPT *oracle algorithm* $\mathsf{Amplify}_c$ *such that* if *there is an adaptive k_{max}-wise CNS adversarial prover $\{P^*_\lambda\}_{\lambda\in\mathbb{N}}$ such that*

$$\Pr\left[\begin{array}{l}1 \leftarrow V_1^{\otimes\lambda}(\mathsf{st}, \varphi_0, A_0)\ \wedge\\ 1 \leftarrow V_1^{\otimes\lambda}(\mathsf{st}, \varphi_1, A_1)\end{array}\middle|\begin{array}{l}(Q, \mathsf{st}) \leftarrow V_0^{\otimes\lambda}(1^\lambda)\\ (\varphi_0, \varphi_1, A_0, A_1) \leftarrow P^*_\lambda(Q)\end{array}\right] \geq \lambda^{-c}$$

for infinitely many λ, then $\{\mathsf{Amplify}_c^{P^*_\lambda}(1^\lambda, \cdot)\}_{\lambda\in\mathbb{N}}$ is an adaptive $(k_{max} - \lambda \cdot k)$-wise adversarial prover ensemble and for any $r = \omega(\log \lambda)$, there is a negligible function negl *such that for infinitely many λ,*

$$\Pr\left[\begin{array}{l}1 \leftarrow V_1^{\geq\lambda-r}(\mathsf{st}, \varphi_0, A_0)\ \wedge\\ 1 \leftarrow V_1^{\geq\lambda-r}(\mathsf{st}, \varphi_1, A_1)\end{array}\middle|\begin{array}{l}(Q, \mathsf{st}) \leftarrow V_0^{\otimes\lambda}(1^\lambda)\\ (\varphi_0, \varphi_1, A_0, A_1) \leftarrow \mathsf{Amplify}_c^{P^*_\lambda}(1^\lambda, Q)\end{array}\right] \geq 1 - \mathsf{negl}(\lambda).$$

Furthermore, the distributions on (φ_0, φ_1) obtained by sampling the conditional distribution

$$\begin{array}{l}(Q, \mathsf{st}) \leftarrow V_0^{\otimes\lambda}(1^\lambda)\\ (\varphi_0, \varphi_1, A_0, A_1) \leftarrow P^*_\lambda(Q)\end{array}\ \middle|\ \begin{array}{l}1 \leftarrow V_1^{\otimes\lambda}(\mathsf{st}, \varphi_0, A_0)\ \wedge\\ 1 \leftarrow V_1^{\otimes\lambda}(\mathsf{st}, \varphi_1, A_1)\end{array}$$

and

$$\begin{array}{l}(Q, \mathsf{st}) \leftarrow V_0^{\otimes\lambda}(1^\lambda)\\ (\varphi_0, \varphi_1, A_0, A_1) \leftarrow \mathsf{Amplify}_c^{P^*_\lambda}(1^\lambda, Q)\end{array}$$

are computationally indistinguishable.

Proof. By assumption, there exists a constant $c > 0$ such that for infinitely many λ, P^*_λ convinces V with probability at least λ^{-c}. Let Λ denote this set of λ.

We describe the algorithm $\mathsf{Amplify}_c$. On an input set of queries $Q \in \Sigma^{k_{max} - \lambda \cdot k}$, $\mathsf{Amplify}_c$ independently samples $(Q_i, \mathsf{st}_i) \leftarrow V_0^{\otimes \lambda}(1^\lambda)$ and $(\varphi_i^0, \varphi_i^1, A_i^0, A_i^1) \leftarrow P_\lambda^*(Q \cup Q_i)$ for $i = 1, \ldots, \lambda^{c+1}$.

$\mathsf{Amplify}_c$ then outputs $(\varphi_{i^*}^0, \varphi_{i^*}^1, (A_{i^*}^0)_Q, (A_{i^*}^1)_Q)$ for the first i^* such that

$$V_1^{\otimes \lambda}(\mathsf{st}_{i^*}, \varphi_{i^*}^0, (A_{i^*}^0)_{Q_{i^*}}) = 1 \wedge V_1^{\otimes \lambda}(\mathsf{st}_{i^*}, \varphi_{i^*}^1, (A_{i^*}^1)_{Q_{i^*}}) = 1.$$

Call such an i^* "good". If no good i^* exists, then $\mathsf{Amplify}_c$ outputs $\perp$ (in Claim 2 below, we show that for $\lambda \in \Lambda$, this happens with negligible probability).

Let us first show the "furthermore" of the lemma.

Claim 1. *The distribution on (φ_0, φ_1) obtained by sampling the conditional distribution*

$$\left.\begin{array}{l} (Q, \mathsf{st}) \leftarrow V_0^{\otimes \lambda}(1^\lambda) \\ (\varphi_0, \varphi_1, A_0, A_1) \leftarrow P_\lambda^*(Q) \end{array}\right| \begin{array}{l} 1 \leftarrow V_1^{\otimes \lambda}(\mathsf{st}, \varphi_0, A_0) \ \wedge \\ 1 \leftarrow V_1^{\otimes \lambda}(\mathsf{st}, \varphi_1, A_1) \end{array} \tag{1}$$

and the distribution on (φ_0, φ_1) obtained by sampling

$$\begin{array}{l} (Q, \mathsf{st}) \leftarrow V_0^{\otimes \lambda}(1^\lambda) \\ (\varphi_0, \varphi_1, A_0, A_1) \leftarrow \mathsf{Amplify}_c^{P_\lambda^*}(1^\lambda, Q) \end{array} \tag{2}$$

are computationally indistinguishable.

Proof. Recall that the definition of computational no-signaling says that for all $Q'_\lambda \subseteq Q_\lambda$, such that $|Q_\lambda| \leq k_{max}$, the distributions

$$(\varphi_0, \varphi_1, (A_0)|_{Q'}, (A_1)|_{Q'_\lambda})$$

and

$$(\varphi'_0, \varphi'_1, A'_0, A'_1)$$

are computationally indistinguishable, when sampling

$$\begin{array}{r} (\varphi'_0, \varphi'_1, A'_0, A'_1) \leftarrow P_\lambda^*(Q'_\lambda) \\ (\varphi_0, \varphi_1, A_0, A_1) \leftarrow P_\lambda^*(Q_\lambda). \end{array}$$

This is equivalent to the following, seemingly stronger statement:

Corollary 1. *For all poly-sized auxiliary information aux_λ and sets $Q'_\lambda \subseteq Q_\lambda$, such that $|Q_\lambda| \leq k_{max}$, the distributions*

$$(\mathsf{aux}_\lambda, \varphi_0, \varphi_1, (A_0)|_{Q'}, (A_1)|_{Q'_\lambda})$$

and

$$(\mathsf{aux}_\lambda, \varphi'_0, \varphi'_1, A'_0, A'_1)$$

are computationally indistinguishable, when sampling

$$\begin{array}{r} (\varphi'_0, \varphi'_1, A'_0, A'_1) \leftarrow P_\lambda^*(Q'_\lambda) \\ (\varphi_0, \varphi_1, A_0, A_1) \leftarrow P_\lambda^*(Q_\lambda). \end{array}$$

Consider $\mathsf{Amplify}'_c$, a modified version of $\mathsf{Amplify}_c$ which samples each $(\varphi_i^0, \varphi_i^1, A_i^0, A_i^1)$ from $P_\lambda^*(Q_i)$ instead of from $P_\lambda^*(Q \cup Q_i)$. But, (φ_0, φ_1) are distributed identically in (1) as they are in the conditional probability space

$$\left.\begin{array}{l} (Q, \mathsf{st}) \leftarrow V_0^{\otimes \lambda}(1^\lambda) \\ (\varphi_0, \varphi_1, A_0, A_1) \leftarrow \mathsf{Amplify}'^{P_\lambda^*}_c(1^\lambda, Q) \end{array}\right| \text{a good } i^* \text{ exists}$$

We later show (as part of Claim 2) that a good i^* exists with overwhelming probability, this distribution on (φ_0, φ_1) is statistically close to that obtained by sampling

$$\begin{array}{l} (Q, \mathsf{st}) \leftarrow V_0^{\otimes \lambda}(1^\lambda) \\ (\varphi_0, \varphi_1, A_0, A_1) \leftarrow \mathsf{Amplify}'^{P_\lambda^*}_c(1^\lambda, Q) \end{array} \tag{3}$$

without any condition.

We claim that this distribution on (φ_0, φ_1) is computationally indistinguishable from that obtained by sampling

$$\begin{aligned}&(Q, \mathsf{st}) \leftarrow V_0^{\otimes\lambda}(1^\lambda)\\&(\varphi_0, \varphi_1, A_0, A_1) \leftarrow \mathsf{Amplify}_c{}^{P^*_\lambda}(1^\lambda, Q)\end{aligned} \tag{4}$$

It suffices for us to show that the distribution on

$$(\mathsf{st}_i, \varphi_i^0, \varphi_i^1, (A_i^0)|_{Q_i}, (A_i^1)|_{Q_i})$$

obtained when sampling (3) is computationally indistinguishable from that obtained when sampling (4), because these are all the intermediate values needed to compute the first good i^* and hence the resultant (φ_0, φ_1). But this indistinguishability follows from Corollary 1, where st_i is the auxiliary information. □

Claim 2. *For* $\lambda \in \Lambda$*,*

$$\Pr\left[\begin{array}{l}1 \leftarrow V_1^{\geq\lambda-r}(\mathsf{st}, \varphi_0, A_0) \ \wedge\\ 1 \leftarrow V_1^{\geq\lambda-r}(\mathsf{st}, \varphi_1, A_1)\end{array}\middle|\begin{array}{l}(Q, \mathsf{st}) \leftarrow V_0^{\otimes\lambda}(1^\lambda)\\ (\varphi_0, \varphi_1, A_0, A_1) \leftarrow \mathsf{Amplify}_c^{P^*_\lambda}(1^\lambda, Q)\end{array}\right] \geq 1 - \mathsf{negl}(\lambda).$$

Proof. The probability that the verifier isn't convinced is bounded by the probability that there is no good i^* plus the probability that $V_1^{\geq\lambda-r}(\mathsf{st}_{i^*}, \varphi_{i^*}^0, (A_{i^*}^0)_Q) = 0$ or $V_1^{\geq\lambda-r}(\mathsf{st}_{i^*}, \varphi_{i^*}^1, (A_{i^*}^1)_Q) = 0$ for a good i^*. We show that both of these probabilities are negligible.

First, because P^*_λ convinces $(V_0^{\otimes\lambda}, V_1^\lambda)$ with probability λ^{-c} and because $\mathcal{P}^*$ is CNS, the probability that no good i^* exists is bounded by $(1 - \lambda^{-c} + \mathsf{negl}(\lambda))^{\lambda^{c+1}}$, which is negligible.

The probability that $V_1^{\geq\lambda-r}(\mathsf{st}_{i^*}, \varphi_{i^*}^b, (A_{i^*}^b)_Q) = 0$ for a good i^* is equal to the conditional probability

$$\Pr\left[V_1^{\geq\lambda-r}(\mathsf{st}, \varphi_{i^*}^b, (A_{i^*}^b)_Q) = 0 \middle| \begin{array}{l}V_1^{\otimes\lambda}(\mathsf{st}_{i^*}, \varphi_{i^*}^0, (A_{i^*}^0)_{Q_{i^*}}) = 1 \ \wedge\\ V_1^{\otimes\lambda}(\mathsf{st}_{i^*}, \varphi_{i^*}^1, (A_{i^*}^1)_{Q_{i^*}}) = 1\end{array}\right] \tag{5}$$

in the probability space defined by sampling $(Q, \mathsf{st}) \leftarrow V_0^{\otimes\lambda}(1^\lambda)$ and $(Q_{i^*}, \mathsf{st}_{i^*}) \leftarrow V_0^{\otimes\lambda}(1^\lambda)$ and $(\varphi_{i^*}^0, \varphi_{i^*}^1, A_{i^*}^0, A_{i^*}^1) \leftarrow P^*_\lambda(Q \cup Q_{i^*})$.

This is negligible because:

- P^*_λ convinces with probability λ^{-c} and is k_{max}-CNS, so

  $$\Pr\left[\begin{array}{l}V_1^{\otimes\lambda}(\mathsf{st}_{i^*}, \varphi_{i^*}^0, (A_{i^*}^0)_{Q_{i^*}}) = 1 \ \wedge\\ V_1^{\otimes\lambda}(\mathsf{st}_{i^*}, \varphi_{i^*}^1, (A_{i^*}^1)_{Q_{i^*}}) = 1\end{array}\right] \geq \lambda^{-c} - \mathsf{negl}(\lambda),$$

  which is non-negligible; and

- $$\Pr\left[\left(\begin{array}{l}V_1^{\geq\lambda-r}(\mathsf{st}, \varphi_{i^*}^0, (A_{i^*}^0)_Q) = 0 \ \vee\\ V_1^{\geq\lambda-r}(\mathsf{st}, \varphi_{i^*}^1, (A_{i^*}^1)_Q) = 0\end{array}\right) \wedge \left(\begin{array}{l}V_1^{\otimes\lambda}(\mathsf{st}_{i^*}, \varphi_{i^*}^0, (A_{i^*}^0)_{Q_{i^*}}) = 1 \ \wedge\\ V_1^{\otimes\lambda}(\mathsf{st}_{i^*}, \varphi_{i^*}^1, (A_{i^*}^1)_{Q_{i^*}}) = 1\end{array}\right)\right] \leq \mathsf{negl}(\lambda),$$

  which holds because P^*_λ does not learn Q or Q_{i^*}; only $Q \cup Q_{i^*}$. Let us fix $Q \cup Q_{i^*}$ (but importantly not Q or Q_{i^*} individually), on P^*_λ's responses, and on the randomness used by each instance of V_1. Suppose that a fraction p of the answers given by P^*_λ are accepted by both of the corresponding calls to V_1. Now

  $$\Pr_{Q, Q_{i^*}}\left[\begin{array}{l}V_1^{\otimes\lambda}(\mathsf{st}_{i^*}, \varphi_{i^*}^0, (A_{i^*}^0)_{Q_{i^*}}) = 1 \ \wedge\\ V_1^{\otimes\lambda}(\mathsf{st}_{i^*}, \varphi_{i^*}^1, (A_{i^*}^1)_{Q_{i^*}}) = 1\end{array}\right] \leq p^\lambda,$$

  and by Hoeffding's inequality[8]and a union bound,

  $$\Pr_{Q, Q_{i^*}}\left[\begin{array}{l}V_1^{\geq\lambda-r}(\mathsf{st}, \varphi_{i^*}^0, (A_{i^*}^0)_Q) = 0 \ \vee\\ V_1^{\geq\lambda-r}(\mathsf{st}, \varphi_{i^*}^1, (A_{i^*}^1)_Q) = 0\end{array}\right] \leq e^{-2(\lambda-r-p\lambda)^2}$$

We claim that one of these bounds is negligible in λ. If $p < 1 - \frac{r}{2\lambda}$, then $p^\lambda < e^{-\omega(\log \lambda)}$. Otherwise, $\lambda - r - p\lambda \leq -r/2 = -\omega(\log \lambda)$, and so $e^{-2(\lambda - r - p\lambda)^2} \leq e^{-\omega(\log \lambda)}$. These two cases cover (non-disjointly) all possible p.

So the total probability that $V_1^{\geq \lambda - r}$ is not convinced for a good i^* is negligible. By the definition of conditional probability, $\Pr[A|B] = \Pr[A \wedge B] / \Pr[B]$. In particular, we have shown that the probability in Equation 5 is $\mathsf{negl}(\lambda) \cdot \lambda^{c'}$, which is negligible. □

Claim 3. *$\{\mathsf{Amplify}_c^{P^*_\lambda}(1^\lambda, \cdot)\}_{\lambda \in \mathbb{N}}$ is $(k_{max} - \lambda \cdot k)$-wise CNS.*

Proof. We need to show that for all sets S_λ and Q_λ with $S_\lambda \subset Q_\lambda$ and $|Q_\lambda| \leq k_{max} - \lambda \cdot k$, it holds that

$$\mathsf{Amplify}_c^{P^*_\lambda}(1^\lambda, Q_\lambda)_{S_\lambda} \approx \mathsf{Amplify}_c^{P^*_\lambda}(1^\lambda, S_\lambda).$$

We will show this by defining indistinguishable distributions B_λ and C_λ, and giving an efficiently computable randomized function f such that as distributions,

$$\mathsf{Amplify}_c^{P^*_\lambda}(1^\lambda, Q_\lambda)_{S_\lambda} \equiv f(B_\lambda) \tag{6}$$

and

$$\mathsf{Amplify}_c^{P^*_\lambda}(1^\lambda, S_\lambda) \equiv f(C_\lambda). \tag{7}$$

Here, $\equiv$ denotes equality of distributions.

Define B_λ as the distribution on $\{(i, \mathsf{st}_i, Q_i, \varphi_i^0, \varphi_i^1, (A_i^0)_{S_\lambda \cup Q_i}, (A_i^1)_{S_\lambda \cup Q_i})\}_{i=1}^{\lambda^{c+1}}$ in the probability space defined by sampling, for each i,

$$\begin{aligned} &(\mathsf{st}_i, Q_i) \leftarrow V_0^{\otimes \lambda}(1^\lambda) \\ &(\varphi_i^0, \varphi_i^1, A_i^0, A_i^1) \leftarrow P^*_\lambda(Q_\lambda \cup Q_i) \end{aligned}$$

Define C_λ as the distribution on $\{(i, \mathsf{st}_i, Q_i, \varphi_i^0, \varphi_i^1, A_i^0, A_i^1)\}_{i=1}^{\lambda^{c+1}}$ in the probability space defined by sampling, for each i,

$$\begin{aligned} &(\mathsf{st}_i, Q_i) \leftarrow V_0^{\otimes \lambda}(1^\lambda) \\ &(\varphi_i^0, \varphi_i^1, A_i^0, A_i^1) \leftarrow P^*_\lambda(S_\lambda \cup Q_i) \end{aligned}$$

Then, f is defined to find the first "good" i^*, and output $(\varphi_{i^*}^0, \varphi_{i^*}^1, (A_{i^*}^0)_{S_\lambda}, (A_{i^*}^1)_{S_\lambda})$. Equations (6) and (7) are then easy to verify.

It remains to show that $B_\lambda \approx C_\lambda$. This follows by a simple hybrid argument over the different i's. Indeed, we just need that for each i,

$$P^*_\lambda(Q_\lambda \cup Q_i)_{S_\lambda \cup Q_i} \approx P^*_\lambda(S_\lambda \cup Q_i),$$

which is guaranteed by the fact that $\mathcal{P}^*$ is k_{max}-wise CNS. □

□

[8] Hoeffding's inequality[Hoe63] states that if $X_1, \ldots, X_n$ are independent Bernoulli distributions with parameter p, and H denotes $\sum_i X_i$, then

$$\Pr[H \geq (p + \epsilon)n] \leq e^{-2\epsilon^2 n}.$$

Hoeffding also showed that this inequality holds even if $X_1, \ldots, X_n$ are not independent, but are chosen uniformly without replacement from a population of $\{0, 1\}$'s, a fraction p of which are 1's.

5 Adaptive RAM Delegation

In this section, we use Theorem 1 to construct a RAM delegation scheme. Our construction is essentially the same as in [KP15]: To prove that $M^{D \to D_{\mathsf{new}}} \to y$ for $\mathsf{d}_{\mathsf{new}} = \mathsf{Digest}(D_{\mathsf{new}})$ and $\textsc{Time}(M, D) \leq T$, we use the PCP to prove the local satisfiability of a related 3-CNF $\varphi = \varphi_{M,\mathsf{d},y,\mathsf{d}_{\mathsf{new}},T}$ (where $\mathsf{d} = \mathsf{Digest}(D)$). The formula φ is defined in Section 5.1. It has $O(T \cdot \mathsf{poly}(\lambda))$ variables and verifies a transcript of M using a CRHF $h : \{0,1\}^{2\lambda} \to \{0,1\}^{\lambda}$. If $M^D \not\to y$, then a collision in h can be found from M, D, and any assignment to φ. In fact, even a *local* assignment generator for φ suffices, and this is the property that was used in [KP15].

First, we show how to make the verifier in this scheme run in fixed polynomial time, independent of $\textsc{Time}(M, D)$. To do this, we show how to take advantage of the repetitive and local structure of φ to efficiently compute its clause indicator function's low-degree extension $\hat{\phi}$ (see Claim 3). In [KRR14], the verifier delegates the computation of $\hat{\phi}$ using the protocol of [GKR15]. We note that this is not straight-forward: [KRR14] proved the soundness of this composition against *statistically* no-signaling provers, but the proof does not apply to provers which are only computationally no-signaling. We consider the verifier's direct computation of $\hat{\phi}$ to be a simplification as well as an optimization to the overall protocol.

Finally, we show adaptive soundness. Here we grapple with the fact that even if the PCP verifier accepts a 3-CNF φ given by an adaptive cheating prover P^*, there is not necessarily a local assignment generator for φ. Rather, there is an adaptive local assignment generator Assign which outputs $\varphi_{M,\mathsf{d},y,\mathsf{d}_{\mathsf{new}},T}$ for some distribution on $(M, \mathsf{d} = \mathsf{Digest}(D), y, \mathsf{d}_{\mathsf{new}})$. While this does not allow us to reconstruct the full execution transcript of M^D for any single (M, D), we are still able to find a fixed set of variables W such that by querying $\mathsf{Assign}(W)$, we observe a false hash tree proof (and hence obtain a hash collision) with non-negligible probability.

As mentioned in Section 3, we actually achieve a stronger notion of soundness: no prover can prove two different "correct" $(y, \mathsf{d}_{\mathsf{new}})$ outputs for any adversarially chosen digest d and machine M (in particular, there is not necessarily a D such that $\mathsf{d} = \mathsf{Digest}(D)$). This is stronger because if d were equal to $\mathsf{Digest}(D)$, then completeness guarantees we can prove the correct result, and hence cannot prove any incorrect result.

5.1 RAMs as 3-CNF Formulas

[KP15] gives a 3-CNF that succinctly verifies a RAM machine's execution transcript (under computational assumptions) by using Merkle trees. The following lemma describes its essential properties.

Lemma 2 ([KP15])**.** *There exist deterministic polynomial-time algorithms*

$$\mathsf{MakeCNF}, \mathsf{ProcessDB}, \mathsf{Transcript}, \mathsf{FindCollision},$$

such that for any hash function $h : \{0,1\}^{2\lambda} \to \{0,1\}^{\lambda}$, RAM machine M, digest d, output y, digest d', and time bound T, $\mathsf{MakeCNF}(h, M, \mathsf{d}, y, \mathsf{d}', T)$ is a 3-CNF $\varphi \triangleq \varphi_{in}^{\mathsf{d}} \wedge \varphi^{M,T,h} \wedge \varphi_{out}^{y,\mathsf{d}'}$ with the following structure:

1. *$\varphi^{M,T,h}$ has $T' = T \cdot |M| \cdot \mathsf{poly}(\lambda)$ variables. The set of variables V is partitioned into $T + 1$ disjoint sets $V_0, \ldots, V_T$ with identical sets of clauses in $\varphi^{M,T,h}$ between adjacent layers. Each layer V_i contains (among other things):*
   - *Variables $V_i.q$ whose values encode the RAM machine's local state after i execution steps.*
   - *Variables $V_i.\mathsf{d}$ whose values encode a digest of the database contents after i execution steps.*
   - *Variables $V_i.y$ whose values encode the output of M if it has terminated by the i^{th} step, and encode $\perp$ otherwise.*
2. *$\varphi_{in}^{M,\mathsf{d}}$ has clauses only on V_0, which enforce that $V_0.q = M.q_0$, $V_0.\mathsf{d} = \mathsf{d}$, and $V_0.y = \perp$. $\varphi_{out}^{y,\mathsf{d}'}$ has clauses only on V_T, which enforce that $V_T.y = y$ and $V_T.\mathsf{d} = \mathsf{d}'$.*
3. *If $M^{D \to D_{\mathsf{new}}} \to y$ with $(\mathsf{d}, \mathsf{dt}) = \mathsf{ProcessDB}(h, D)$ and $\mathsf{d}' = \mathsf{Digest}(h, D_{\mathsf{new}})$ and $\textsc{Time}(M^D) \leq T$, then $\mathsf{Transcript}^{\mathsf{dt}}(1^T, M)$ outputs a satisfying assignment A for φ in time $T \cdot |M| \cdot \mathsf{poly}(\lambda)$ such that $A(V_0.\mathsf{d}) = \mathsf{d}$ and $A(V_T.\mathsf{d}) = \mathsf{d}'$.*

4. *If two locally-consistent assignments A and A' for φ, both on $V_i \cup V_{i+1}$, agree on $V_i.q \cup V_i.\mathsf{d} \cup V_i.y$ but disagree on $V_{i+1}.q \cup V_{i+1}.\mathsf{d} \cup V_{i+1}.y$, then* $\mathsf{FindCollision}(1^T, \varphi, A, A')$ *outputs (x, x') such that $x \neq x$ and $h(x) = h(x')$ – in other words, a collision under h.*

Now fix $M, \mathsf{d}, T, h, y, \mathsf{d}'$, and let φ and T' be as above. Let $\phi : V^3 \times \{0,1\}^3$ denote the "clause indicator function" of φ. That is,

$$\phi(v_1, v_2, v_3, b_1, b_2, b_3) = \begin{cases} 1 & \text{if the clause } (v_1 = b_1 \vee v_2 = b_2 \vee v_3 = b_3) \text{ is in } \varphi \\ 0 & \text{otherwise} \end{cases}$$

Let $\mathbb{F}$ be any finite field with a subset H of size $\log T'$, and let m be $\frac{\log T'}{\log \log T'}$. Given any injective mapping $e : V \hookrightarrow H^m$, one can view ϕ as a function mapping $H^{3m+3} \to \{0,1\}$.

$$\phi_e(i_1, i_2, i_3, b_1, b_2, b_3) = \begin{cases} 1 & \text{if the clause } (e^{-1}(i_1) = b_1 \vee e^{-1}(i_2) = b_2 \vee e^{-1}(i_3) = b_3) \text{ is in } \varphi \\ 0 & \text{otherwise (including if } b_j \notin \{0,1\} \text{ or } i_j \notin \mathsf{Img}(e)) \end{cases}$$

Using property 1 of Lemma 2 above, we prove the following lemma, assuming that operations on $\mathbb{F}$ have unit cost.

Lemma 3. *There is an efficiently computable and invertible mapping $e : V \hookrightarrow H^m$ such that the low-degree extension*

$$\hat{\phi} : \mathbb{F}^{3m+3} \to \mathbb{F}$$

of

$$\phi : H^{3m+3} \to \{0,1\}$$

is efficiently computable at any point $z \in \mathbb{F}^{3m+3}$ in time $\mathsf{poly}(|M|, \lambda)$.

Proof. We first note that it suffices to efficiently compute the low-degree extension of $\phi^{M,T,h}$, because $\hat{\phi} = \hat{\phi}^{\mathsf{d}}_{in} + \hat{\phi}^{M,T,h} + \hat{\phi}^{y,\mathsf{d}'}_{out}$, and $\varphi^{\mathsf{d}}_{in}$ and $\varphi^{y,\mathsf{d}'}_{out}$ are on a small ($\mathsf{poly}(\lambda)$) number of variables W, so their LDEs can be computed in time $\tilde{O}(W^3) = \mathsf{poly}(\lambda)$.

We next show a few basic functions whose low-degree extensions can be efficiently computed.

Claim 4. *For any $h^* \in H$, there is a univariate polynomial χ_{h^*} over $\mathbb{F}$ with degree at most $|H| - 1$ such that for all $h \in H$,*

$$\chi_{h^*}(h) = \begin{cases} 1 & \text{if } h = h^* \\ 0 & \text{otherwise} \end{cases}$$

and $\chi_{h^*}(x)$ for $x \in \mathbb{F}$ can be evaluated in $O(|H|)$ time.

Proof. This follows from Lagrange interpolation. □

Claim 5. *For all $k > 0$ and $n > 1$, there is an nk-variate polynomial δ over $\mathbb{F}$ with degree at most $|H| - 1$ in each variable such that for $h_1, \ldots, h_n \in H^k$,*

$$\delta(h_1, \ldots, h_n) = \begin{cases} 1 & \text{if } h_1 = \ldots = h_n \\ 0 & \text{otherwise.} \end{cases}$$

and δ can be evaluated in time $O(nk \cdot |H|^2)$

Proof. We observe that the h_i's are all equal if and only if for all coordinates $j \in [k]$, there is some value x_j such that for all i, $(h_i)_j = x_j$. Furthermore, there can be at most one such x_j.

Thus, for all $h_1, \ldots, h_n \in H^k$,

$$\delta(h_1, \ldots, h_n) = \prod_{j=1}^{k} \sum_{x_j \in H} \prod_{i=1}^{n} \chi_{x_j}((h_i)_j).$$

□

Claim 6. *When H^k is identified with $[|H|^k]$ using a lexicographic ordering, there is a $3k$-variate polynomial s of degree at most $|H|-1$ in each variable such that for every $x, y, z \in H^k$,*

$$s(x,y,z) = \begin{cases} 1 & \textit{if } x = y \wedge z = x+1 \\ 0 & \textit{otherwise.} \end{cases}$$

and s is computable in time $O(k^3 \cdot |H|^3)$.

Proof. We observe that when represented as a string in base $|H|$, $x+1$ has the same digits as x up to some index i, after which $x+1$ is $h+1\|0\cdots\|0$, while x is $h\||H-1|\|\cdots\||H-1|$. As an illustrative example, the representation of 79 in binary is 101111, while 80 is 110000.

Thus,

$$s(x,y,z) = \sum_{h=0}^{|H|-2} \sum_{i=1}^{k} \left(\prod_{j=1}^{i-1} \delta(x_j, y_j, z_j) \right) \cdot \chi_h(x_i)\chi_h(y_i)\chi_{h+1}(z_i) \cdot \left(\prod_{j=i+1}^{k} \chi_0(z_j)\chi_{|H|-1}(x_j)\chi_{|H|-1}(y_j) \right).$$

□

We now have the necessary tools to prove Lemma 3. Let $k = \lceil \log_{|H|} T \rceil$, and define $e : V \hookrightarrow H^k \times H^{m-k}$ so that the j^{th} variable in V_i maps to (i,j), with lexicographic correspondences between $[T]$ and H^k and between $[|V_i|]$ and H^{m-k}. We will let W denote the size of each layer. Note $|W| = \mathsf{poly}(M, \lambda)$.

Now, recall that the clauses in φ are repeated and identical between adjacent layers. We consider separately the clauses which are entirely contained within a layer, and the clauses which have variables in both layers. We can assume without loss of generality that the latter type of clauses have one variable in layer $i+1$ and two variables in layer i.

These clauses can be described by functions

$$\psi_A : [W]^3 \times \{0,1\}^3 \to \{0,1\}$$

and

$$\psi_B : [W]^3 \times \{0,1\}^3 \to \{0,1\}$$

such that φ contains the clause $(v_{t,w_1} = b_1 \vee v_{t,w_2} = b_2 \vee v_{t,w_3} = b_3)$ iff $\psi_A(w_1, w_2, w_3, b_1, b_2, b_3) = 1$ and contains the clause $(v_{t,w_1} = b_1 \vee v_{t,w_2} = b_2 \vee v_{t+1,w_3} = b_3)$ iff $\psi_B(w_1, w_2, w_3, b_1, b_2, b_3) = 1$.

Now, we claim that $\hat{\phi}$ can be evaluated efficiently in terms of the low-degree extensions $\hat{\psi}_A$ and $\hat{\psi}_B$, which themselves take only $\tilde{O}(W) = \mathsf{poly}(\lambda)$ time. This follows from the following formula for $\hat{\phi}$, where each $i_j \in \mathbb{F}^m$ is parsed as a tuple $(t_j, w_j) \in \mathbb{F}^k \times \mathbb{F}^{m-k}$.

$$\begin{aligned} \hat{\phi}(i_1, i_2, i_3, b_1, b_2, b_3) = \delta(t_1, t_2, t_3)\hat{\psi}_A(w_1, w_2, w_3, b_1, b_2, b_3) \\ + s(t_1, t_2, t_3)\hat{\psi}_B(w_1, w_2, w_3, b_1, b_2, b_3) \end{aligned}$$

□

We will next combine this lemma with Theorem 1 to obtain an adaptive RAM delegation protocol.

5.2 RAM Delegation Protocol

First, we observe that without loss of generality we can assume a fixed polynomial poly such that the RAM machines delegated are of size at most $\mathsf{poly}(\lambda)$. Otherwise, we can instead delegate a universal RAM machine, with a preceding sequence of delegated constant-sized computations that add the desired machine to the persistent database. For machines of this size, let $\ell(\lambda) = \max_i\{|V_i|\}$, where $\{V_i\}$ are the variables of the CNF given by Lemma 2. Let $(V_0, V_1, P_{\mathsf{PCP}})$ denote the PCP of Theorem 1 and let $k_{max}(\lambda) = 2\ell(\lambda)\ell_0(\lambda) + \lambda^2$, where $\ell_0(\cdot)$ is defined in the statement of Theorem 1. We will use a succinct PIR scheme $(\mathsf{ScPIR.Send}, \mathsf{ScPIR.Respond}, \mathsf{ScPIR.Decode})$ as defined in Definition 4.

Construction 1. *The algorithms* $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{ProcessDB}, \mathsf{Prove}, \mathsf{Verify})$ *are defined as below.*

Setup $\mathsf{Setup}(1^\lambda)$ *samples a collision-resistant hash function* $h : \{0,1\}^{2\lambda} \to \{0,1\}^\lambda$ *and outputs* $\mathsf{pp} = h$.

Key Generation $\mathsf{KeyGen}(1^\lambda)$ *samples*

$$(Q, \mathsf{st}) \leftarrow V_0(1^\lambda),$$

and a random injection

$$\zeta : Q \hookrightarrow [k_{max}].$$

It then defines, for each $i \in [k_{max}]$,

$$(\tilde{q}_i, s_i) = \begin{cases} \mathsf{ScPIR.Send}(1^\lambda, q) & \textit{if } \zeta(q) = i \textit{ for some (unique) } q \in Q \\ \mathsf{ScPIR.Send}(1^\lambda, 0) & \textit{otherwise} \end{cases}$$

KeyGen *outputs*

$$\mathsf{pk} = (\tilde{q}_1, \ldots, \tilde{q}_{k_{max}})$$

and

$$\mathsf{sk} = (\mathsf{st}, Q, \zeta, (s_1, \ldots, s_{k_{max}}))$$

Processing Database $\mathsf{ProcessDB}$ *is the same as in Lemma 2.*[9]

Proving $\mathsf{Prove}^{\mathsf{dt}}(\mathsf{pp}, \mathsf{pk}, M)$ *outputs* $(y, \mathsf{d}_{\mathsf{new}}, T, (\tilde{a}_1, \ldots, \tilde{a}_{k_{max}}))$ *after sampling*

$$\begin{aligned}
&y \leftarrow M^{D \to D_{\mathsf{new}}} \\
&T = \textsc{Time}(M, D) \\
&(\mathsf{d}', \mathsf{dt}') = \mathsf{ProcessDB}(D_{\mathsf{new}}) \\
&\varphi \leftarrow \mathsf{MakeCNF}(h, M, \mathsf{d}, y, \mathsf{d}', T) \\
&w \leftarrow \mathsf{Transcript}^{\mathsf{dt}}(1^T, M) \\
&\pi \leftarrow P_{\mathsf{PCP}}(1^\lambda, \varphi, w) \\
&\tilde{a}_i \leftarrow \mathsf{ScPIR.Respond}(\tilde{q}_i, \pi) \ \textit{ for } i = 1, \ldots, k_{max}
\end{aligned}$$

Verifying $\mathsf{Verify}(\mathsf{sk}, \mathsf{pp}, (M, \mathsf{d}, y, \mathsf{d}', T), \mathsf{pf} = (\tilde{a}_1, \ldots, \tilde{a}_{k_{max}})$ *computes*

$$A : Q \to \Gamma$$

where for every $q \in Q$,

$$A(q) = \mathsf{ScPIR.Decode}(s_{\zeta(q)}, \tilde{a}_{\zeta(q)})$$

and outputs $V_1^{\hat{\phi}}(\mathsf{st}, A)$, *where* $\hat{\phi}$ *is the low-degree extension of* $\varphi = \mathsf{MakeCNF}(h, M, \mathsf{d}, y, \mathsf{d}', T)$ *and queries to* $\hat{\phi}$ *are efficiently answerable by Lemma 3.*

Theorem 2. *Construction 1 is an adaptively secure non-interactive RAM delegation protocol.*

Proof. Completeness is easy to see; we therefore focus on proving soundness.

Suppose for contradiction that there is a poly-sized prover ensemble $\{P^*_\lambda\}_\lambda$ and a constant $c > 0$ such that for infinitely many λ (let Λ denote this set of λ),

$$\Pr\left[\begin{array}{l} \mathsf{Verify}(\mathsf{sk}, h, (M, \mathsf{d}, y_0, \mathsf{d}'_0, T), \mathsf{pf}_0) = 1 \ \wedge \\ \mathsf{Verify}(\mathsf{sk}, h, (M, \mathsf{d}, y_1, \mathsf{d}'_1, T), \mathsf{pf}_1) = 1 \end{array} \middle| \begin{array}{l} h \leftarrow \mathsf{Setup}(1^\lambda) \\ (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda) \\ (M, \mathsf{d}, y_0, \mathsf{d}'_0, \mathsf{pf}_0, y_1, \mathsf{d}'_1, \mathsf{pf}_1, 1^T) \leftarrow P^*_\lambda(h, \mathsf{pk}) \end{array}\right] > \lambda^{-c}$$

Let H_λ denote the set of h for which

$$\Pr\left[\begin{array}{l} \mathsf{Verify}(\mathsf{sk}, h, (M, \mathsf{d}, y_0, \mathsf{d}'_0, T), \mathsf{pf}_0) = 1 \ \wedge \\ \mathsf{Verify}(\mathsf{sk}, h, (M, \mathsf{d}, y_1, \mathsf{d}'_1, T), \mathsf{pf}_1) = 1 \end{array} \middle| \begin{array}{l} (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda) \\ (M, \mathsf{d}, y_0, \mathsf{d}'_0, \mathsf{pf}_0, y_1, \mathsf{d}'_1, \mathsf{pf}_1, 1^T) \leftarrow P^*_\lambda(h, \mathsf{pk}) \end{array}\right] > \frac{\lambda^{-c}}{2}.$$

[9] For the curious reader, $\mathsf{ProcessDB}(D, h)$ simply computes the hash-tree of D with respect to h, and lets d be the root.

By a simple probability argument, the above equations, together with the definition of H_λ, imply that for every $\lambda \in \Lambda$,

$$\Pr_{h \leftarrow \mathsf{Setup}(1^\lambda)} [h \in H_\lambda] \geq \frac{\lambda^{-c}}{2}$$

We define the ensemble $\{P^*_{\mathsf{PCP,h}}(\cdot)\}_\lambda$ such that on input $Q \subseteq N$ (where N is the length of a proof), it does the following:

1. Pick a random injection $\zeta : Q \to [k_{max}]$ and, for each $i \in k_{max}$, define
$$(\tilde{q}_i, s_i) = \begin{cases} \mathsf{ScPIR.Send}(1^\lambda, q) & \text{if } \zeta(q) = i \text{ for some (unique) } q \in Q \\ \mathsf{ScPIR.Send}(1^\lambda, 0) & \text{otherwise} \end{cases}$$
2. Define a public-key $\mathsf{pk} = (\tilde{q}_1, \ldots, \tilde{q}_{k_{max}})$.
3. Compute $(M, \mathsf{d}, y_0, \mathsf{d}'_0, \mathsf{pf}_0, y_1, \mathsf{d}'_1, \mathsf{pf}_1, 1^T) \leftarrow P^*_\lambda(h, \mathsf{pk})$, where $\mathsf{pf}_b = (\tilde{a}^b_1, \ldots, \tilde{a}^b_{k_{max}})$.
4. Decrypt $a^b_i \leftarrow \mathsf{ScPIR.Decode}(s_i, \tilde{a}^b_i)$ for each $b \in \{0,1\}$ and $i \in \{1, \ldots, k_{max}\}$. Define
$$A_b : Q \to \Gamma$$
so that for every $q \in [Q]$,
$$A_b(q) = a^b_{\zeta(q)}$$
5. For $b \in \{0,1\}$, compute $\varphi_b \leftarrow \mathsf{MakeCNF}(h, M, \mathsf{d}, y_b, \mathsf{d}'_b, T)$.
6. Output $(\varphi_0, \varphi_1, A_0, A_1)$.

By definition, it follows immediately that

$$\Pr\left[\begin{array}{l} V_1(\mathsf{st}, \varphi_0, A_0) = 1 \ \wedge \\ V_1(\mathsf{st}, \varphi_1, A_1) = 1 \end{array} \middle| \begin{array}{l} (Q, \mathsf{st}) \leftarrow V_0(1^\lambda) \\ (\varphi_0, \varphi_1, A_0, A_1) \leftarrow P^*_{\mathsf{PCP,h}}(Q) \end{array} \right]$$

is the same as

$$\Pr\left[\begin{array}{l} \mathsf{Verify}(\mathsf{sk}, h, (M, \mathsf{d}, y_0, \mathsf{d}'_0, T), \mathsf{pf}_0) = 1 \ \wedge \\ \mathsf{Verify}(\mathsf{sk}, h, (M, \mathsf{d}, y_1, \mathsf{d}'_1, T), \mathsf{pf}_1) = 1 \end{array} \middle| \begin{array}{l} (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda) \\ (M, \mathsf{d}, y_0, \mathsf{d}'_0, \mathsf{pf}_0, y_1, \mathsf{d}'_1, \mathsf{pf}_1, 1^T) \leftarrow P^*_\lambda(h, \mathsf{pk}) \end{array} \right].$$

If $h \in H_\lambda$, this probability is at least $\lambda^{-c}/2$. Furthermore, for each h, $P^*_{\mathsf{PCP,h}}(\cdot)$ produces φ_0, φ_1 distributed as

$$\begin{array}{l} (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda) \\ (M, \mathsf{d}, y_0, \mathsf{d}'_0, y_1, \mathsf{d}'_1) \leftarrow P^*_\lambda(h, \mathsf{pk}) \\ \varphi_b \leftarrow \mathsf{MakeCNF}(h, M, \mathsf{d}, y_b, \mathsf{d}'_b, T) \end{array}$$

Claim 7. *For all h, $\{P^*_{\mathsf{PCP,h}}(\cdot)\}_\lambda$ is a k_{max}-computationally no-signaling ensemble.*

Proof. This follows from the security of the PIR scheme. Suppose otherwise – namely, suppose there exists $Q' \subseteq Q$ such that the distributions

$$(\varphi_0, \varphi_1, (A_0)_{Q'}, (A_1)_{Q'}) \Big| (\varphi_0, \varphi_1, A_0, A_1) \leftarrow P^*_{\mathsf{PCP,h}}(Q) \tag{8}$$

and

$$(\varphi_0, \varphi_1, A_0, A_1) \Big| (\varphi_0, \varphi_1, A_0, A_1) \leftarrow P^*_{\mathsf{PCP,h}}(Q') \tag{9}$$

are efficiently distinguishable. By averaging, there exists some injection $\zeta : Q \to [k_{max}]$ such that using this ζ in the definition of $P^*_{\mathsf{PCP,h}}$ still yields distinguishable results. By a hybrid argument, we can assume without loss of generality that $|Q| = |Q'| + 1$. In particular, let q^* denote the sole element of $Q \setminus Q'$ and let i^* denote $\zeta(q^*)$. Now, given a PIR query $\tilde{q}^* = \mathsf{ScPIR.Send}(1^\lambda, q, N)$, we show how to distinguish the case when $q = q^*$ from the case when $q = 0$.

First, for all $i \in [k_{max}] \setminus \{i^*\}$, sample

$$(\tilde{q}_i, s_i) = \begin{cases} \mathsf{ScPIR.Send}(1^\lambda, q) & \text{if } i = \zeta(q) \text{ for some (unique) } q \in Q' \\ \mathsf{ScPIR.Send}(1^\lambda, 0) & \text{otherwise.} \end{cases}$$

Define $\tilde{q}_{i^*} = \tilde{q}^*$. Then, compute

$$(M, \mathsf{d}, y_0, \mathsf{d}'_0, \tilde{\mathbf{a}}^0, y_1, \mathsf{d}'_1, \tilde{\mathbf{a}}^1) \leftarrow P^*_\lambda(h, (T_\lambda, (\mathsf{pk}_1, \dots, \mathsf{pk}_{k_{max}}), (\tilde{q}_1, \dots, \tilde{q}_{k_{max}}))).$$

For each $i \in \zeta(Q')$, compute

$$a_i^b \leftarrow \mathsf{ScPIR.Decode}(s_i, \tilde{\mathbf{a}}_i^b).$$

From this, compute $(\varphi_0, \varphi_1, A_0|_{Q'}, A_1|_{Q'})$ as in $P^*_{\mathsf{PCP,h}}(\cdot)$.

If q were q^*, then $(\varphi_0, \varphi_1, A_0|_{Q'}, A_1|_{Q'})$ would be distributed as in (8). If it were 0, then it would be distributed as in (9). By assumption, these distributions are efficiently distinguishable, so we have broken the security of the PIR scheme. □

Now, for $h \in H_\lambda$, we have a k_{max}-CNS PCP prover which with probability $\lambda^{-c}/2$ convinces the PCP verifier to accept two conflicting proofs. Thus, by Theorem 1, there is a PPT oracle algorithm Assign_c and a negligible function negl such that for all $\lambda \in \Lambda$ and all $h \in H_\lambda$, $\mathsf{Assign}_c^{P^*_{\mathsf{PCP,h}}(\cdot)}$ is an adaptive $2\ell(\lambda)$-local assignment generator. Furthermore, for any set of variables $V \subseteq [T'_\lambda]$ with $|V| \le 2\ell(\lambda)$, when we sample

$$(\varphi_0, \varphi_1, A_0, A_1) \leftarrow \mathsf{Assign}_c^{P^*_{\mathsf{PCP,h}}(\cdot)}(1^\lambda, V),$$

we have that the distribution on (φ_0, φ_1) is indistinguishable from that obtained in the course of running $P^*_{\mathsf{PCP,h}}$, conditioned on $P^*_{\mathsf{PCP,h}}$ convincing the verifier. In particular, there are (with high probability) $(M, \mathsf{d}, y_0, \mathsf{d}'_0, y_1, \mathsf{d}'_1, T)$ such that:

For each $b \in \{0, 1\}$, φ_b is the formula

$$\varphi_{in}^{\mathsf{d}} \wedge \varphi^{M,T,h} \wedge \varphi_{out}^{y,\mathsf{d}'}.$$

We know that any locally consistent assignments to such pairs of formulas must agree on $V_0.q \cup V_0.\mathsf{d} \cup V_0.y$ and with non-negligible probability they must disagree on $V_T.q \cup V_T.\mathsf{d} \cup V_T.y$. Thus, there exists some i such that when querying

$$(\varphi_0, \varphi_1, A_0, A_1) \leftarrow \mathsf{Assign}_c^{P^*_{\mathsf{PCP,h}}(\cdot)}(1^\lambda, V_i \cup V_{i+1}),$$

A_0 and A_1 are two locally consistent assignments to $\varphi^{M,T_\lambda,h}$ such that A_0 and A_1 agree on $V_i.q \cup V_i.\mathsf{d} \cup V_i.y$ but disagree on $V_{i+1}.q \cup V_{i+1}.\mathsf{d} \cup V_{i+1}.y$.

Thus for $h \in H_\lambda$, with non-negligible probability, $\mathsf{FindCollision}(1^T, \varphi^{M,T,h}, A_0, A_1)$ outputs a collision in h. Since $h \in H_\lambda$ with non-negligible probability, this violates the assumption that h is sampled from a collision-resistant hash family. □

6 Batch Arguments of Knowledge for NP

In this section, we consider *batch arguments of knowledge* for NP languages, in which a prover wants to prove that each of $x_1, \dots, x_k$ is in $\mathcal{L}$. We will show a 2-message protocol where the communication complexity and the running time of the verifier are both $m \cdot \mathsf{poly}(\lambda)$, where m is the size of a witness that a single x_i is in $\mathcal{L}$, and as previously, λ is the security parameter.

We emphasize that we only prove non-adaptive soundness of this 2-message protocol, as opposed to RAM delegation where we proved adaptive soundness and hence got a non-interactive delegation scheme. Intuitively, the reason for this discrepancy is that in the case of NP, a polynomial size distinguisher cannot distinguish between $x_i \in \mathcal{L}$ and $x_i \notin \mathcal{L}$, whereas in the case of polynomial-time RAM computations, a polynomial time distinguisher can do the computation on its own. We elaborate on this later.

Definition 11 (BARKs). *A 2-message Batch Argument of Knowledge (BARK) for an NP language $\mathcal{L}$ defined by $R_{\mathcal{L}}$ is a tuple of algorithms (V_0, V_1, P) satisfying the following properties:*

Completeness *For all $x_1, \dots, x_k$ and $w_1, \dots, w_k$ such that for each $i \in [k]$, $R_{\mathcal{L}}(x_i, w_i) = 1$, it holds that*

$$\Pr\left[V_1(\mathsf{st}, \pi) = 1 \middle| \begin{array}{l} (q, \mathsf{st}) \leftarrow V_0(1^\lambda, x_1, \dots, x_k) \\ \pi \leftarrow P(q, (x_1, \dots, x_k), (w_1, \dots, w_k)) \end{array} \right] = 1$$

(Non-Adaptive) Proof of Knowledge *There exists an algorithm E such that for all poly-sized P^*_λ and all $x_1, \dots, x_k$, if*

$$\Pr\left[V_1(\mathsf{st}, \pi) = 1 \middle| \begin{array}{l} (q, \mathsf{st}) \leftarrow V_0(1^\lambda, x_1, \dots, x_k) \\ \pi \leftarrow P^*_\lambda(q) \end{array} \right] \geq \epsilon$$

then $E(P^*, (x_1, \dots, x_k))$ outputs witnesses $(w_1, \dots, w_k)$ in time $\mathsf{poly}(|P^*|, |x_1| + \dots + |x_k|, \frac{1}{\epsilon})$ such that for each i, $R_{\mathcal{L}}(x_i, w_i) = 1$.

Remark. In our construction, the knowledge extractor uses P^* as a black-box, and runs in time $\mathsf{poly}(|x_1| + \dots + |x_k|, \frac{1}{\epsilon})$. For the sake of generality, the definition above allows for non-black extractors.

Theorem 3. *Let $\mathcal{L} \in \mathsf{NP}$ be any language defined by a relation $R_{\mathcal{L}}$ such that $x \in \mathcal{L}$ if and only if there is a witness w such that $R_{\mathcal{L}}(x, w) = 1$. If a succinct PIR scheme exists as in Definition 4, then there is a BARK for $\mathcal{L}$ with the following efficiency:*

- *The length of a proof π when sampling*

$$\pi \leftarrow P(q, (x_1, \dots, x_k), (w_1, \dots, w_k))$$

is $(\max_i |w_i|) \cdot \mathsf{poly}(\lambda)$. Furthermore the total communication complexity of V's interaction with P is also $(\max_i |w_i|) \cdot \mathsf{poly}(\lambda)$.

- *The total running time of (V_0, V_1) is $(\sum_i |x_i| + \max_i |w_i|) \cdot \mathsf{poly}(\lambda)$.*

Remark. If the verifier is given (or has pre-computed) the digests of each x_i, the verifier can actually run in time $(k + \max_i |w_i|) \cdot \mathsf{poly}(\lambda)$.

Remark. From now on, we write $f = \tilde{O}(g)$ if there is a polynomial poly such that $f(\lambda) \leq g(\lambda) \cdot \mathsf{poly}(\lambda)$, departing from the usual convention that $\tilde{O}$ hides logarithmic factors.

Proof Overview. Let us suppose for simplicity that all the x_i's have the same length n, and all w_i's have the same length m.

Recall that Theorem 1[10], together with a succinct PIR (as in Theorem 2), allows one to prove for any ℓ and any satisfiable 3-CNF φ, that φ is ℓ-locally satisfiable via a 2-message (adaptively sound) protocol with communication complexity $\tilde{O}(\ell)$, where the running time of the verifier is $\tilde{O}(\ell + T_{\hat{\phi}})$, where $T_{\hat{\phi}}$ is the time it takes to evaluate the low-degree extension of ϕ, where ϕ is the clause indicator function of φ. We denote by $\hat{\phi}$ the low-degree extension of ϕ.

It therefore suffices to define a 3-CNF $\varphi_{x_1,\dots,x_k}$ such that:

- $\varphi_{x_1,\dots,x_k}$ is $\tilde{O}(m)$-locally satisfiable if and only if $x_i \in \mathcal{L}$ for every i. Furthermore, for any i, a witness that $x_i \in \mathcal{L}$ can be efficiently extracted from any such local assignment generator.

[10] In fact, a weaker variant of Theorem 1 without adaptivity suffices.

- $\hat{\phi}_{x_1,\ldots,x_k}$ is computable in time $\tilde{O}(kn+m)$.

First we construct φ_x that is satisfiable if and only if $x \in \mathcal{L}$, for a single x, and we ensure that given a $\tilde{O}(m)$-local assignment generator for φ_x, one can efficiently extract a witness w such that $R_{\mathcal{L}}(x,w)=1$. Once we have done this, we build $\varphi_{x_1,\ldots,x_k} = \bigwedge_i \varphi_{x_i}$, and give each φ_{x_i} a disjoint set of variables. One point that we must carefully address is that the clause indicator function of $\varphi_{x_1,\ldots,x_k}$ needs to be efficiently computable (i.e. in time $\tilde{O}(kn+m)$). This is done in Lemma 5.

We first show the existence of such a φ_x when $m=\Omega(n)$. To this end, consider the RAM machine M which operates on a database whose initial contents are $x\|w$, and computes $R_{\mathcal{L}}(x,w)$. Let T be a bound on the running time of $R_{\mathcal{L}}$, let $h:\{0,1\}^{2\lambda}\to\{0,1\}^{\lambda}$ be a hash function, and let d be the digest of $x\|w$ with respect to the hash function h. Lemma 2 constructs a 3-CNF $\varphi \triangleq \varphi_{in}^{\mathsf{d}} \wedge \varphi^{M,T,h} \wedge \varphi_{out}$ of a particular structure,[11] so that (we show this "local-to-global" argument in Theorem 2) given a $\mathsf{poly}(\lambda)$-local assignment generator Assign for φ and a database $D \in \mathsf{Digest}^{-1}(\mathsf{d})$, either $M^D \to 1$ or one can efficiently find a collision in h. Importantly, this holds even if the database D is chosen adaptively as a (no-signaling) function of the queries that Assign receives. A stronger version of this "local-to-global" claim is part of Theorem 2.

We define φ_x as the 3-CNF consisting of $\varphi'_{in} \wedge \varphi^{M,T,h} \wedge \varphi_{out}$, where $\varphi^{M,T,h}$ and φ_{out} are the 3-CNFs given by Lemma 2 as above, and where φ'_{in} is a 3-CNF on new $v=\tilde{O}(n+m)$ variables, where the first $n+m$ variables, denoted by V_D, supposedly contain $x\|w$, and the other $v-(n+m)$ variables are auxiliary variables, denoted by V_{aux}. The clauses of φ'_{in} ensure that $V_0.\mathsf{d} = \mathsf{Digest}(V_D)$, and ensure that the first n variables of V_D are x.

Now suppose we are given a $\tilde{O}(n)$-local assignment generator $\mathcal{G}$ for φ_x, and imagine querying $V_D \cup V_{aux} \cup V_0.\mathsf{d} \cup Q$ for any Q with $|Q| \le \mathsf{poly}(\lambda)$. We can view this as an adaptive $\mathsf{poly}(\lambda)$-local assignment generator for $\varphi^{M,T,h}$ on queries Q, which by local satisifiability, must sastify the following:

- Define D by the answers on V_D. Then D must be of the form $x\|w$.
- The answers on V_{d} must be equal to $\mathsf{Digest}(x\|w)$.
- The answers on Q must locally satisfy $\varphi_{M,\mathsf{Digest}(x\|w),h}$.

By the "local-to-global" argument of Theorem 2 referenced above, it means that $M^D \to 1$. Furthermore, the answers on V_D directly give us a witness to this fact, so extractability is straight-forward.

Note that above the locality of the local assignment generator is $\tilde{O}(m+n)$. Note that if $m=o(n)$ then we do not get the efficiency we desire, of locality $\tilde{O}(m)$. To obtain our desired efficiency, we think of M as a RAM with two databases, one which initially holds x and the other which initially holds w. The point here is that instead of computing the digest of $x\|w$, φ_x will have the digest of x hard-coded, and will only compute the digest of w. As a result we will need a $\tilde{O}(m)$-local assignment generator $\mathcal{G}$ for φ_x, in order to extract w, as desired.

We proceed with a formal proof of Theorem 3.

Proof. Let $T(\cdot)$ and $m(\cdot)$ be polynomials such that there is a Turing machine M which computes $R_{\mathcal{L}}(x,w)$ in time $T(|x|)$, provided that $|w| \le m(|x|)$. Without loss of generality, we can think of M as a two-tape machine, where the first tape initially holds x and the second tape initially holds w.

We begin with the following lemma, which sums up the required modifications to Lemma 2.

Lemma 4. *There exist deterministic polynomial-time algorithms*

$$\mathsf{MakeCNF}, \mathsf{Transcript}, \mathsf{FindCollision},$$

such that for any hash function $h:\{0,1\}^{2\lambda}\to\{0,1\}^{\lambda}$, any input x with digest d_x, $\mathsf{MakeCNF}(h,x)$ is a 3-CNF φ with the following structure: Let $T=T(|x|)$ and $m=m(|x|)$. Then, $\varphi \triangleq \varphi_{in}^{\mathsf{d}_x} \wedge \varphi_{wit}^{h,m} \wedge \varphi_{main}^{M,T,h} \wedge \varphi_{out}$. We will now define each component.

[11] In Lemma 2, the 3-CNF $\varphi_{out} = \varphi_{out}^{y,d'}$ checks that the output is indeed y and checks that the digest of the database at the end of the computation is d'. In our setting, we do not care about d', and we check that the output of the computation is 1, therefore we omit the notation of y, d' from φ_{out}.

1. *φ has $T' = \tilde{O}(T \cdot |M| + m)$ variables. The set of variables V contains $T+1$ disjoint sets $V_0, \ldots, V_T$ each of size $\tilde{O}(|M|)$, as well as a set V_{wit} with $\tilde{O}(m)$ variables. Each set V_i (for $i = 1, \ldots, T$) contains (in addition to $\tilde{O}(m)$ auxiliary variables):*

   - *Variables $V_i.q$ whose values encode the Turing machine's local state after i execution steps.*
   - *Variables $V_i.\mathsf{d}_1$ and variables $V_i.\mathsf{d}_2$ whose values are the digests of the two tapes after i execution steps. The first tape initially holds x and the second tape initially holds w.*
   - *Variables $V_i.y$ whose values encode the output of M if it has terminated by the i^{th} step, and encode $\perp$ otherwise.*

   V_{wit} has variables $V_{wit}.w$ whose values encode a witness that $x \in \mathcal{L}$. V_{wit} also has auxiliary variables $V_{wit}.aux$.

2. *$\varphi_{in}^{\mathsf{d}_x}$ has clauses on V_0, which enforce that $V_0.q = M.q_0$, $V_0.\mathsf{d}_1 = \mathsf{d}_x$, and $V_0.y = \perp$.*

3. *$\varphi_{wit}^{h,m}$ has $\tilde{O}(m)$ clauses on $V_{wit} \cup V_0$ which enforce that $V_0.\mathsf{d}_2 = \mathsf{Digest}(h, V_{wit}.w)$,*

4. *$\varphi_{main}^{M,T,h}$ has clauses only within each V_i and between V_i and V_{i+1}. These clauses are identically repeated between every pair of (V_i, V_{i+1}).*

5. *φ_{out} has only one clause on V_T, which enforces that $V_T.y = 1$.*

Furthermore:

1. *If $R_{\mathcal{L}}(x, w) = 1$ then $\mathsf{Transcript}(x, w)$ outputs a satisfying assignment A for φ in time $T \cdot |M| \cdot \mathsf{poly}(\lambda)$ such that $A(V_{wit}.w) = w$. If $R_{\mathcal{L}}(x, w) = 0$, then $\mathsf{Transcript}(x, w)$ outputs an assignment which satisfies all clauses except for the one in φ_{out} (that is, $V_T.y = 0$).*

2. *For any two locally-consistent assignments A and A' for φ, both on $V_i \cup V_{i+1}$, such that:*

   - *$A|_{V_i.q \cup V_i.\mathsf{d}_1 \cup V_i.\mathsf{d}_2 \cup V_i.y} \equiv A'|_{V_i.q \cup V_i.\mathsf{d} \cup V_i.y}$ (i.e. are equal as functions) but*
   - *$A|_{V_{i+1}.q \cup V_{i+1}.\mathsf{d}_1 \cup V_{i+1}.\mathsf{d}_2 \cup V_{i+1}.y} \not\equiv A'|_{V_{i+1}.q \cup V_{i+1}.\mathsf{d} \cup V_{i+1}.y}$*

   it holds that $\mathsf{FindCollision}(1^T, \varphi, A, A')$ outputs (z, z') such that $z \neq z'$ and $h(z) = h(z')$ – in other words, a collision under h.

Proof. (Sketch) An assignment to any layer V_i contains the two digests corresponding to the two work tapes of the Turing machine M at computation step i (where initially the first work tape contains x and the second worktape contains a corresponding witness w), it contains the memory operations (i.e. reads or writes) that are performed by M on the i^{th} step, as well as the alleged results of these operations (i.e. the value read from memory or the value written to memory). This is used to compute the internal state of M on the $i + 1^{th}$ step, which in turn is used to compute the updated digests of the two work tapes in step $i + 1$. The constraints on V_i use cryptographic machinery (based on Merkle trees) to ensure that the claimed memory operation results are consistent with the previous memory digests. □

Claim 8. *Fix any x and any hash function h chosen from a collision-resistant hash family. Let φ be the 3-CNF as in Lemma 4, corresponding to x and h. Given a (non-adaptive) $\tilde{O}(m)$-local assignment generator* Assign *for φ, one can efficiently obtain a witness w such that $R_{\mathcal{L}}(x, w) = 1$ with overwhelming probability.*

Proof. We will show that querying $\mathsf{Assign}(V_{wit}.w)$ yields the desired witness with overwhelming probability. We know from local consistency and the definition of $\varphi_{in}^{\mathsf{d}_x}$ and $\varphi_{wit}^{h,m}$ that if we query $A \leftarrow \mathsf{Assign}(V_{wit} \cup V_0)$ and then compute $A' \leftarrow \mathsf{Transcript}(x, A(V_{wit}.w))$, then $A|_{V_0.q \cup V_0.\mathsf{d}_1 \cup V_0.\mathsf{d}_2 \cup V_0.y}$ is equal (as a string) to $A'|_{V_0.q \cup V_0.\mathsf{d}_1 \cup V_0.\mathsf{d}_2 \cup V_0.y}$ with overwhelming probability (and both are equal to $(q_0, d_x, d_w, \perp)$ with overwhelming probability, where d_w is the digest of $A(V_{wit}.w)$).

Similarly, by local consistency, and by the definition of φ_{out}, if we query $A \leftarrow \mathsf{Assign}(V_{wit} \cup V_T)$, then $A(V_T.y) = 1$ with overwhelming probability.

We next show that with overwhelming probability $A'(V_T.y) = 1$ where $A' \leftarrow \mathsf{Transcript}(x, A(V_{wit}.w))$, which implies that $R_{\mathcal{L}}(x, V_{wit}.w) = 1$. Suppose otherwise – that with non-negligible probability $A'(V_T.y) \neq 1$. Then there is some i such that when querying $A \leftarrow \mathsf{Assign}(V_{wit} \cup V_i \cup V_{i+1})$ and $A' \leftarrow \mathsf{Transcript}(x, A(V_{wit}.w))$, it holds with non-negligible probability that

$$A|_{V_i.q \cup V_i.\mathsf{d}_1 \cup V_i.\mathsf{d}_2 \cup V_i.y} \equiv A'|_{V_i.q \cup V_i.\mathsf{d}_1 \cup V_i.\mathsf{d}_2 \cup V_i.y}$$

but

$$A|_{V_{i+1}.q \cup V_{i+1}.\mathsf{d}_1 \cup V_{i+1}.\mathsf{d}_2 \cup V_{i+1}.y} \not\equiv A'|_{V_{i+1}.q \cup V_{i+1}.\mathsf{d}_1 \cup V_{i+1}.\mathsf{d}_1 \cup V_{i+1}.y},$$

where again $\equiv$ denotes equality as functions. This implies that we can efficiently find a collision in h by using $\mathsf{FindCollision}$, which is a contradiction.

In this argument, we only ever needed to query Assign at $\tilde{O}(m)$ variables – specifically, we queried the number of variables in V_{wit} plus the number of variables in any V_i. Hence, it is only required that Assign is a $\tilde{O}(m)$-local assignment generator. □

Corollary 2. *For any $x_1, \ldots, x_k$, let $\varphi_1, \ldots, \varphi_k$ be as in Lemma 4. Given an $\tilde{O}(m)$-local assignment generator for $\varphi = \bigwedge_i \varphi_i$, one can efficiently find witnesses $w_1, \ldots, w_k$ so that for each i, $R_{\mathcal{L}}(x_i, w_i) = 1$.*

Proof. This follows from the fact that an $\tilde{O}(m)$-local assignment generator for $\varphi = \bigwedge_i \varphi_i$ is an $\tilde{O}(m)$-local assignment generator for each φ_i. □

Lemma 5. *Let $x_1, \ldots, x_k \in \{0,1\}^n$ be any strings. Let $h : \{0,1\}^{2\lambda} \to \{0,1\}^\lambda$ be any hash function. For each $i \in \{1, \ldots, k\}$, define*

$$\varphi_i = \mathsf{MakeCNF}(h, x_i).$$

Let V be the variables of $\varphi = \bigwedge_i \varphi_i$.

For any finite field $\mathbb{F}$ with $|\mathbb{F}| > \log|V|$, and any subset $H \subseteq \mathbb{F}$ with $|H| = \lceil \log|V| \rceil$, let $m = \lceil \log|V| / \log|H| \rceil$. There is an efficiently computable and invertible mapping $e : V \hookrightarrow H^m$ such that the low-degree extension

$$\hat{\phi} : \mathbb{F}^{3m+3} \to \mathbb{F}$$

of the clause-indicator function

$$\phi : H^{3m+3} \to \{0,1\}$$

$$\phi(i_1, i_2, i_3, b_1, b_2, b_3) = \begin{cases} 1 & \textit{if } \varphi \textit{ contains the clause } e^{-1}(i_1) = b_1 \vee e^{-1}(i_2) = b_2 \vee e^{-1}(i_3) = b_3 \\ 0 & \textit{otherwise} \end{cases}$$

is efficiently computable at any point $z \in \mathbb{F}^{3m+3}$ in time $\tilde{O}(kn + m)$.

Proof. We will need the following claims about functions whose low-degree extensions can be efficiently computed.

Claim 9. *For any k, we can compute the low-degree extension of*

$$\mathsf{eq}^{\geq k} : H^{3m} \to \{0,1\}$$

$$\mathsf{eq}^{\geq k}(x, y, z) = \begin{cases} 1 & \textit{if } x = y = z \geq k \\ 0 & \textit{otherwise} \end{cases}$$

in $\mathsf{poly}(|H|, m)$ time (assuming field operations have unit cost).

Proof. We will think of x, y, z, and k all as being m-digit base-$|H|$ numbers, left-padding with 0's if necessary. Then, $x > k$ if and only if either:

- The most significant digit of x is larger than the most significant digit of k or

- The most significant digit of x is equal to the most significant digit of k and the remaining digits of x and k form x' and k' with $x' > k'$.

Thus let $k = k_1 \| \cdots \| k_m$ where k_1 is the most significant digit. We can write the conditions for $\mathsf{eq}^{\geq k}(x, y, z)$ as a small number of disjoint cases corresponding to the first digit i at which x (or equivalent y or z) differs from k, plus another case for if $x = k$. This results in the following formula.

$$\mathsf{eq}^{\geq k}(x,y,z) = \sum_{i=1}^{m} \left(\prod_{j=1}^{i-1} \delta(x_j, y_j, z_j, k_j) \right) \left(\sum_{h > k_i} \delta(x_i, y_i, z_i, h) \right) \left(\prod_{j=i+1}^{m} \delta(x_j, y_j, z_j) \right) + \prod_{i=1}^{m} \delta(x_i, y_i, z_i, k_i)$$

□

Claim 10. *For any $k \in [H^m]$, we can compute the low-degree extension of*

$$s^{\geq k} : H^{3m} \to \{0, 1\}$$

$$s^{\geq k}(x,y,z) = \begin{cases} 1 & \textit{if } x = y = z - 1 \geq k \\ 0 & \textit{otherwise} \end{cases}$$

in $\mathsf{poly}(|H|, m)$ *time (assuming field operations have unit cost).*

Proof. This is similar to the proof of Claim 9, but more involved. We now sum over at most m^2 disjoint cases in which x can be greater than k, guessing two values:

- The first digit a at which x differs from k (and in particular, is larger).
- The first digit b after which the rest of the digits of x have the value $|H| - 1$.

Now $s^{\geq k}(x, y, z)$ is the sum of m^2 terms for each possible choice of a and b. For convenience of notation, we will consider three cases, plus an extra case for if $x = k$:

- When $a < b$, the term is

$$\left(\prod_{i=1}^{a-1} \delta(x_i, y_i, z_i, k_i) \right) \cdot \left(\sum_{h > k_a} \delta(x_i, y_i, z_i, h) \right) \cdot \left(\prod_{i=a+1}^{b-1} \delta(x_i, y_i, z_i) \right) \cdot \left(\sum_{h < |H|-1} \delta(x_b, y_b, h) \delta(z_b, h+1) \right) \cdot \left(\prod_{i=b+1}^{m} \delta(x_i, y_i, |H|-1) \delta(z_i, 0) \right)$$

- When $a = b$, the term is

$$\left(\prod_{i=1}^{a-1} \delta(x_i, y_i, z_i, k_i) \right) \cdot \left(\sum_{k_a < h < |H|-1} \delta(x_a, y_a, h) \delta(z_a, h+1) \right) \cdot \left(\prod_{i=a+1}^{m} \delta(x_i, y_i, |H|-1) \delta(z_i, 0) \right)$$

- When $a > b$, the term is

$$\left(\prod_{i=1}^{b-1} \delta(x_i, y_i, z_i, k_i) \right) \cdot \delta(x_b, y_b, k_b) \delta(z_b, k_b + 1) \cdot \left(\prod_{i=b+1}^{m} \delta(x_i, y_i, |H|-1) \delta(z_i, 0) \right)$$

if $k_b < |H| - 1$ and $k_a < |H| - 1$ and for each $i \in \{b+1, \ldots, a-1\}$, $k_i = |H| - 1$. Otherwise, the term is 0.

- The term for if $x = k$ is
$$\prod_{i=1}^{m} \delta(x_i, y_i, k_i)\delta(z_i, (k+1)_i)$$

□

Armed with the ability to compute low-degree extensions of $\mathsf{eq}^{\geq k}$ and $s^{\geq k}$, we can now write the formula for $\hat{\phi} : \mathbb{F}^{3m+3} \to \mathbb{F}$.

Let $a = \lceil \log_{|H|} k \rceil$, let $b = \lceil \log_{|H|} T \rceil$, and let $\ell = \lceil \log_{|H|} |V_{wit}| \rceil$. We will define e to map the variables of each φ_i into $\{i\} \times H^{m-a}$, where $i \in [k]$ is interpreted in H^a as number in base $|H|$. In particular, the w^{th} variable of layer V_j in φ_i is mapped to $(i, w, j + \ell) \in H^a \times H^{m-a-b} \times H^b$. The variables of V_{wit} are mapped to

$$H^a \times H^{m-a-b} \times \{t \in H^b : t < \ell\}$$

in an arbitrary way.

Recall from the statement of Lemma 4 that the clauses of each φ_i are highly repetitive. In particular, there are "small" clause-indicator functions ψ_A and ψ_B such that for any $i \in [k]$ and any $t \geq \ell$, φ_i contains the clause

$$e^{-1}((i, j_1, t)) = b_1 \vee e^{-1}((i, j_2, t)) = b_2 \vee e^{-1}((i, j_3, t)) = b_3$$

iff $\psi_A(j_1, j_2, j_3, b_1, b_2, b_3) = 1$. Also, φ_i contains the clause

$$e^{-1}((i, j_1, t)) = b_1 \vee e^{-1}((i, j_2, t)) = b_2 \vee e^{-1}((i, j_3, t+1)) = b_3$$

iff $\psi_B(j_1, j_2, j_3, b_1, b_2, b_3) = 1$.

Because φ_{wit} has $\tilde{O}(m)$ (efficiently enumerable) clauses, its low-degree extension $\hat{\phi}_{wit}$ can be computed in $\tilde{O}(m)$ time. φ_{in}^{h,M,x_i} has $\mathsf{poly}(\lambda)$ constraints which take $\tilde{O}(|x_i|)$ time to enumerate, and each of $\tilde{\psi}_A$ and $\tilde{\psi}_B$ are computable in time $\mathsf{poly}(\lambda)$. Thus, Lemma 5 follows from the following formula, where each z_j is interpreted as a tuple $(i_j, w_j, t_j) \in \mathbb{F}^a \times \mathbb{F}^{m-a-b} \times \mathbb{F}^b$.

$$\hat{\phi}(z_1, z_2, z_3, b_1, b_2, b_3) = \delta(i_1, i_2, i_3) \cdot \begin{pmatrix} \mathsf{eq}^{\geq t}(t_1, t_2, t_3)\hat{\psi}_A(j_1, j_2, j_3, b_1, b_2, b_3) + \\ s^{\geq t}(t_1, t_2, t_3)\hat{\psi}_B(j_1, j_2, j_3, b_1, b_2, b_3) + \\ \hat{\phi}_{wit}^{h,m}(w_1, t_1, w_2, t_2, w_3, t_3, b_1, b_3) \end{pmatrix} + \sum_i \hat{\phi}_{in}^{h,M,x_i}(z_1, z_2, z_3, b_1, b_2, b_3)$$

□

Now that we have shown how to efficiently compute the low-degree extension of ϕ, the construction and proof of Theorem 3 proceeds similarly to the proof of Theorem 2. We can now describe our BARK scheme. Let M be the Turing machine such that $x \in \mathcal{L} \cap \{0,1\}^n$ if and only if there is a witness $w \in \{0,1\}^{m(n)}$ such that $M(x, w) = 1$ in time $T(n)$. Let $(V_{\mathsf{PCP}}^0, V_{\mathsf{PCP}}^1, P_{\mathsf{PCP}})$ be the PCP of Theorem 1 and let

$$(\mathsf{ScPIR.Send}, \mathsf{ScPIR.Respond}, \mathsf{ScPIR.Decode})$$

be a succinct PIR scheme as defined in Definition 4.

Verifier's Message $V_0(1^\lambda, (x_1, \ldots, x_k))$ first computes

$$(Q, \mathsf{st}) \leftarrow V_{\mathsf{PCP}}^0(1^\lambda).$$

Let m be the maximum length of any witness corresponding to an x_i, and let m' be the locality in the statement of Claim 8. Let T be the corresponding running time of M. Let $k_{max} = m' \cdot \ell_0(\lambda) + \lambda^2$, where $\ell_0(\cdot)$ is defined as in the statement of Theorem 1.

V_0 samples a random injection

$$\zeta : Q \hookrightarrow [k_{max}].$$

It then defines, for each $i \in [k_{max}]$,

$$(\tilde{q}_i, s_i) = \begin{cases} \mathsf{ScPIR.Send}(1^\lambda, q) & \text{if } \zeta(q) = i \text{ for some (unique) } q \in Q \\ \mathsf{ScPIR.Send}(1^\lambda, 0) & \text{otherwise} \end{cases}$$

and sends $(\tilde{q}_1, \ldots, \tilde{q}_{m'})$ to the prover.

Prover's Message $P((h, \tilde{q}_1, \ldots, \tilde{q}_{m'}), (x_1, \ldots, x_k), (w_1, \ldots, w_k))$ uses $(w_1, \ldots, w_k)$ to compute a satisfying assignment w to the 3-CNF $\varphi = \bigwedge_i \varphi_i$, where $\varphi_i \leftarrow \mathsf{MakeCNF}(h, M, x_i, m(|x_i|), T(|x_i|))$. It generates $\pi \leftarrow P_{\mathsf{PCP}}(1^\lambda, \varphi, w)$ and computes $\tilde{a}_i \leftarrow \mathsf{ScPIR.Respond}(\tilde{q}_i, \pi)$ for $i = 1, \ldots, m'$.

The prover sends $(\tilde{a}_1, \ldots, \tilde{a}_m')$ to the verifier.

Verifier Checks The verifier then computes $A : Q \to \Gamma$ where for every $q \in Q$,

$$A(q) = \mathsf{ScPIR.Decode}(s_{\zeta(q)}, \tilde{a}_{\zeta(q)})$$

and outputs $(V^1_{\mathsf{PCP}})^{\hat{\phi}}(\mathsf{st}, A)$, where $\hat{\phi}$ is the low-degree extension of $\bigwedge_i \varphi_i$, and queries to $\hat{\phi}$ are efficiently answerable by Lemma 5.

Extraction First, the security of the PIR implies that any BARK prover P^* for which

$$\Pr\left[V_1(\mathsf{st}, \pi) = 1 \middle| \begin{array}{l} (q, \mathsf{st}) \leftarrow V_0(1^\lambda, x_1, \ldots, x_k) \\ \pi \leftarrow P^*(q) \end{array} \right] \geq \epsilon(\lambda)$$

can be turned into a k_{max}-wise CNS PCP prover P^*_{PCP} such that

$$\Pr\left[V^1_{\mathsf{PCP}}(\mathsf{st}, \varphi, A) = 1 \middle| \begin{array}{l} (Q, \mathsf{st}) \leftarrow V^0_{\mathsf{PCP}}(1^\lambda) \\ A \leftarrow P^*_{\mathsf{PCP}}(Q) \end{array} \right] \geq \epsilon(\lambda)$$

where $\varphi = \bigwedge_i \varphi_i$ and $\varphi_i = \mathsf{MakeCNF}(h, M, x_i, m, T)$. By Theorem 1, any such P^*_{PCP} can be turned into an $\tilde{O}(m)$-local assignment generator for φ. By Corollary 2, this can be used, for each i, to extract a witness that $x_i \in \mathcal{L}$.

□

References

[ABOR00] William Aiello, Sandeep N. Bhatt, Rafail Ostrovsky, and Sivaramakrishnan Rajagopalan. Fast verification of any remote procedure call: Short witness-indistinguishable one-round proofs for NP. In *ICALP*, volume 1853 of *Lecture Notes in Computer Science*, pages 463–474. Springer, 2000.

[ACC$^+$15] Prabhanjan Ananth, Yu-Chi Chen, Kai-Min Chung, Huijia Lin, and Wei-Kai Lin. Delegating RAM computations with adaptive soundness and privacy. *IACR Cryptology ePrint Archive*, 2015:1082, 2015.

[AIK10] Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. From secrecy to soundness: Efficient verification via secure computation. In *ICALP (1)*, volume 6198 of *Lecture Notes in Computer Science*, pages 152–163. Springer, 2010.

[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.

[AS98] Sanjeev Arora and Shmuel Safra. Probabilistic checking of proofs: A new characterization of NP. *J. ACM*, 45(1):70–122, 1998.

[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.

[BCCT12] Nir Bitansky, Ran Canetti, Alessandro Chiesa, and Eran Tromer. From extractable collision resistance to succinct non-interactive arguments of knowledge, and back again. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, ITCS '12, pages 326–349, 2012.

[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.

[BV14] Zvika Brakerski and Vinod Vaikuntanathan. Lattice-based FHE as secure as PKE. In Moni Naor, editor, *Innovations in Theoretical Computer Science, ITCS'14, Princeton, NJ, USA, January 12-14, 2014*, pages 1–12. ACM, 2014.

[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.

[CCHR15] Ran Canetti, Yilei Chen, Justin Holmgren, and Mariana Raykova. Succinct adaptive garbled RAM. *IACR Cryptology ePrint Archive*, 2015:1074, 2015.

[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.

[CKLR11] Kai-Min Chung, Yael Tauman Kalai, Feng-Hao Liu, and Ran Raz. Memory delegation. In *Advances in Cryptology - CRYPTO 2011 - 31st Annual Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2011. Proceedings*, pages 151–168, 2011.

[CKV10] Kai-Min Chung, Yael Tauman Kalai, and Salil P. Vadhan. Improved delegation of computation using fully homomorphic encryption. In Rabin [Rab10], pages 483–501.

[CMS99] Christian Cachin, Silvio Micali, and Markus Stadler. Computationally private information retrieval with polylogarithmic communication. In Jacques Stern, editor, *Advances in Cryptology - EUROCRYPT '99, International Conference on the Theory and Application of Cryptographic Techniques, Prague, Czech Republic, May 2-6, 1999, Proceeding*, volume 1592 of *Lecture Notes in Computer Science*, pages 402–414. Springer, 1999.

[DFH12] Ivan Damgård, 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.

[DHRW16] Yevgeniy Dodis, Shai Halevi, Ron D. Rothblum, and Daniel Wichs. Spooky encryption and its applications. Cryptology ePrint Archive, Report 2016/272, 2016. `http://eprint.iacr.org/`.

[DLN$^+$01] Cynthia Dwork, Michael Langberg, Moni Naor, Kobbi Nissim, and Omer Reingold. Succinct proofs for np and spooky interactions. Unpublished manuscript, 2001.

[DNR16] Cynthia Dwork, Moni Naor, and Guy N. Rothblum. Spooky interaction and its discontents: Compilers for succinct two-message argument systems. Cryptology ePrint Archive, Report 2016/291, 2016. `http://eprint.iacr.org/`.

[Gen09] Craig Gentry. *A fully homomorphic encryption scheme*. PhD thesis, Stanford University, 2009.

[GGP10] Rosario Gennaro, Craig Gentry, and Bryan Parno. Non-interactive verifiable computing: Outsourcing computation to untrusted workers. In Rabin [Rab10], pages 465–482.

[GK16] Shafi Goldwasser and Yael Tauman Kalai. Cryptographic assumptions: A position paper. In *Theory of Cryptography - 13th International Conference, TCC 2016-A, Tel Aviv, Israel, January 10-13, 2016, Proceedings, Part I*, pages 505–522, 2016.

[GKR08] Shafi Goldwasser, Yael Tauman Kalai, and Guy N. Rothblum. Delegating computation: interactive proofs for muggles. In Cynthia Dwork, editor, *Proceedings of the 40th Annual ACM Symposium on Theory of Computing, Victoria, British Columbia, Canada, May 17-20, 2008*, pages 113–122. ACM, 2008. Full version in [GKR15].

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

[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.

[Hoe63] Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, 1963.

[IKO05] Yuval Ishai, Eyal Kushilevitz, and Rafail Ostrovsky. Sufficient conditions for collision-resistant hashing. In Joe Kilian, editor, *Theory of Cryptography, Second Theory of Cryptography Conference, TCC 2005, Cambridge, MA, USA, February 10-12, 2005, Proceedings*, volume 3378 of *Lecture Notes in Computer Science*, pages 445–456. Springer, 2005.

[Kil92] Joe Kilian. A note on efficient zero-knowledge proofs and arguments (extended abstract). In *STOC*, 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.

[KR09] Yael Tauman Kalai and Ran Raz. Probabilistically checkable arguments. In *Advances in Cryptology - CRYPTO 2009, 29th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2009. Proceedings*, pages 143–159, 2009.

[KRR13] Yael Tauman Kalai, Ran Raz, and Ron D. Rothblum. Delegation for bounded space. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013*, pages 565–574. ACM, 2013.

[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.

[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.

[PR14] Omer Paneth and Guy N. Rothblum. Publicly verifiable non-interactive arguments for delegating computation. *IACR Cryptology ePrint Archive*, 2014:981, 2014.

[Rab10] Tal Rabin, editor. *Advances in Cryptology - CRYPTO 2010, 30th Annual Cryptology Conference, Santa Barbara, CA, USA, August 15-19, 2010. Proceedings*, volume 6223 of *Lecture Notes in Computer Science*. Springer, 2010.

[RV10] Guy N. Rothblum and Salil P. Vadhan. Are pcps inherent in efficient arguments? *Computational Complexity*, 19(2):265–304, 2010.

A The Base PCP

As in the remark following Definition 6, we can assume we are given N which bounds the number of variables of φ. First pad φ so that it has exactly N variables. Let $x = (x_1, \dots, x_N) \in \{0,1\}^N$ denote a satisfying assignment to the variables of φ. Let

$$X : \mathbb{F}^m \to \mathbb{F}$$

be the low-degree extension of x (as defined in Section 2.1), where $\mathbb{F}$ is defined so that

$$O(\log^2 N) \leq |\mathbb{F}| \leq \mathrm{polylog}(N).$$

Namely, let $H = \{0, 1, \dots, \log N - 1\}$ and let $m = \frac{\log N}{\log \log N}$, so that $N = |H|^m$. (For simplicity and without loss of generality we assume that $\log N$ and $\frac{\log N}{\log \log N}$ are integers). Since $N = |H|^m$, we can identify $[N]$ and H^m by the lexicographic order on H^m. Thus, we can view $x_1, \dots, x_N$ as indexed by $i \in H^m$ (rather than $i \in [N]$). We can hence view $x = (x_1, \dots, x_N)$ as a function $x : H^m \to \{0,1\}$ (given by $x(i) = x_i$, where we identify $[N]$ and H^m). The low-degree extension of x is the (unique) multi-variate polynomial

$$X : \mathbb{F}^m \to \mathbb{F}$$

of degree $|H| - 1$ in each variable, such that $X|_{H^m} \equiv x$

Let $\phi : (H^m)^3 \times \{0,1\}^3 \to \{0,1\}$ be the clause indicator function of φ. Namely,

$$\phi(i_1, i_2, i_3, b_1, b_2, b_3) = 1 \text{ if and only if the clause } (w_{i_1} = b_1) \vee (w_{i_2} = b_2) \vee (w_{i_3} = b_3) \text{ appears in } \varphi.$$

In what follows, we think of ϕ as a function $\phi : (H^m)^3 \times H^3 \to \{0,1\}$ where for every $(b_1, b_2, b_3) \in H^3 \setminus \{0,1\}^3$ and for every $(i_1, i_2, i_3) \in (H^m)^3$,

$$\phi(i_1, i_2, i_3, b_1, b_2, b_3) = 0.$$

We denote by $\ell = 3m + 3$, and thus $\phi : H^\ell \to \{0,1\}$.

Let

$$\hat{\phi} : \mathbb{F}^\ell \to \mathbb{F}$$

be the low-degree extension of ϕ (of degree at most $|H| - 1$ in each variable).

The fact that $X : \mathbb{F}^m \to \mathbb{F}$ encodes a satisfying assignment for φ implies that for every $z = (i_1, i_2, i_3, b_1, b_2, b_3) \in (H^m)^3 \times H^3 = H^\ell$,

$$\hat{\phi}(z) \cdot (X(i_1) - b_1) \cdot (X(i_2) - b_2) \cdot (X(i_3) - b_3) = 0 \tag{10}$$

Let $P_0 : \mathbb{F}^\ell \to \mathbb{F}$ be the ℓ-variate polynomial defined as follows:
For $z = (i_1, i_2, i_3, b_1, b_2, b_3) \in (\mathbb{F}^m)^3 \times \mathbb{F}^3 = \mathbb{F}^\ell$,

$$P_0(z) \triangleq \hat{\phi}(z) \cdot (X(i_1) - b_1) \cdot (X(i_2) - b_2) \cdot (X(i_3) - b_3)$$

Equation (10) implies that $P_0|_{H^\ell} \equiv 0$ (assuming that indeed X encodes a satisfying assignment). Moreover, the fact that X has degree $< |H|$ in each variable, and $\hat{\phi}$ has degree $< |H|$ in each variable, implies that P_0 has degree $< 2|H|$ in each variable, and hence total degree $< 2|H|\ell$.

Next we define $P_1 : \mathbb{F}^\ell \to \mathbb{F}$. For every $z = (z_1, \dots, z_\ell) \in \mathbb{F}^\ell$, let

$$P_1(z) = \sum_{h \in H} P_0(h, z_2, \dots, z_\ell) {z_1}^h$$

Note that $P_1|_{\mathbb{F} \times H^{\ell-1}} \equiv 0$. More generally, we define by induction $P_1, \dots, P_\ell : \mathbb{F}^\ell \to \mathbb{F}$ where for every $z = (z_1, \dots, z_\ell) \in \mathbb{F}^\ell$,

$$P_i(z) = \sum_{h \in H} P_{i-1}(z_1, \dots, z_{i-1}, h, z_{i+1}, \dots, z_\ell) {z_i}^h$$

Note that $P_1, \ldots, P_{\ell-1}$ have degree $< 2|H|$ in each variable, and hence total degree $< 2|H|\ell$. Note also that $P_i|_{\mathbb{F}^i \times H^{\ell-i}} \equiv 0$, and in particular $P_\ell \equiv 0$.

The PCP proof for the fact that φ is satisfiable consists of the polynomial $X : \mathbb{F}^m \to \mathbb{F}$ (which is the low-degree extension of a satisfiable assignment), and the ℓ polynomials $P_i : \mathbb{F}^\ell \to \mathbb{F}$, for $i = 0, \ldots, \ell - 1$. To these polynomials we add the polynomial $P_\ell \equiv 0$. The polynomial P_ℓ can be removed from the PCP proof (as it is the 0 polynomial) and is added just for simplicity of the notation. When the verifier queries $P_\ell(z)$ she gets 0 automatically.

Let $D_X = \mathbb{F}^m$ be the domain of X, and let $D_0, \ldots, D_\ell$ be $\ell + 1$ copies of $\mathbb{F}^\ell$, the domain of $P_0, \ldots, P_\ell$. We view $D_X, D_0, \ldots, D_\ell$ as the domains of $X, P_0, \ldots, P_\ell$, respectively. Denote,

$$D = D_X \cup D_0 \cup \ldots \cup D_\ell$$

The set D is the alphabet of queries in the PCP. We will refer to D as the domain of the PCP.

Remark. Note that the entire PCP proof can be generated in time $\mathsf{poly}(N)$.

A.1 The PCP Verifier, $V = (V_0, V_1)$

As mentioned above, we assume that the verifier V is not given φ explicitly, since we require the runtime of V to be significantly smaller than $|\varphi|$. Rather, we assume that the verifier is given oracle access to $\hat{\phi}$ which is the low-degree extension of ϕ, where ϕ is the clause indicator function of φ and is viewed as a function $\phi : H^{3m+3} \to \{0, 1\}$.[12] We also assume that the verifier is given N, the number of variables of φ.

Thus, we assume that the verifier has oracle access to honestly generated $\hat{\phi}$. That is, we assume that V can get the correct value of $\hat{\phi}(z)$ for free, for as many points $z \in \mathbb{F}^\ell$ as she wants.

Notation. We denote by λ the security parameter, we denote by a_i the i^{th} coordinate of a vector a. In particular, for a line $L : \mathbb{F} \to \mathbb{F}^\ell$, a field element $t \in \mathbb{F}$ and a coordinate $i \in \{1, \ldots, \ell\}$, we denote by $L(t)_i$ the i^{th} coordinate of the point $L(t) \in \mathbb{F}^\ell$. We say that a line $L : \mathbb{F} \to \mathbb{F}^\ell$ is orthogonal to the i^{th} coordinate if for every $t_1, t_2 \in \mathbb{F}$, we have $L(t_1)_i = L(t_2)_i$.

The verifier V makes the following tests on the PCP proof (the exact tests are described below):

- **Low Degree Test for** X.
- **Low Degree Tests for** P_i.
- **Sum Check for** P_i.
- **Consistency Test of** X **and** P_0.

We note that we will have four types of low degree tests for each P_i, rather than just one. This is only for the simplicity of the analysis. It would be sufficient to do only one test, similar to the low degree test for X (but repeated on $O(|\mathbb{F}|^2)$ random lines, rather than a single random line), since all four types of tests that we actually do (and are formally described below) can be embedded in such a test.

Formally, the verifier V makes the following tests, and accepts if and only if the PCP proof passes all of them:

1. **Low Degree Test for** X**:** Choose a random line $L : \mathbb{F} \to \mathbb{F}^m$, and query X on all the points $\{L(t)\}_{t \in \mathbb{F}}$. Check that the univariate polynomial $X \circ L : \mathbb{F} \to \mathbb{F}$ is of degree $< m|H|$.
2. **Low Degree Test for** P_i**: Type 1 (fixed** $L(0)_{i+1}$**):** For every $i \in \{0, \ldots, \ell - 1\}$ and every $u \in \mathbb{F}$, choose a random line $L : \mathbb{F} \to \mathbb{F}^\ell$, such that $L(0)_{i+1} = u$. Query P_i on all the points $\{L(t)\}_{t \in \mathbb{F}}$, and check that the univariate polynomial $P_i \circ L : \mathbb{F} \to \mathbb{F}$ is of degree $< 2|H|\ell$.

[12] Jumping ahead, we note that when we use this PCP for delegation, we will use it with specific formulas φ for which the verifier can compute $\hat{\phi}$ on his own efficiently (in time $\text{polylog}(N)$).

3. **Low Degree Test for P_i: Type 2 (orthogonal to the $(i+1)^{th}$ coordinate):** For every $i \in \{0, \ldots, \ell-1\}$, choose a random line $L : \mathbb{F} \to \mathbb{F}^\ell$ that is orthogonal to the $(i+1)^{th}$ coordinate.

   Query P_i on all the points $\{L(t)\}_{t \in \mathbb{F}}$, and check that the univariate polynomial $P_i \circ L : \mathbb{F} \to \mathbb{F}$ is of degree $< 2|H|\ell$.

4. **Low Degree Test for P_i: Type 3 (fixed $L(0)_{i+1}$; orthogonal to the i^{th} coordinate):** For every $i \in \{1, \ldots, \ell-1\}$, and every $u \in \mathbb{F}$, choose a random line $L : \mathbb{F} \to \mathbb{F}^\ell$ that is orthogonal to the i^{th} coordinate, and satisfies $L(0)_{i+1} = u$. Query P_i on all the points $\{L(t)\}_{t \in \mathbb{F}}$, and check that the univariate polynomial $P_i \circ L : \mathbb{F} \to \mathbb{F}$ is of degree $< 2|H|\ell$.

5. **Low Degree Test for P_i: Type 4 (fixed $L(0)_i$; orthogonal to the $(i+1)^{th}$ coordinate):** For every $i \in \{1, \ldots, \ell-1\}$, and every $u \in \mathbb{F}$, choose a random line $L : \mathbb{F} \to \mathbb{F}^\ell$ orthogonal to the $(i+1)^{th}$ coordinate, and satisfies $L(0)_i = u$. Query P_i on all the points $\{L(t)\}_{t \in \mathbb{F}}$, and check that the univariate polynomial $P_i \circ L : \mathbb{F} \to \mathbb{F}$ is of degree $< 2|H|\ell$.

6. **Sum Check for P_i:** For every $i \in \{1, \ldots, \ell\}$, choose a random point $z = (z_1, \ldots, z_\ell) \in \mathbb{F}^\ell$. Query P_i, P_{i-1} on all the points
   $\{(z_1, \ldots, z_{i-1}, t, z_{i+1}, \ldots, z_\ell)\}_{t \in \mathbb{F}}$, and check that for every $t \in \mathbb{F}$,

   $$P_i(z_1, \ldots, z_{i-1}, t, z_{i+1}, \ldots, z_\ell) = \sum_{h \in H} P_{i-1}(z_1, \ldots, z_{i-1}, h, z_{i+1}, \ldots, z_\ell) t^h$$

7. **Consistency of X and P_0:** Choose a random point $z = (i_1, i_2, i_3, b_1, b_2, b_3) \in (\mathbb{F}^m)^3 \times \mathbb{F}^3 = \mathbb{F}^\ell$. Query P_0 on the point z and X on the points i_1, i_2, i_3, and check that

   $$P_0(z) = \hat{\phi}(z) \cdot (X(i_1) - b_1) \cdot (X(i_2) - b_2) \cdot (X(i_3) - b_3)$$

A.1.1 Complexity of the Verifier

Note that the total number of queries made by V to the PCP proof, as well as the total number of queries made by V to the function $\hat{\phi}$, are both at most $k = 6\ell|\mathbb{F}|^2$. The time complexity of V is $\mathrm{polylog}(N)$.

B From Amplified Cheating Prover to Assignment Generator

Lemma 6. *Let (V_0, V_1, P) be the PCP from Appendix A. There exists a probabilistic polynomial-time oracle machine* Assign *and a polynomial ℓ_0 such that for every negligible $\epsilon = \epsilon(\lambda)$, every polynomial ℓ, every security parameter $\lambda \in \mathbb{N}$, and every adaptive $k_{max} = \ell \cdot \ell_0$-computational no-signaling cheating prover* Prover^**, if*

$$\Pr\left[\begin{array}{l} (Q, \mathsf{st}) \leftarrow V_0^{\otimes\lambda}(1^\lambda); \\ (\varphi_1, \varphi_2, A_1, A_2) \leftarrow \mathsf{Prover}^*(Q); \\ 1 \leftarrow \left(V_1^{\geq\lambda-r}\right)^{\hat{\phi}_1, \hat{\phi}_2}(\mathsf{st}, A_1, A_2) \end{array}\right] \geq 1 - \epsilon,$$

then $\mathsf{Assign}^{\mathsf{Prover}^*}$ *is an adaptive (ℓ, ϵ')-local assignment generator with*

$$\epsilon' = \epsilon \cdot \mathsf{poly}(\lambda) + \mathsf{negl}(\lambda) = \mathsf{negl}(\lambda).$$

Moreover, the distribution (φ'_0, φ'_1), generated by

$$(\varphi'_0, \varphi'_1, A_0, A_1) \leftarrow \mathsf{Assign}_c^{\mathsf{Prover}^*}(1^\lambda, W),$$

is computationally indistinguishable from (φ_0, φ_1) generated as above.

Fix $\ell_0 = \lambda \cdot |\mathbb{F}|^2 \leq \mathsf{poly}(\lambda)$. The probabilistic polynomial-time oracle machine Assign is defined as follows: For any $\ell \leq \mathsf{poly}(\lambda)$, any $\epsilon = \epsilon(\lambda) > 0$, and any adaptive k_{max}-computational no-signaling cheating prover Prover^*, where $k_{max} = \ell \cdot \ell_0$, such that

$$\Pr\left[\begin{array}{l} (Q, \mathsf{st}) \leftarrow V_0^{\otimes\lambda}(1^\lambda); \\ (\varphi_1, \varphi_2, , A_1, A_2) \leftarrow \mathsf{Prover}^*(Q); \\ 1 \leftarrow \left(V_1^{\geq \lambda - r}\right)^{\hat{\phi}_1, \hat{\phi}_2}(\mathsf{st}, A_1, A_2) \end{array}\right] \geq 1 - \epsilon,$$

$\mathsf{Assign}^{\mathsf{Prover}^*}(\lambda, \mathsf{q}_1, \ldots, \mathsf{q}_\ell)$ does the following:

1. For each $i \in [\ell]$, choose λ random lines $L_{i,1}, \ldots, L_{i,\lambda} : \mathbb{F} \to D_X$ such that for every $j \in [\lambda]$ it holds that $L_{i,j}(0) = q_i$. (Note that $q_i \in D_X$.)

2. Run
$$(\varphi_1', \varphi_2', \{\mathsf{a}_{i,j}^{(1)}(t)\}, \{\mathsf{a}_{i,j}^{(2)}(t)\}) \leftarrow \mathsf{Prover}^*\left(\{L_{i,j}(t)\}_{i \in [\ell], j \in [\lambda], t \in \mathbb{F}}\right).$$

3. For every $i \in [\ell]$ and every $j \in [\lambda]$ do the following for every $v \in \mathbb{F}$:

   (a) Define $f^v : \mathbb{F} \to \mathbb{F}$ by setting $f^v(t) = \mathsf{a}_{i,j}^{(1)}(t)$ for every $t \neq 0$ and setting $f^v(0) = v$. Similarly, define $g^v : \mathbb{F} \to \mathbb{F}$ by setting $g^v(t) = \mathsf{a}_{i,j}^{(2)}(t)$ for every $t \neq 0$ and setting $g^v(0) = v$.

   (b) If f^v is a polynomial of degree $\leq m \cdot |H|$ then set $v_{i,j}^{(1)} \triangleq v$. If no such v exists then set $v_{i,j}^{(1)} = \perp$. Similarly, if g^v is a polynomial of degree $\leq m \cdot |H|$ then set $v_{i,j}^{(2)} \triangleq v$. If no such v exists then set $v_{i,j}^{(2)} = \perp$.

4. For every $i \in [\ell]$, let $v_i^{(1)} = \mathsf{maj}\{v_{i,1}^{(1)}, \ldots, v_{i,\lambda}^{(1)}\}$ and let $v_i^{(2)} = \mathsf{maj}\{v_{i,1}^{(2)}, \ldots, v_{i,\lambda}^{(2)}\}$.

5. Output $\{\varphi_1', \varphi_2', (v_1^{(1)}, \ldots, v_\ell^{(1)}), (v_1^{(2)}, \ldots, v_\ell^{(2)})\}$.

We need to prove that $\mathsf{Assign}^{\mathsf{Prover}^*}$ is an adaptive (ℓ, ϵ')-local assignment generator with

$$\epsilon' = \epsilon \cdot \mathsf{poly}(\lambda) + \mathsf{negl}(\lambda).$$

Moreover, we need to prove that for any $k' \leq k_{\max}$, any $(\mathsf{q}_1, \ldots, \mathsf{q}_{k'}) \in D^{k'}$, and any $Q \subset D_X$ such that $|Q| \leq \ell$,

$$(\varphi_1, \varphi_2) \approx (\varphi_1', \varphi_2'),$$

where (φ_1, φ_2) denotes the instances that $\mathsf{Prover}^*(Q)$ outputs, and (φ_1', φ_2') denotes the instances that the assigner $\mathsf{Assign}^{\mathsf{Prover}^*}(Q)$ outputs. The latter follows immediately by the fact that Prover^* is computationally no-signaling, together with the fact that $\mathsf{Assign}^{\mathsf{Prover}^*}(Q)$ always runs the prover Prover^* with some set of queries $Q' \subseteq D_X$ and outputs the instances (φ_1', φ_2') that $\mathsf{Prover}^*(Q')$ outputs. A similar argument shows that Assign is computational no-signaling.

It thus remains to prove that Assign satisfies the local soundness condition. The proof of the latter is quite involved, and is very similar to the proof from [KRR14].[13]

Local Soundness. Fix $\ell = \ell(\lambda)$ and $\epsilon = \epsilon(\lambda)$, and fix an adaptive k_{max}-computational no-signaling cheating prover Prover^*, where $k_{max} = \ell \cdot \ell_0$, such that

$$\Pr\left[\begin{array}{l} (Q, \mathsf{st}) \leftarrow V_0^{\otimes\lambda}(1^\lambda); \\ (\varphi_1, \varphi_2, A_1, A_2) \leftarrow \mathsf{Prover}^*(Q); \\ 1 \leftarrow \left(V_1^{\geq \lambda - r}\right)^{\hat{\phi}_1, \hat{\phi}_2}(\mathsf{st}, A_1, A_2) \end{array}\right] \geq 1 - \epsilon.$$

[13] Much of the text below is indeed taken from [KRR14].

We assume, without loss of generality that $|\varphi_1|, |\varphi_2| \geq \lambda$. This is without loss of generality, since otherwise a proof can consist of an entire satisfying assignment. Roughly speaking, the proof of local soundness can be partitioned into three parts.

We prove local soundness of φ_1. The proof of local soundness of φ_2 is identical. Denote the computationally no-signaling strategy of Prover^* by

$$\{\mathcal{A}_S\}_{S \subset D, |S| \leq k_{max}},$$

where $\mathcal{A}_S$ is a distribution that generates (φ, A) as follows: Run $(\varphi_1, \varphi_2, A_1, A_2) \leftarrow \mathsf{Prover}^*(S)$, and set $\varphi = \varphi_1$ and $A = A_1$.

Part 1. We start with the following definition that will be useful in the proof. Intuitively, a point z satisfies property $\mathcal{Z}(\epsilon', r')$ if when taking λ lines through it, and sending all these points to Prover^*, then with high probability, for most of these lines, the answers of Prover^* correspond to low degree polynomials that "evaluate" the point z to 0.

Definition 12. Property $\mathcal{Z}(\epsilon', r')$:
Let $\epsilon' \geq 0$ and $r' \geq 0$. Let $i \in \{0, \ldots, \ell\}$. Let $z \in D_i$.

Let $L_1, \ldots, L_\lambda : \mathbb{F} \to D_i$ be λ random lines, such that for every $L \in \{L_1, \ldots, L_\lambda\}$, we have $L(0) = z$. Let $S = \{L_j(t)\}_{j \in [\lambda], t \in \mathbb{F}} \subset D_i$. Let $(\varphi, A) \in_R \mathcal{A}_S$.

Define $A^0 : S \to \mathbb{F}$ by $A^0(z') = A(z')$ for $z' \neq z$ and $A^0(z) = 0$.

We say that the point z satisfies property $\mathcal{Z}(\epsilon', r')$ (also denoted $z \in \mathcal{Z}(\epsilon', r')$) if with probability $\geq 1 - \epsilon'$, for at least $\lambda - r'$ of the lines $L \in \{L_1, \ldots, L_\lambda\}$, we have that $A^0 \circ L : \mathbb{F} \to \mathbb{F}$ is a univariate polynomial of degree $< 2\ell|H|$ (where the probability is over $L_1, \ldots, L_\lambda, A$).

Our main lemma about property $\mathcal{Z}(\epsilon', r')$ is that the property is satisfied, with small ϵ' and r', for *any* point $z = (z_1, \ldots, z_\ell) \in D_0$ such that $z_1, \ldots, z_\ell \in H$. (Intuitively, this is analogous to the formula $P_0|_{H^\ell} \equiv 0$, that is satisfied for $x \in \mathcal{L}$).

Lemma 7. *There exists a negligible function $\mu = \mu(\lambda)$ such that for any $z = (z_1, \ldots, z_\ell) \in D_0$ such that $z_1, \ldots, z_\ell \in H$, we have $z \in \mathcal{Z}(\epsilon', r')$, where $\epsilon' = 6\ell|\mathbb{F}|\epsilon + \mu$ and $r' = 8\ell|\mathbb{F}|r$.*

Part 2. In the second part of the proof we show that when taking a large number of lines through a point $z \in D_X$, with high probability, there exists a value $v \in \mathbb{F}$, such that for most of these lines, the answers of Prover^* correspond to low degree polynomials that "evaluate" the point z to v.

Lemma 8. *There exists a negligible function $\mu = \mu(\lambda)$ such that for any $z \in D_X$ the following holds. Let $L_1, \ldots, L_\lambda : \mathbb{F} \to D_X$ be λ random lines, such that for every $L \in \{L_1, \ldots, L_\lambda\}$, we have $L(0) = z$. Let $S = \{L_j(t)\}_{j \in [\lambda], t \in \mathbb{F}} \subset D_X$. Let $(\varphi, A) \in_R \mathcal{A}_S$.*

For any $v \in \mathbb{F}$, define $A^v : S \to \mathbb{F}$ by $A^v(z') = A(z')$ for $z' \neq z$ and $A^v(z) = v$. Let $r' = 30|\mathbb{F}|r$ and let $\epsilon' = 2|\mathbb{F}|\epsilon + \mu$. Then, with probability $\geq 1 - \epsilon'$, there exists $v \in \mathbb{F}$, such that, for at least $\lambda - r'$ of the indices $j \in [\lambda]$, $A^v \circ L_j : \mathbb{F} \to \mathbb{F}$ is a univariate polynomial of degree $< m|H|$ (where the probability is over $L_1, \ldots, L_\lambda, A$).

Part 3. In the third part of the proof we use Lemma 7 and Lemma 8 above to prove the local consistency guarantee.

Lemma 9. *There exists a negligible function $\mu = \mu(\lambda)$ such that the following holds. Let $r' = 9\ell|\mathbb{F}|r$ and let $\epsilon' = 7\ell|\mathbb{F}|\epsilon + \mu$. Let $i_1, i_2, i_3 \in H^m$ and view i_1, i_2, i_3 as points in D_X.*

Let $L_{1,1}, \ldots, L_{1,\lambda} : \mathbb{F} \to D_X$ be λ random lines, such that for every $L \in \{L_{1,1}, \ldots, L_{1,\lambda}\}$, we have $L(0) = i_1$. Let $L_{2,1}, \ldots, L_{2,\lambda} : \mathbb{F} \to D_X$ be λ random lines, such that for every $L \in \{L_{2,1}, \ldots, L_{2,\lambda}\}$, we have $L(0) = i_2$. Let $L_{3,1}, \ldots, L_{3,\lambda} : \mathbb{F} \to D_X$ be λ random lines, such that for every $L \in \{L_{3,1}, \ldots, L_{3,\lambda}\}$, we have $L(0) = i_3$.

Let $S = \{L_{1,j}(t), L_{2,j}(t), L_{3,j}(t)\}_{j \in [\lambda], t \in \mathbb{F}} \subset D_X$. Let $(\varphi, A) \in_R \mathcal{A}_S$. For any $i \in D_X$ and $v \in \mathbb{F}$, define $A^{i \to v} : S \to \mathbb{F}$ by $A^{i \to v}(i') = A(i')$ for $i' \neq i$ and $A^{i \to v}(i) = v$.

With probability $\geq 1 - \epsilon'$, there exist $v_1, v_2, v_3 \in \mathbb{F}$, such that, for at least $\lambda - r'$ of the indices $j \in [\lambda]$, the following is satisfied (where the probability is over $L_{1,1}, \ldots, L_{1,\lambda}, L_{2,1}, \ldots, L_{2,\lambda}, L_{3,1}, \ldots, L_{3,\lambda}, \varphi, A$):

1. *$A^{i_1 \to v_1} \circ L_{1,j} : \mathbb{F} \to \mathbb{F}$ is a univariate polynomial of degree $< m|H|$.*

2. *$A^{i_2 \to v_2} \circ L_{2,j} : \mathbb{F} \to \mathbb{F}$ is a univariate polynomial of degree $< m|H|$.*

3. *$A^{i_3 \to v_3} \circ L_{3,j} : \mathbb{F} \to \mathbb{F}$ is a univariate polynomial of degree $< m|H|$.*

4. *If φ contains a clause of the form $(w_{i_1} = b_1) \vee (w_{i_2} = b_2) \vee (w_{i_3} = b_3)$, then $(v_1 - b_1) \cdot (v_2 - b_2) \cdot (v_3 - b_3) = 0$.*

Note that Lemma 9, together with the computational no-signaling property, implies local soundness. As mentioned above, we prove this lemma using Lemma 7 and Lemma 8 above.

B.1 Proof of Lemma 7

In the proof of Lemma 7 we use a variant of property $\mathcal{Z}(\epsilon', r')$, where the random lines are restricted to be orthogonal to the $(i')^{th}$ coordinate. (We will use this property only for $i' \in \{i, i+1\}$). In particular, we use the following definition.

Definition 13. Property $\mathcal{Z}^{i'}(\epsilon', r')$:
Let $\epsilon' \geq 0$ and $r' \geq 0$. Let $i' \in \{1, \ldots, \ell\}$. Let $i \in \{0, \ldots, \ell\}$. Let $z \in D_i$.

Let $L_1, \ldots, L_\lambda : \mathbb{F} \to D_i$ be λ random lines, such that for every $L \in \{L_1, \ldots, L_\lambda\}$ we have $L(0) = z$, and L is orthogonal to the $(i')^{th}$ coordinate. Let $S = \{L_j(t)\}_{j \in [\lambda], t \in \mathbb{F}} \subset D_i$. Let $(\varphi, A) \in_R \mathcal{A}_S$.

Define $A^0 : S \to \mathbb{F}$ by $A^0(z') = A(z')$ for $z' \neq z$ and $A^0(z) = 0$.

We say that the point z satisfies property $\mathcal{Z}^{i'}(\epsilon', r')$ (also denoted $z \in \mathcal{Z}^{i'}(\epsilon', r')$) if with probability $\geq 1 - \epsilon'$, for at least $\lambda - r'$ of the lines $L \in \{L_1, \ldots, L_\lambda\}$, we have that $A^0 \circ L : \mathbb{F} \to \mathbb{F}$ is a univariate polynomial of degree $< 2\ell|H|$ (where the probability is over $L_1, \ldots, L_\lambda, A$).

The proof of Lemma 7 follows from the following three claims (Claims 11, 12, and 13).

Claim 11. *There exists a negligible function $\mu = \mu(\lambda) = \mathsf{negl}(\lambda)$ such that for every $\epsilon_1 \geq 0$, every $r_1 \geq 0$, every $i \in \{1, \ldots, \ell - 1\}$, and every $z \in D_i$, if $z \in \mathcal{Z}^{i+1}(\epsilon_1, r_1)$ then $z \in \mathcal{Z}^{i}(\epsilon_2, r_2)$, where $\epsilon_2 = \epsilon_1 + 2|\mathbb{F}|\epsilon + \mu(\lambda)$, and $r_2 = r_1 + 3|\mathbb{F}|r$.*

Claim 12. *There exists a negligible function $\mu = \mu(\lambda) = \mathsf{negl}(\lambda)$ such that for every $\epsilon_1 \geq 0$, every $r_1 \geq 0$, every $i \in \{0, \ldots, \ell - 1\}$, and every $z \in D_i$, if $z \in \mathcal{Z}^{i+1}(\epsilon_1, r_1)$ then $z \in \mathcal{Z}(\epsilon_2, r_2)$, where $\epsilon_2 = \epsilon_1 + 2|\mathbb{F}|\epsilon + \mu(\lambda)$, and $r_2 = r_1 + 3|\mathbb{F}|r$.*

Claim 13. *There exists a negligible function $\mu = \mu(\lambda) = \mathsf{negl}(\lambda)$ such that for every $\epsilon_1 \geq 0$, every $r_1 \geq 0$, and every $i \in \{1, \ldots, \ell\}$, the following holds: Let $z = (z_1, \ldots, z_\ell) \in \mathbb{F}^\ell$ be a point such that $z_i \in H$. For every $t \in \mathbb{F}$, let $z(t) = (z_1, \ldots, z_{i-1}, t, z_{i+1}, \ldots, z_\ell) \in \mathbb{F}^\ell$. Assume that for every $t \in \mathbb{F}$, the point $z(t)$, viewed as a point in D_i, satisfies property $\mathcal{Z}^i(\epsilon_1, r_1)$. Then the point z, viewed as a point in D_{i-1}, satisfies property $\mathcal{Z}^i(\epsilon_2, r_2)$, where $\epsilon_2 = \frac{\epsilon_1}{1-\gamma} + |\mathbb{F}|\epsilon + \mu$, and $r_2 = \frac{r_1}{1-\gamma} + 2|\mathbb{F}|r$, and $\gamma = \sqrt{\frac{|H|}{|\mathbb{F}|}}$.*

Before we prove these claims, we prove Lemma 7 based on these claims.

Proof of Lemma 7. Recall that we assume that for every distribution $\mathcal{A}_S$ in the family $\{\mathcal{A}_S\}$, every query in $S \cap D_\ell$ is answered by 0 with probability 1 (since the polynomial P_ℓ was just the 0 polynomial and was added to the PCP proof for simplicity of notations). Therefore, any point $z \in D_\ell$ satisfies property $\mathcal{Z}^\ell(\epsilon_\ell, r_\ell)$, where $\epsilon_\ell = 0$ and $r_\ell = 0$.

Combining Claim 11 and Claim 13 we obtain the following claim.

Claim 14. *There exists a negligible function $\mu = \mu(\lambda)$ for which the following holds: Let $\epsilon_1 \geq 0$. Let $r_1 \geq 0$. Let $i \in \{2, \ldots, \ell\}$. Let $z = (z_1, \ldots, z_\ell) \in \mathbb{F}^\ell$ be a point such that $z_i \in H$. For every $t \in \mathbb{F}$, let $z(t) = (z_1, \ldots, z_{i-1}, t, z_{i+1}, \ldots, z_\ell) \in \mathbb{F}^\ell$. Assume that for every $t \in \mathbb{F}$, the point $z(t)$, viewed as a point in D_i, satisfies property $\mathcal{Z}^i(\epsilon_1, r_1)$. Then the point z, viewed as a point in D_{i-1}, satisfies property $\mathcal{Z}^{i-1}(\epsilon_2, r_2)$, where $\epsilon_2 = \frac{\epsilon_1}{1-\gamma} + 3|\mathbb{F}|\epsilon + \mu$, and $r_2 = \frac{r_1}{1-\gamma} + 5|\mathbb{F}|r$, and $\gamma = \sqrt{\frac{|H|}{|\mathbb{F}|}}$.*

By inductive application of Claim 14, for any $i \in \{1, \dots, \ell - 1\}$, any point $z = (z_1, \dots, z_\ell) \in D_i$, such that $z_{i+1}, \dots, z_\ell \in H$, satisfies property $\mathcal{Z}^i(\epsilon_i, r_i)$, where $\epsilon_i = \frac{\epsilon_{i+1}}{1-\gamma} + 3|\mathbb{F}|\epsilon + \mu$ and $r_i = \frac{r_{i+1}}{1-\gamma} + 5|\mathbb{F}|r$, and $\gamma = \sqrt{\frac{|H|}{|\mathbb{F}|}}$.

In particular, any point $z = (z_1, \dots, z_\ell) \in D_1$, such that $z_2, \dots, z_\ell \in H$, satisfies property $\mathcal{Z}^1(\epsilon_1, r_1)$, where $\epsilon_1 \le \frac{3\ell|\mathbb{F}|\epsilon + \ell \cdot \mu}{(1-\gamma)^\ell} < 4\ell|\mathbb{F}|\epsilon + 2\ell \cdot \mu$ and $r_1 \le \frac{5\ell|\mathbb{F}|r}{(1-\gamma)^\ell} < 6\ell|\mathbb{F}|r$.

Hence, by Claim 13, any point $z = (z_1, \dots, z_\ell) \in D_0$, such that $z_1, \dots, z_\ell \in H$, satisfies property $\mathcal{Z}^1(\epsilon_0, r_0)$, where $\epsilon_0 = \frac{\epsilon_1}{1-\gamma} + |\mathbb{F}|\epsilon + \mu < 5\ell|\mathbb{F}|\epsilon + 3\ell \cdot \mu$ and $r_0 = \frac{r_1}{1-\gamma} + 2|\mathbb{F}|r < 7\ell|\mathbb{F}|r$.

Finally, by Claim 12, any point $z = (z_1, \dots, z_\ell) \in D_0$, such that $z_1, \dots, z_\ell \in H$, satisfies property $\mathcal{Z}(\epsilon', r')$, where $\epsilon' < 6\ell|\mathbb{F}|\epsilon + 4\ell \cdot \mu$ and $r' < 8\ell|\mathbb{F}|r$.

□

In what follows we prove Claims 11, 12, and 13.

Proof of Claim 11. Assume that $z \in \mathcal{Z}^{i+1}(\epsilon_1, r_1)$.

Let $L_1, \dots, L_\lambda : \mathbb{F} \to D_i$ be λ random lines, such that for every $L \in \{L_1, \dots, L_\lambda\}$ we have $L(0) = z$, and L is orthogonal to the $(i+1)^{st}$ coordinate. Let $L'_1, \dots, L'_\lambda : \mathbb{F} \to D_i$ be λ random lines, such that for every $L' \in \{L'_1, \dots, L'_\lambda\}$ we have $L'(0) = z$, and L' is orthogonal to the i^{th} coordinate.

Let $M_1, \dots, M_\lambda : \mathbb{F}^2 \to D_i$ be λ planes, where $M_j(t_1, t_2) = L_j(t_1) + L'_j(t_2) - z$ (where the addition/substraction are over the vector space $D_i = \mathbb{F}^\ell$).

Let $S = \{M_j(t_1, t_2)\}_{j \in [k], t_1, t_2 \in \mathbb{F}} \subset D_i$. Let $(\varphi, A) \in_R \mathcal{A}_S$. Define $A^0 : S \to \mathbb{F}$ by $A^0(z') = A(z')$ for $z' \neq z$ and $A^0(z) = 0$.

We say that M_j is *good* if the following is satisfied:

1. For every $t_1 \in \mathbb{F} \setminus \{0\}$, the function $A^0 \circ M_j(t_1, *) : \mathbb{F} \to \mathbb{F}$ is a univariate polynomial of degree $< 2\ell|H|$.

2. For every $t_2 \in \mathbb{F}$, the function $A^0 \circ M_j(*, t_2) : \mathbb{F} \to \mathbb{F}$ is a univariate polynomial of degree $< 2\ell|H|$.

By Claim 15 below (applied with $f = A^0 \circ M_j$ and $d = 2\ell|H|$), if M_j is good then $A^0 \circ L'_j = A^0 \circ M_j(0, *) : \mathbb{F} \to \mathbb{F}$ is a univariate polynomial of degree $< 2\ell|H|$.

Claim 15. *Let $f : \mathbb{F}^2 \to \mathbb{F}$ be a function. Assume that for every $t_1 \in \mathbb{F} \setminus \{0\}$, the function $f_{(t_1,*)} : \mathbb{F} \to \mathbb{F}$ is a univariate polynomial of degree $< d$, and for every $t_2 \in \mathbb{F}$, the function $f_{(*,t_2)} : \mathbb{F} \to \mathbb{F}$ is a univariate polynomial of degree $< d$, where $d < |\mathbb{F}|$. Then, $f_{(0,*)} : \mathbb{F} \to \mathbb{F}$ is a univariate polynomial of degree $< d$.*

Proof. For every $t_2 \in \mathbb{F}$, the function $f_{(*,t_2)} : \mathbb{F} \to \mathbb{F}$ is a univariate polynomial of degree $<\ d$. Therefore, there exist $a_1, \dots, a_d \in \mathbb{F}$, (where $a_1, \dots, a_d$ are the Lagrange interpolation coefficients), such that for every $t_2 \in \mathbb{F}$, we have $f(0, t_2) = \sum_{t=1}^d a_t \cdot f(t, t_2)$. That is, $f_{(0,*)} = \sum_{t=1}^d a_t \cdot f_{(t,*)}$. Since $f_{(1,*)}, \dots, f_{(d,*)}$ are univariate polynomials of degree $< d$, their linear combination $f_{(0,*)}$ is also a univariate polynomial of degree $< d$. □

We will show that with high probability, at least $\lambda - r_2$ of the planes $M \in \{M_1, \dots, M_\lambda\}$ are good (where the probability is over $L_1, \dots, L_\lambda, L'_1, \dots, L'_\lambda, A$). By Claim 15, this implies that with high probability, at least $\lambda - r_2$ of the lines $L' \in \{L'_1, \dots, L'_\lambda\}$ satisfy that $A^0 \circ L' : \mathbb{F} \to \mathbb{F}$ is a univariate polynomial of degree $< 2\ell|H|$ (where the probability is over $L_1, \dots, L_\lambda, L'_1, \dots, L'_\lambda, A$).

Claim 16. *There exists a negligible function $\mu = \mu(\lambda)$ (independent of $i \in [\ell]$) such that with probability $\ge 1 - \epsilon_1 - 2|\mathbb{F}|\epsilon - \mu$, for at least $\lambda - r_1 - 3|\mathbb{F}|r$ of the indices $j \in [\lambda]$, we have that M_j is good.*

Proof. For every $t_1 \in \mathbb{F} \setminus \{0\}$, consider the set of lines $\{M_j(t_1, *)\}_{j \in [\lambda]}$ and note that this is a set of λ random lines in D_i, such that every line $L \in \{M_j(t_1, *)\}_{j \in [\lambda]}$ is orthogonal to the i^{th} coordinate, and satisfies $L(0)_{i+1} = z_{i+1}$. Hence, by the computational no-signaling condition, there exists a negligible function $\mu_1 = \mu_1(\lambda)$ (independent of $i \in [\ell]$) such that with probability $> 1 - \epsilon - \mu_1$, for at least $\lambda - r$ of the indices $j \in [\lambda]$, we have that $A \circ M_j(t_1, *) : \mathbb{F} \to \mathbb{F}$ is a univariate polynomial of degree $< 2\ell|H|$.

Denote by E_j the event that the lines L_j, L'_j are in general position, that is, the vectors $L_j(1) - L_j(0), L'_j(1) - L'_j(0)$ span a linear subspace of dimension 2 (as vectors in $D_i = \mathbb{F}^\ell$). Note that event E_j occurs with probability $1 - \frac{1}{|\mathbb{F}|^{\ell-1}}$. Moreover, note that if event E_j occurs then $z \notin M_j(t_1, *)$, and hence $A^0 \circ M_j(t_1, *) = A \circ M_j(t_1, *)$. Therefore, if

$A \circ M_j(t_1, *) : \mathbb{F} \to \mathbb{F}$ is a univariate polynomial of degree $< 2\ell|H|$ and event E_j holds, then $A^0 \circ M_j(t_1, *) : \mathbb{F} \to \mathbb{F}$ is also a univariate polynomial of degree $< 2\ell|H|$.

For every $t_2 \in \mathbb{F} \setminus \{0\}$, consider the set of lines $\{M_j(*, t_2)\}_{j \in [\lambda]}$ and note that this is a set of λ random lines in D_i such that every line $L \in \{M_j(*, t_2)\}_{j \in [\lambda]}$ is orthogonal to the $(i+1)^{th}$ coordinate, and satisfies $L(0)_i = z_i$. Hence, by the computational no-signaling condition, there exists a negligible function $\mu_2 = \mu_2(\lambda)$ (independent of $i \in [\ell]$) such that with probability $> 1 - \epsilon - \mu_2$, for at least $\lambda - r$ of the indices $j \in [\lambda]$, we have that $A \circ M_j(*, t_2) : \mathbb{F} \to \mathbb{F}$ is a univariate polynomial of degree $< 2\ell|H|$. As above, if event E_j occurs then $z \notin M_j(*, t_2)$, and hence $A^0 \circ M_j(*, t_2) = A \circ M_j(*, t_2)$. Therefore, if $A \circ M_j(*, t_2) : \mathbb{F} \to \mathbb{F}$ is a univariate polynomial of degree $< 2\ell|H|$ and event E_j holds then $A^0 \circ M_j(*, t_2) : \mathbb{F} \to \mathbb{F}$ is also a univariate polynomial of degree $< 2\ell|H|$.

Consider the set of lines $\{M_j(*, 0)\}_{j \in [\lambda]}$ and note that $M_j(*, 0) = L_j$. The fact that $z \in \mathcal{Z}^{i+1}(\epsilon_1, r_1)$, together with the computational no-signaling condition, implies that there exists a negligible function $\mu_3 = \mu_3(\lambda)$ (independent of $i \in [\ell]$) such that with probability $\geq 1 - \epsilon_1 - \mu_3$, for at least $\lambda - r_1$ of the indices $j \in [\lambda]$, we have that $A^0 \circ M_j(*, 0) : \mathbb{F} \to \mathbb{F}$ is a univariate polynomial of degree $< 2\ell|H|$.

Recall that each event E_j occurs with probability $1 - \frac{1}{|\mathbb{F}|^{\ell-1}}$, and these events are independent. This, together with our assumption that

$$|\mathbb{F}|^{\ell-1} \geq |H|^{3\cdot(\ell-1)} \geq |H|^{3\cdot 3m} \geq \lambda^9$$

(where the latter inequality follows from our assumption that $|H|^m \geq \lambda$), implies that

$$\Pr\left[|\{j : \neg E_j\}| \geq |\mathbb{F}|\right] \leq \binom{\lambda}{|\mathbb{F}|} \cdot \left(\frac{1}{|\mathbb{F}|^{\ell-1}}\right)^{|\mathbb{F}|} \leq \left(\frac{e\lambda}{|\mathbb{F}|^{\ell}}\right)^{|\mathbb{F}|} \leq \left(\frac{1}{\lambda^8}\right)^{|\mathbb{F}|}.$$

Let $\mu_0 = \left(\frac{1}{\lambda^8}\right)^{|\mathbb{F}|}$. Note that $\mu_0 = \mathsf{negl}(\lambda)$.

Adding up all this, by the union bound, we obtain that with probability $\geq 1 - \epsilon_1 - 2|\mathbb{F}|\epsilon - \mu$, where $\mu = \mu_0 + \mu_1 + \mu_2 + \mu_3$, for at least $\lambda - r_1 - 2|\mathbb{F}|r - |\mathbb{F}| \geq \lambda - r_1 - 3|\mathbb{F}|r$ of the indices $j \in [\lambda]$, we have that:

1. For every $t_1 \in \mathbb{F} \setminus \{0\}$, $A^0 \circ M_j(t_1, *)$ is a univariate polynomial of degree $< 2\ell|H|$.
2. For every $t_2 \in \mathbb{F} \setminus \{0\}$, $A^0 \circ M_j(*, t_2)$ is a univariate polynomial of degree $< 2\ell|H|$.
3. $A^0 \circ M_j(*, 0)$ is a univariate polynomial of degree $< 2\ell|H|$.

That is, with probability at least $1 - \epsilon_1 - 2|\mathbb{F}|\epsilon - \mu$, for at least $\lambda - r_1 - 3|\mathbb{F}|r$ of the indices $j \in [\lambda]$, we have that M_j is good.

□

Proof of Claim 12. The proof of this claim is identical to the proof of Claim 11, except that we let $L'_1, \ldots, L'_\lambda : \mathbb{F} \to D_i$ be λ random lines such that for every $L' \in \{L_1, \ldots, L_\lambda\}$ we have that $L'(0) = z$ (without the requirement that L' is orthogonal to the i^{th} coordinate). □

Proof of Claim 13. Fix $\epsilon_1 \geq 0$, $r_1 \geq 0$, and $i \in \{1, \ldots, \ell\}$. Fix $z = (z_1, \ldots, z_\ell) \in \mathbb{F}^\ell$ such that $z_i \in H$. Assume that for every $t \in \mathbb{F}$, the point $z(t) = (z_1, \ldots, z_{i-1}, t, z_{i+1}, \ldots, z_\ell)$, viewed as a point in D_i, satisfies property $\mathcal{Z}^i(\epsilon_1, r_1)$.

Let $L_1, \ldots, L_\lambda : \mathbb{F} \to \mathbb{F}^\ell$ be λ random lines, such that for every $L \in \{L_1, \ldots, L_\lambda\}$, we have $L(0) = 0$, and L is orthogonal to the i^{th} coordinate.

Let $M_1, \ldots, M_\lambda : \mathbb{F}^2 \to \mathbb{F}^\ell$ be λ planes, where $M_j(t_1, t_2) = L_j(t_1) + z(t_2)$ (where the addition is over the vector space $\mathbb{F}^\ell$).

Let S^i and S^{i-1} be two copies of the set of points $\{M_j(t_1, t_2)\}_{j \in [\lambda], t_1, t_2 \in \mathbb{F}} \subset \mathbb{F}^\ell$, and view S^i as a subset of D_i and S^{i-1} as a subset of D_{i-1}. Let $S = S^i \cup S^{i-1} \subset D$. Let $(\varphi, A) \in_R \mathcal{A}_S$. Recall that we view A as a function $A : S \to \mathbb{F}$, and we denote by A_i, A_{i-1} the restriction of that function to S^i, S^{i-1}, respectively.

Define $A^0_i : S^i \to \mathbb{F}$ by $A^0_i(z') = A_i(z')$ for $z' \notin \{z(t)\}_{t \in \mathbb{F}}$, and $A^0_i(z') = 0$ for $z' \in \{z(t)\}_{t \in \mathbb{F}}$. Define $A^0_{i-1} : S^{i-1} \to \mathbb{F}$ by $A^0_{i-1}(z') = A_{i-1}(z')$ for $z' \notin \{z(t)\}_{t \in \mathbb{F}}$ and $A^0_{i-1}(z') = 0$ for $z' \in \{z(t)\}_{t \in \mathbb{F}}$.

We say that M_j is *good* if the following is satisfied:

1. For every $t_1 \in \mathbb{F}$, and every $t \in \mathbb{F}$,

$$A_i^0(M_j(t_1,t)) = \sum_{h\in H} A_{i-1}^0(M_j(t_1,h))t^h$$

2. For at least $|H|$ values $t_2 \in \mathbb{F}$, the function $A_i^0 \circ M_j(*,t_2) : \mathbb{F} \to \mathbb{F}$ is a univariate polynomial of degree $< 2\ell|H|$.

By Claim 17 below (applied with $f = A_i^0 \circ M_j$, $f' = A_{i-1}^0 \circ M_j$ and $d = 2\ell|H|$), if M_j is good then for every $t_2 \in H$, the function $A_{i-1}^0 \circ M_j(*,t_2) : \mathbb{F} \to \mathbb{F}$ is a univariate polynomial of degree $< 2\ell|H|$.

Claim 17. *Let $f : \mathbb{F}^2 \to \mathbb{F}$ and $f' : \mathbb{F}^2 \to \mathbb{F}$ be two functions. Assume that:*

1. For every $t_1 \in \mathbb{F}$ and every $t \in \mathbb{F}$,

$$f(t_1,t) = \sum_{h\in H} f'(t_1,h)t^h$$

2. For at least $|H|$ values $t_2 \in \mathbb{F}$, the function $f_{(*,t_2)} : \mathbb{F} \to \mathbb{F}$ is a univariate polynomial of degree $< d$.

Then, for every $t_2 \in H$, the function $f'_{(*,t_2)} : \mathbb{F} \to \mathbb{F}$ is a univariate polynomial of degree $< d$.

Proof. For every $h \in H$, present the function $f'_{(*,h)} : \mathbb{F} \to \mathbb{F}$ as a univariate polynomial (in the free variable y),

$$f'(y,h) = f'_{(*,h)}(y) = \sum_{s=0}^{|\mathbb{F}|-1} a_{h,s} \cdot y^s$$

where $a_{h,0}, \ldots, a_{h,|\mathbb{F}|-1} \in \mathbb{F}$. Thus, for every $y \in \mathbb{F}$ and every $t \in \mathbb{F}$,

$$f_{(*,t)}(y) = f(y,t) = \sum_{h\in H} f'(y,h)t^h = \sum_{h\in H}\sum_{s=0}^{|\mathbb{F}|-1} a_{h,s} \cdot y^s \cdot t^h = \sum_{s=0}^{|\mathbb{F}|-1}\left(\sum_{h\in H} a_{h,s} \cdot t^h\right) \cdot y^s$$

Assume for a contradiction that for some $s \geq d$, the polynomial $\sum_{h\in H} a_{h,s} \cdot t^h$ is not the identically 0 polynomial, and let s be the largest such index. Since $\sum_{h\in H} a_{h,s} \cdot t^h$ is not identically 0, and its degree is $\leq |H| - 1$, it gives 0 on at most $|H| - 1$ values of $t \in \mathbb{F}$. Hence, the polynomial $f_{(*,t)}(y)$ is of degree $< s$ for at most $|H| - 1$ values of $t \in \mathbb{F}$, which is a contradiction to the assumption that for at least $|H|$ values $t \in \mathbb{F}$, the function $f_{(*,t)} : \mathbb{F} \to \mathbb{F}$ is a univariate polynomial of degree $< d$.

Thus, for every $s \geq d$, the polynomial $\sum_{h\in H} a_{h,s} \cdot t^h$ is the identically 0 polynomial. That is, for every $s \geq d$ and every $h \in H$ we have $a_{h,s} = 0$. Hence, for every $h \in H$, the function $f'_{(*,h)} : \mathbb{F} \to \mathbb{F}$ is a univariate polynomial of degree $< d$. □

We will show that with high probability, at least $\lambda - r_2$ of the planes $M \in \{M_1, \ldots, M_\lambda\}$ are good (where the probability is over $L_1, \ldots, L_\lambda, A$).

Claim 18. *There exists a negligible function $\mu = \mu(\lambda)$ (independent of $i \in [\ell]$) such that with probability $\geq 1 - |\mathbb{F}|\epsilon - \frac{\epsilon_1}{1-\gamma} - \mu$, for at least $\lambda - 2|\mathbb{F}|r - \frac{r_1}{1-\gamma}$ of the indices $j \in [\lambda]$, we have that M_j is good, where $\gamma = \sqrt{\frac{|H|}{|\mathbb{F}|}}$.*

Proof. First note that for $t_1 = 0$,

$$A_i^0(M_j(t_1,t)) = \sum_{h\in H} A_{i-1}^0(M_j(t_1,h))t^h$$

is satisfied trivially (for every $j \in [\lambda]$, and every $t \in \mathbb{F}$), since $A_i^0 \circ M_j(0,*)$ and $A_{i-1}^0 \circ M_j(0,*)$ are the identically 0 function (by the definitions).

For every $t_1 \in \mathbb{F} \setminus \{0\}$, consider the set of points $\{M_j(t_1,0)\}_{j\in[\lambda]}$ and note that this is a set of λ random points in $\mathbb{F}^\ell$, such that the i^{th} coordinate of each of these points is 0 (that is, all other coordinates of all these points are uniformly distributed and independent random variables). Hence, by the computational no-signaling property, there exists a negligible function $\mu_1 = \mu_1(\lambda)$ (independent of $i \in [\ell]$) such that with probability $> 1 - \epsilon - \mu_1$, for at least $\lambda - r$ of the indices $j \in [\lambda]$, the following is satisfied for every $t \in \mathbb{F}$:

$$A_i(M_j(t_1, t)) = \sum_{h \in H} A_{i-1}(M_j(t_1, h))t^h$$

For every $j \in [\lambda]$, denote by E_j the event that the line L_j is in a general position (as a line in $\mathbb{F}^\ell$), that is, its image is not just a single point. Note that event E_j occurs with probability $1 - \frac{1}{|\mathbb{F}|^{\ell-1}}$. Moreover, note that if event E_j occurs, then for every $t \in \mathbb{F}$ we have that $z(t) \notin M_j(t_1, t)$, and hence $A_i^0(M_j(t_1, t)) = A_i(M_j(t_1, t))$ and $A_{i-1}^0(M_j(t_1, t)) = A_{i-1}(M_j(t_1, t))$. Therefore, if event E_j occurs and $A_i(M_j(t_1, t)) = \sum_{h \in H} A_{i-1}(M_j(t_1, h))t^h$ then

$$A_i^0(M_j(t_1, t)) = \sum_{h \in H} A_{i-1}^0(M_j(t_1, h))t^h$$

(and recall that for $t_1 = 0$ this is satisfied trivially).

Recall that each event E_j occurs with probability $1 - \frac{1}{|\mathbb{F}|^{\ell-1}}$, and these events are independent. This, together with our assumption that

$$|\mathbb{F}|^{\ell-1} \geq |H|^{3 \cdot (\ell-1)} \geq |H|^{3 \cdot 3m} \geq \lambda^9$$

(where the latter inequality follows from our assumption that $|H|^m \geq \lambda$), implies that

$$\Pr\left[|\{j : \neg E_j\}| \geq |\mathbb{F}|\right] \leq \binom{\lambda}{|\mathbb{F}|} \cdot \left(\frac{1}{|\mathbb{F}|^{\ell-1}}\right)^{|\mathbb{F}|} \leq \left(\frac{e\lambda}{|\mathbb{F}|^\ell}\right)^{|\mathbb{F}|} \leq \left(\frac{1}{\lambda^8}\right)^{|\mathbb{F}|}.$$

Denote by $\mu_0 = \mu_0(\lambda) = \left(\frac{1}{\lambda^8}\right)^{|\mathbb{F}|}$, and note that $\mu_0 = \mathsf{negl}(\lambda)$. Adding up all this, by the union bound, with probability $> 1 - |\mathbb{F}|\epsilon - |\mathbb{F}|\mu_1 - \mu_0$, for at least $\lambda - |\mathbb{F}|r - |\mathbb{F}| \geq \lambda - 2|\mathbb{F}|r$ of the indices $j \in [\lambda]$, the following is satisfied for every $t_1 \in \mathbb{F}$ and every $t \in \mathbb{F}$:

$$A_i^0(M_j(t_1, t)) = \sum_{h \in H} A_{i-1}^0(M_j(t_1, h))t^h \tag{11}$$

For every $t_2 \in \mathbb{F}$, consider the set of lines $\{M_j(*, t_2)\}_{j \in [\lambda]}$ and note that this is a set of λ random lines, such that for every $L \in \{M_j(*, t_2)\}_{j \in [\lambda]}$, we have $L(0) = z(t_2)$, and L is orthogonal to the i^{th} coordinate. The fact that $z(t_2)$, viewed as a point in D_i, satisfies property $\mathcal{Z}^i(\epsilon_1, r_1)$, together with the computational no-signaling property, implies that there exists a negligible function $\mu_2 = \mu_2(\lambda)$ (independent of $i \in [\lambda]$) such that with probability $\geq 1 - \epsilon_1 - \mu_2$, for at least $\lambda - r_1$ of the indices $j \in [\lambda]$, we have that $A_i^0 \circ M_j(*, t_2) : \mathbb{F} \to \mathbb{F}$ is a univariate polynomial of degree $< 2\ell|H|$.

Since this is true for every $t_2 \in \mathbb{F}$, by Claim 19 below, applied with $\alpha = \epsilon_1 + \mu_2$, we obtain the following for any $\gamma < 1$:

With probability $\geq 1 - \frac{\epsilon_1 + \mu_2}{1-\gamma}$, for at least $\gamma|\mathbb{F}|$ values $t_2 \in \mathbb{F}$ we have that for at least $\lambda - r_1$ of the indices $j \in [\lambda]$, the function $A_i^0 \circ M_j(*, t_2) : \mathbb{F} \to \mathbb{F}$ is a univariate polynomial of degree $< 2\ell|H|$.

Claim 19. *Let $\{V_t\}_{t \in \mathbb{F}}$ be a set of events, such that, for every $t \in \mathbb{F}$, $\Pr[V_t] \geq 1 - \alpha$. Then, for any $\gamma < 1$, with probability of at least $1 - \frac{\alpha}{1-\gamma}$, at least $\gamma|\mathbb{F}|$ events in $\{V_t\}_{t \in \mathbb{F}}$ occur.*

Proof. Let I_t be the characteristic function of the event $\neg V_t$. Let $I = \sum_{t \in \mathbb{F}} I_t$. Thus, $\mathbf{E}[I] \leq \alpha|\mathbb{F}|$. By Markov's inequality, $\Pr[I > (1-\gamma)|\mathbb{F}|] < \alpha/(1-\gamma)$. Thus, with probability of at least $1 - \alpha/(1-\gamma)$, at least $\gamma|\mathbb{F}|$ events in $\{V_t\}_{t \in \mathbb{F}}$ occur. □

Recall that with probability $\geq 1 - \frac{\epsilon_1 + \mu_2}{1-\gamma}$, for at least $\gamma|\mathbb{F}|$ values $t_2 \in \mathbb{F}$ we have that for at most r_1 of the indices $j \in [\lambda]$, the function $A_i^0 \circ M_j(*, t_2) : \mathbb{F} \to \mathbb{F}$ is not a univariate polynomial of degree $< 2\ell|H|$.

Since in a $\{0,1\}$-matrix with $\gamma|\mathbb{F}|$ rows and $[\lambda]$ columns, with at most r_1 ones in each row, there are at most $\frac{\gamma|\mathbb{F}|r_1}{\gamma|\mathbb{F}| - |H|}$ columns with more than $\gamma|\mathbb{F}| - |H|$ ones (otherwise, the total number of ones is $> \gamma|\mathbb{F}|r_1$), this implies that with probability $\geq 1 - \frac{\epsilon_1+\mu_2}{1-\gamma}$, for all but at most $\frac{\gamma|\mathbb{F}|r_1}{\gamma|\mathbb{F}|-|H|}$ indices $j \in [\lambda]$ we have that for at least $|H|$ of the values $t_2 \in \mathbb{F}$, the function $A_i^0 \circ M_j(*, t_2) : \mathbb{F} \to \mathbb{F}$ is a univariate polynomial of degree $< 2\ell|H|$.

Combined with Equation (11), by the union bound, with probability $> 1 - |\mathbb{F}|\epsilon - |\mathbb{F}|\mu_1 - \mu_0 - \frac{\epsilon_1+\mu_2}{1-\gamma}$, for at least $\lambda - 2|\mathbb{F}|r - \frac{\gamma|\mathbb{F}|r_1}{\gamma|\mathbb{F}|-|H|}$ of the indices $j \in [\lambda]$, we have that:

1. For every $t_1 \in \mathbb{F}$ and every $t \in \mathbb{F}$:

$$A_i^0(M_j(t_1, t)) = \sum_{h \in H} A_{i-1}^0(M_j(t_1, h))t^h$$

2. For at least $|H|$ of the values $t_2 \in \mathbb{F}$ the function $A_i^0 \circ M_j(*, t_2) : \mathbb{F} \to \mathbb{F}$ is a univariate polynomial of degree $< 2\ell|H|$.

That is, with probability $\geq 1 - |\mathbb{F}|\epsilon - |\mathbb{F}|\mu_1 - \mu_0 - \frac{\epsilon_1+\mu_2}{1-\gamma}$, for at least $\lambda - 2|\mathbb{F}|r - \frac{\gamma|\mathbb{F}|r_1}{\gamma|\mathbb{F}|-|H|}$ of the indices $j \in [\lambda]$, we have that M_j is good. In particular, for $\gamma = \sqrt{\frac{|H|}{|\mathbb{F}|}}$, we have that with probability $\geq 1 - |\mathbb{F}|\epsilon - |\mathbb{F}|\mu_1 - \mu_0 - \frac{\epsilon_1+\mu_2}{1-\gamma}$, for at least $\lambda - 2|\mathbb{F}|r - \frac{r_1}{1-\gamma}$ of the indices $j \in [\lambda]$, we have that M_j is good.

Setting $\mu = |\mathbb{F}|\mu_1 + \mu_0 + \frac{\mu_2}{1-\gamma}$, we conclude that with probability $\geq 1 - |\mathbb{F}|\epsilon - \frac{\epsilon_1}{1-\gamma} - \mu$, for at least $\lambda - 2|\mathbb{F}|r - \frac{r_1}{1-\gamma}$ of the indices $j \in [\lambda]$, we have that M_j is good. □

By Claim 17, Claim 18 implies that with probability $\geq 1 - |\mathbb{F}|\epsilon - \frac{\epsilon_1}{1-\gamma} - \mu$, at least $\lambda - r_2$ of the indices $j \in [\lambda]$ satisfy that for every $t_2 \in H$, the function $A_{i-1}^0 \circ M_j(*, t_2) : \mathbb{F} \to \mathbb{F}$ is a univariate polynomial of degree $< 2\ell|H|$, (where the probability is over $L_1, \ldots, L_\lambda, A$).

Fix $t_2 = z_i$. Consider the set of lines $\{M_j(*, t_2)\}_{j \in [\lambda]}$ and note that this is a set of λ random lines, such that for every $L \in \{M_j(*, t_2)\}_{j \in [\lambda]}$, we have $L(0) = z(t_2) = z$, and L is orthogonal to the i^{th} coordinate.

Thus, by the above, with probability $\geq 1 - |\mathbb{F}|\epsilon - \frac{\epsilon_1}{1-\gamma} - \mu$, at least $\lambda - r_2$ of the indices $j \in [\lambda]$ satisfy that the function $A_{i-1}^0 \circ M_j(*, t_2) : \mathbb{F} \to \mathbb{F}$ is a univariate polynomial of degree $< 2\ell|H|$. Thus, by the computational no-signaling property, there exists a negligible function $\mu_3 = \mu_3(\lambda)$ (independent of $i \in [\lambda]$) such that the point z, viewed as a point in D_{i-1}, satisfies property $\mathcal{Z}^i(\epsilon_2, r_2)$, with $\epsilon_2 = |\mathbb{F}|\epsilon - \frac{\epsilon_1}{1-\gamma} - \mu^*$, for $\mu^* = \mu + \mu_3$.

This concludes the proof of Claim 13.

□

□

B.2 Proof of Lemma 8

Proof. Fix $z \in D_X$. Let $L_1, \ldots, L_\lambda : \mathbb{F} \to D_X$ be λ random lines, such that for every $L \in \{L_1, \ldots, L_\lambda\}$, we have $L(0) = z$.

Let $L'_1, \ldots, L'_\lambda : \mathbb{F} \to D_X$ be λ random lines, such that for every $L \in \{L'_1, \ldots, L'_\lambda\}$, we have $L(0) = z$. Let $M_1, \ldots, M_\lambda : \mathbb{F}^2 \to D_X$ be λ planes, where $M_j(t_1, t_2) = L_j(t_1) + L'_j(t_2) - z$ (where the addition/substraction are over the vector space $D_X = \mathbb{F}^m$).

Let $S = \{M_j(t_1, t_2)\}_{j \in [\lambda], t_1, t_2 \in \mathbb{F}} \subset D_X$. Let $(\varphi, A) \in_R \mathcal{A}_S$. For any $v \in \mathbb{F}$, define $A^v : S \to \mathbb{F}$ by $A^v(z') = A(z')$ for $z' \neq z$ and $A^v(z) = v$.

We say that M_j is *good* if the following is satisfied:

1. For every $t_1 \in \mathbb{F} \setminus \{0\}$, the function $A \circ M_j(t_1, *) : \mathbb{F} \to \mathbb{F}$ is a univariate polynomial of degree $< m|H|$.
2. For every $t_2 \in \mathbb{F} \setminus \{0\}$, the function $A \circ M_j(*, t_2) : \mathbb{F} \to \mathbb{F}$ is a univariate polynomial of degree $< m|H|$.

For every $j \in [\lambda]$, let E_j denote the event that the lines L_j and L'_j are in general position; i.e., the vectors $L_j(1) - L_j(0)$ and $L'_j(1) - L'_j(0)$ span a linear subspace of dimension two. Note that each event E_j occurs with probability $1 - \frac{1}{|\mathbb{F}|^{\ell-1}}$

Claim 20 below (applied with $f = A \circ M_j$ and $d = m|H|$) implies that if M_j is good and event E_j holds, then there exists $v \in \mathbb{F}$, such that, $A^v \circ L_j = A^v \circ M_j(*, 0) : \mathbb{F} \to \mathbb{F}$ and $A^v \circ L'_j = A^v \circ M_j(0, *) : \mathbb{F} \to \mathbb{F}$ are both univariate polynomials of degree $< m|H|$.

Claim 20. *Let $f : \mathbb{F}^2 \to \mathbb{F}$ be a function. Assume that for every $t_1 \in \mathbb{F} \setminus \{0\}$, the function $f_{(t_1,*)} : \mathbb{F} \to \mathbb{F}$ is a univariate polynomial of degree $< d$, and for every $t_2 \in \mathbb{F} \setminus \{0\}$, the function $f_{(*,t_2)} : \mathbb{F} \to \mathbb{F}$ is a univariate polynomial of degree $< d$, where $d < |\mathbb{F}|$. For any $v \in \mathbb{F}$, define $f^v : \mathbb{F}^2 \to \mathbb{F}$ by $f^v(t_1, t_2) = f(t_1, t_2)$ for $(t_1, t_2) \neq (0, 0)$ and $f^v(0, 0) = v$. Then, there exists $v \in \mathbb{F}$, such that, $f^v_{(0,*)} : \mathbb{F} \to \mathbb{F}$ and $f^v_{(*,0)} : \mathbb{F} \to \mathbb{F}$ are both univariate polynomials of degree $< d$.*

Proof. For every $t_2 \in \mathbb{F} \setminus \{0\}$, the function $f_{(*,t_2)} : \mathbb{F} \to \mathbb{F}$ is a univariate polynomial of degree $< d$. Therefore, there exist $a_1, \ldots, a_d \in \mathbb{F}$, (where $a_1, \ldots, a_d$ are the Lagrange interpolation coefficients), such that for every $t_2 \in \mathbb{F} \setminus \{0\}$, we have $f(0, t_2) = \sum_{t=1}^{d} a_t \cdot f(t, t_2)$. Since $f^v(t_1, t_2) = f(t_1, t_2)$ for $(t_1, t_2) \neq (0, 0)$, this implies that for every $t_2 \in \mathbb{F} \setminus \{0\}$ and every $v \in \mathbb{F}$, we have $f^v(0, t_2) = \sum_{t=1}^{d} a_t \cdot f^v(t, t_2)$.

Let $v = \sum_{t=1}^{d} a_t \cdot f(t, 0)$. Since $f^v(0, 0) = v$, we now have for every $t_2 \in \mathbb{F}$ (including $t_2 = 0$), $f^v(0, t_2) = \sum_{t=1}^{d} a_t \cdot f^v(t, t_2)$. That is, $f^v_{(0,*)} = \sum_{t=1}^{d} a_t \cdot f^v_{(t,*)}$. Since $f^v_{(1,*)}, \ldots, f^v_{(d,*)}$ are identical to $f_{(1,*)}, \ldots, f_{(d,*)}$ and are hence univariate polynomials of degree $< d$, their linear combination $f^v_{(0,*)}$ is also a univariate polynomial of degree $< d$.

The proof now follows from Claim 15, applied on the function f^v (with variables t_1, t_2 switched).

□

We show (in Claim 21 below) that with high probability, for at least $\lambda - 3|\mathbb{F}|r$ of the indices $j \in [\lambda]$, the event E_j holds and the plane M_j is good (where the probability is over $L_1, \ldots, L_k, L'_1, \ldots, L'_k, A$). By Claim 20, this implies that with high probability, for at least $\lambda - 3|\mathbb{F}|r$ of the indices $j \in [\lambda]$, there exists $v \in \mathbb{F}$ (which may depend on j) such that $A^v \circ L_j = A^v \circ M_j(*, 0) : \mathbb{F} \to \mathbb{F}$ and $A^v \circ L'_j = A^v \circ M_j(0, *) : \mathbb{F} \to \mathbb{F}$ are both univariate polynomials of degree $< m|H|$ (where the probability is over $L_1, \ldots, L_k, L'_1, \ldots, L'_k, A$).

Claim 21. *There exists a negligible function $\mu = \mathsf{negl}(\lambda)$ such that with probability $\geq 1 - 2|\mathbb{F}|\epsilon - \mu$, for at least $\lambda - 3|\mathbb{F}|r$ of the indices $j \in [\lambda]$, we have that event E_j holds and M_j is good.*

Proof. Recall that each event E_j occurs with probability $1 - \frac{1}{|\mathbb{F}|^{\ell-1}}$, and these events are independent.

This, together with our assumption that

$$|\mathbb{F}|^{\ell-1} \geq |H|^{3\cdot(\ell-1)} \geq |H|^{3\cdot 3m} \geq \lambda^9$$

(where the latter inequality follows from our assumption that $|H|^m \geq \lambda$), implies that

$$\Pr\left[|\{j : \neg E_j\}| \geq |\mathbb{F}|\right] \leq \binom{\lambda}{|\mathbb{F}|} \cdot \left(\frac{1}{|\mathbb{F}|^{\ell-1}}\right)^{|\mathbb{F}|} \leq \left(\frac{e\lambda}{|\mathbb{F}|^{\ell}}\right)^{|\mathbb{F}|} \leq \left(\frac{1}{\lambda^8}\right)^{|\mathbb{F}|}.$$

Denote by $\mu_0 = \mu_0(\lambda) = \left(\frac{1}{\lambda^8}\right)^{|\mathbb{F}|}$, and note that $\mu_0 = \mathsf{negl}(\lambda)$.

For every $t_1 \in \mathbb{F} \setminus \{0\}$, consider the set of lines $\{M_j(t_1, *)\}_{j\in[\lambda]}$ and note that this is a set of λ random lines in D_X. Hence, by the computational no signaling, there exists a negligible function $\mu_1 = \mu_1(\lambda)$ such that with probability $> 1 - \epsilon - \mu_1$, for at least $\lambda - r$ of the indices $j \in [\lambda]$, we have that $A \circ M_j(t_1, *) : \mathbb{F} \to \mathbb{F}$ is a univariate polynomial of degree $< m|H|$.

For every $t_2 \in \mathbb{F} \setminus \{0\}$, consider the set of lines $\{M_j(*, t_2)\}_{j\in[\lambda]}$ and note that this is a set of λ random lines in D_X. Hence, by the computational no-signaling property, with probability $> 1 - \epsilon - \mu_1$, for at least $\lambda - r$ of the indices $j \in [\lambda]$, we have that $A \circ M_j(*, t_2) : \mathbb{F} \to \mathbb{F}$ is a univariate polynomial of degree $< m|H|$.

Adding up these facts, by the union bound, we obtain that with probability $\geq 1 - 2|\mathbb{F}|\epsilon - 2|\mathbb{F}|\mu_1$, for at least $\lambda - 2|\mathbb{F}|r$ of the indices $j \in [\lambda]$, we have that:

1. For every $t_1 \in \mathbb{F} \setminus \{0\}$, $A \circ M_j(t_1, *) : \mathbb{F} \to \mathbb{F}$ is a univariate polynomial of degree $< m|H|$.

2. For every $t_2 \in \mathbb{F} \setminus \{0\}$, $A \circ M_j(*, t_2) : \mathbb{F} \to \mathbb{F}$ is a univariate polynomial of degree $< m|H|$.

That is, with probability $\geq 1 - 2|\mathbb{F}|\epsilon - 2|\mathbb{F}|\mu_1$, for at least $\lambda - 2|\mathbb{F}|r$ of the indices $j \in [\lambda]$, we have that M_j is good.

By setting $\mu = 2|\mathbb{F}|\mu_1 + \mu_0$ we conclude that with probability $\geq 1 - 2|\mathbb{F}|\epsilon - \mu$ for at least $\lambda - 3|\mathbb{F}|r$ of the indices $j \in [\lambda]$, we have that event E_j holds and M_j is good, as desired.

□

So far we proved that with probability $\geq 1 - 2|\mathbb{F}|\epsilon - \mu$ it holds that for at least $\lambda - 3|\mathbb{F}|r$ of the indices $j \in [\lambda]$ there exists a value $v \in \mathbb{F}$ (which may depend on j), such that $A^v \circ L_j : \mathbb{F} \to \mathbb{F}$ and $A^v \circ L'_j : \mathbb{F} \to \mathbb{F}$ are univariate polynomials of degree $< m|H|$.

To conclude the proof, we need to prove that there exists a negligible function μ' such that with probability $\geq 1 - 2|\mathbb{F}|\epsilon - \mu'$ there exists a single value $v \in \mathbb{F}$ such that for at least $\lambda - r'$ of the indices $j \in [\lambda]$ it holds that $A^v \circ L_j$ is univariate polynomials of degree $< m|H|$, where $r' = 30|\mathbb{F}|r$.

To this end, denote by E the event that indeed there exists $v \in \mathbb{F}$, such that for at least $\lambda - r'$ of the indices $j \in [\lambda]$, $A^v \circ L_j : \mathbb{F} \to \mathbb{F}$ is a univariate polynomial of degree $< m|H|$. We need to prove that there exists a negligible function μ' such that $\Pr[E] \geq 1 - 2|\mathbb{F}|\epsilon - \mu'$ (where the probability is over $L_1, \ldots, L_{2\lambda}, A$).

Denote by E' the event that for at least $\lambda - 3|\mathbb{F}|r$ of the indices $j \in [\lambda]$, there exists $v \in \mathbb{F}$ (that may depend on j), such that both $A^v \circ L_j : \mathbb{F} \to \mathbb{F}$ and $A^v \circ L'_j : \mathbb{F} \to \mathbb{F}$ are univariate polynomials of degree $< m|H|$. By Claim 21,

$$\Pr[E'] \geq 1 - 2|\mathbb{F}|\epsilon - \mu.$$

Claim 22. $\Pr[E' \mid \neg E] = \mathsf{negl}(\lambda)$

Proof. In what follows, to simplify notation, we denote the random lines $L'_1, \ldots, L'_\lambda$ by $L_{\lambda+1}, \ldots, L_{2\lambda}$, and consider the 2λ lines $L_1, \ldots, L_{2\lambda}$. Note that these are 2λ random lines such that for every $j \in [2\lambda]$, it holds that $L_j : \mathbb{F} \to D_X$ and $L_j(0) = z$.

For every $v \in \mathbb{F}$, let J_v be the set of indices $j \in [2\lambda]$ such that $A^v \circ L_j : \mathbb{F} \to \mathbb{F}$ is a univariate polynomial of degree $< m|H|$. Note that for every $v \neq v' \in \mathbb{F}$,

$$J_v \cap J_{v'} = \emptyset.$$

If event $\neg E$ occurs then for every $v \in \mathbb{F}$,

$$|J_v| < 2\lambda - r'.$$

Denote by J the largest set J_v and by $\bar{J}$ the complement of J in $[2\lambda]$. Thus, if event $\neg E$ occurs then $|\bar{J}| > r'$.

Note that given the sets $\{J_v\}_{v \in \mathbb{F}}$, the probability that E' occurs is the probability that when partitioning $[2\lambda]$ randomly into λ pairs, for at least $\lambda - 3|\mathbb{F}|r$ pairs the two indices in the pair are in the same set J_v. Assuming that $|\bar{J}| > r'$, this probability can be bounded by $2^{-|\mathbb{F}|r}$ by the following argument:

Choose the partition as follows: First choose randomly $\lambda' = r'/2 = 15|\mathbb{F}|r$ different indices $j_1, \ldots, j_{\lambda'}$ in $\bar{J}$. Match the indices $j_1, \ldots, j_{\lambda'}$ one by one, each to a random index in $[2\lambda]$ that was still not chosen. Say that $j_t \in \{j_1, \ldots, j_{\lambda'}\}$ is good if it was matched to an index in a set J_v such that $j_t \in J_v$. Finally, extend the partial partition randomly into a partition of $[2\lambda]$ into λ pairs. Note that the probability for an index j_t to be good is at most $\frac{\lambda}{2\lambda - r'} < 0.51$, independently of all previous choices of indices. Thus, the probability that at least $\lambda' - 3|\mathbb{F}|r$ indices $j_t \in \{j_1, \ldots, j_{\lambda'}\}$ are good is at most

$$\binom{\lambda'}{3|\mathbb{F}|r} \cdot 0.51^{\lambda' - 3|\mathbb{F}|r} \leq \left(\frac{\lambda' \cdot e}{3|\mathbb{F}|r}\right)^{3|\mathbb{F}|r} \cdot 0.51^{12|\mathbb{F}|r} = \left((5e)^3 \cdot 0.51^{12}\right)^{|\mathbb{F}|r} \leq 0.8^{|\mathbb{F}|r} = \mathsf{negl}(\lambda),$$

where the first inequality follows from the standard inequality that $\binom{n}{k} \leq \left(\frac{n \cdot e}{k}\right)^k$, and the second to last inequality follows from basic calculations.

Thus, $\Pr[E' \mid \neg E] = \mathsf{negl}(\lambda)$. □

We can now bound,

$$1 - 2|\mathbb{F}|\epsilon - \mu \leq \Pr[E'] \leq \Pr[E' \mid \neg E] + \Pr[E]$$

Thus,

$$\Pr[E] > 1 - 2|\mathbb{F}|\epsilon - \mu - \Pr[E' \mid \neg E].$$

Let $\mu' = \mu + \Pr[E' \mid \neg E]$. Thus,

$$\Pr[E] > 1 - 2|\mathbb{F}|\epsilon - \mu',$$

and by Claim 22, μ' is indeed a negligible function, as desired. □

B.3 Proof of Lemma 9

Proof. Since φ is polynomially-sized, it suffices to fix any $i_1, i_2, i_3 \in H^m$ and $b_1, b_2, b_3 \in \{0, 1\}$ and view i_1, i_2, i_3 as points in D_X. Let $L_{1,1}, \ldots, L_{1,\lambda} : \mathbb{F} \to D_X$ be λ random lines, such that for every $L \in \{L_{1,1}, \ldots, L_{1,\lambda}\}$ we have $L(0) = i_1$. Let $L_{2,1}, \ldots, L_{2,\lambda} : \mathbb{F} \to D_X$ be λ random lines, such that for every $L \in \{L_{2,1}, \ldots, L_{2,\lambda}\}$ we have $L(0) = i_2$. Let $L_{3,1}, \ldots, L_{3,\lambda} : \mathbb{F} \to D_X$ be λ random lines, such that for every $L \in \{L_{3,1}, \ldots, L_{3,\lambda}\}$ we have $L(0) = i_3$. Let

$$S^X = \{L_{1,j}(t), L_{2,j}(t), L_{3,j}(t)\}_{j \in [\lambda], t \in \mathbb{F}} .$$

Let $z = (i_1, i_2, i_3, b_1, b_2, b_3) \in H^\ell$. Let $L^1_{b_1,b_2,b_3}, \ldots, L^\lambda_{b_1,b_2,b_3} : \mathbb{F} \to D_0$ be λ random lines such that for every $j \in [\lambda]$ the following holds:

1. $L^j_{b_1,b_2,b_3}(0) = z$.
2. $L_{1,j} : \mathbb{F} \to D_X$ is the restriction of $L^j_{b_1,b_2,b_3}$ to coordinates $\{1, \ldots, m\}$.
3. $L_{2,j} : \mathbb{F} \to D_X$ is the restriction of $L^j_{b_1,b_2,b_3}$ to coordinates $\{m + 1, \ldots, 2m\}$.
4. $L_{3,j} : \mathbb{F} \to D_X$ is the restriction of $L^j_{b_1,b_2,b_3}$ to coordinates $\{2m + 1, \ldots, 3m\}$.

Let

$$S^0_{b_1,b_2,b_3} = \left\{L^j_{b_1,b_2,b_3}(t)\right\}_{j \in [\lambda], t \in \mathbb{F}} .$$

Let $S^0 = \{S^0_{b_1,b_2,b_3}\}_{b_1,b_2,b_3 \in \{0,1\}}$ and let $S = S^0 \cup S^X \subset D$. Let $(\varphi, A) \in_R \mathcal{A}_S$. We denote $A = A_X \cup A_0$, where A_X corresponds to the answers corresponding to the queries in S^X and A_0 are the answers corresponding to the queries in S^0.

In what follows, for any $i \in D_X$ and $v \in \mathbb{F}$, we define $A_X^{i \to v} : S^X \to \mathbb{F}$ by $A_X^{i \to v}(i') = A_X(i')$ for $i' \neq i$ and $A_X^{i \to v}(i) = v$.

Claim 23. *There exists a negligible function μ such that with probability $\geq 1 - 7\ell|\mathbb{F}|\epsilon - \mu$, there exist $v_1, v_2, v_3 \in \mathbb{F}$ such that for at least $\lambda - 9\ell|\mathbb{F}|r$ of the indices $j \in [\lambda]$, the following is satisfied:*

1. $A_X^{i_1 \to v_1} \circ L_{1,j} : \mathbb{F} \to \mathbb{F}$ *is a univariate polynomial of degree* $< m|H|$.
2. $A_X^{i_2 \to v_2} \circ L_{2,j} : \mathbb{F} \to \mathbb{F}$ *is a univariate polynomial of degree* $< m|H|$.
3. $A_X^{i_3 \to v_3} \circ L_{3,j} : \mathbb{F} \to \mathbb{F}$ *is a univariate polynomial of degree* $< m|H|$.
4. $\phi(i_1, i_2, i_3, b_1, b_2, b_3) \cdot (v_1 - b_1) \cdot (v_2 - b_2) \cdot (v_3 - b_3) = 0.$

By computational no-signaling requirement, Claim 23 immediately implies Lemma 9. Thus, in the remaining of the proof we focus on proving Claim 23.

Let $A^0_0 \circ L^j_{b_1,b_2,b_3} : \mathbb{F} \to \mathbb{F}$ be the function defined by $A^0_0 \circ L^j_{b_1,b_2,b_3}(0) = 0$ and $A^0_0 \circ L^j_{b_1,b_2,b_3}(t) = A_0 \circ L^j_{b_1,b_2,b_3}(t)$ for every $t \neq 0$. By Lemma 7, together with the computational no-signaling property, there exists a negligible function $\mu_1 = \mu_1(\lambda)$, such that with probability $\geq 1 - 6\ell|\mathbb{F}|\epsilon - \mu_1$, for at least $\lambda - 8\ell|\mathbb{F}|r$ of the indices $j \in [\lambda]$, we have that

$$A^0_0 \circ L^j_{b_1,b_2,b_3} : \mathbb{F} \to \mathbb{F}$$

is a univariate polynomial of degree $< 2\ell|H|$.

By Lemma 8, together with the computational no-signaling property, there exists a negligible function $\mu_2 = \mu_2(\lambda)$, such that the following holds:

1. With probability $\geq 1 - 2|\mathbb{F}|\epsilon - \mu_2$, there exists $v_1 \in \mathbb{F}$, such that, for at least $\lambda - 30|\mathbb{F}|r$ of the indices $j \in [\lambda]$, we have that $A_X^{i_1 \to v_1} \circ L_j^1 : \mathbb{F} \to \mathbb{F}$ is a univariate polynomial of degree $< m|H|$.
2. With probability $\geq 1 - 2|\mathbb{F}|\epsilon - \mu_2$, there exists $v_2 \in \mathbb{F}$, such that, for at least $\lambda - 30|\mathbb{F}|r$ of the indices $j \in [\lambda]$, we have that $A_X^{i_2 \to v_2} \circ L_j^2 : \mathbb{F} \to \mathbb{F}$ is a univariate polynomial of degree $< m|H|$.
3. With probability $\geq 1 - 2|\mathbb{F}|\epsilon - \mu_2$, there exists $v_3 \in \mathbb{F}$, such that, for at least $\lambda - 30|\mathbb{F}|r$ of the indices $j \in [\lambda]$, we have that $A_X^{i_3 \to v_3} \circ L_j^3 : \mathbb{F} \to \mathbb{F}$ is a univariate polynomial of degree $< m|H|$.

For every $t \in \mathbb{F} \setminus \{0\}$, consider the set of points $\left\{ L^j_{b_1,b_2,b_3}(t) \right\}_{j \in [\lambda]}$ and note that this is a set of λ random points in D_0. Each point $L^j_{b_1,b_2,b_3}(t) \in \mathbb{F}^\ell$ can be written as

$$L^j_{b_1,b_2,b_3}(t) = \left(L_{1,j}(t), L_{2,j}(t), L_{3,j}(t), \left(L^j_{b_1,b_2,b_3}(t)\right)_{\ell-2}, \left(L^j_{b_1,b_2,b_3}(t)\right)_{\ell-1}, \left(L^j_{b_1,b_2,b_3}(t)\right)_{\ell} \right) \in (\mathbb{F}^m)^3 \times \mathbb{F}^3 = \mathbb{F}^\ell$$

where $\left(L^j_{b_1,b_2,b_3}(t)\right)_{\ell-2}, \left(L^j_{b_1,b_2,b_3}(t)\right)_{\ell-1}, \left(L^j_{b_1,b_2,b_3}(t)\right)_{\ell}$ are the last 3 coordinates of $L^j_{b_1,b_2,b_3}(t)$. By the computational no-signaling condition, there exists a negligible function $\mu_3 = \mu_3(\lambda)$ such that for every $t \in \mathbb{F} \setminus \{0\}$, with probability $> 1 - \epsilon - \mu_3$, for at least $\lambda - r$ of the indices $j \in [\lambda]$, we have

$$\begin{aligned} A_0\left(L^j_{b_1,b_2,b_3}(t)\right) = \hat{\phi}\left(L^j_{b_1,b_2,b_3}(t)\right) & \\ \cdot \left(A_X(L_{1,j}(t)) - \left(L^j_{b_1,b_2,b_3}(t)\right)_{\ell-2} \right) & \\ \cdot \left(A_X(L_{2,j}(t)) - \left(L^j_{b_1,b_2,b_3}(t)\right)_{\ell-1} \right) & \\ \cdot \left(A_X(L_{3,j}(t)) - \left(L^j_{b_1,b_2,b_3}(t)\right)_{\ell} \right) & \end{aligned}$$

For every $j \in [\lambda]$ denote by E_j the event that for every $w \in \{1,2,3\}$ the line $L_{w,j}$ is in a general position (as a line in $\mathbb{F}^m$); i.e., it's image is not a single point. Note that event E_j occurs with probability at least $1 - \frac{3}{|\mathbb{F}|^{m-1}}$.

Moreover, note that if event E_j holds then for every $t \in \mathbb{F} \setminus \{0\}$ it holds that $L_{1,j}(t) \neq i_1$ and $L_{2,j}(t) \neq i_2$ and $L_{3,j}(t) \neq i_3$. In particular, $L^j_{b_1,b_2,b_3}(t) \neq z$. Thus, if the equation above holds and event E_j holds, then for every $v_1, v_2, v_3 \in \mathbb{F}$,

$$\begin{aligned} A_0^0\left(L^j_{b_1,b_2,b_3}(t)\right) = \; & \hat{\phi}\left(L^j_{b_1,b_2,b_3}(t)\right) \\ & \cdot \left(A_X^{i_1 \to v_1}(L_{1,j}(t)) - \left(L^j_{b_1,b_2,b_3}(t)\right)_{\ell-2} \right) \\ & \cdot \left(A_X^{i_2 \to v_2}(L_{2,j}(t)) - \left(L^j_{b_1,b_2,b_3}(t)\right)_{\ell-1} \right) \\ & \cdot \left(A_X^{i_3 \to v_3}(L_{3,j}(t)) - \left(L^j_{b_1,b_2,b_3}(t)\right)_{\ell} \right) \end{aligned}$$

Our assumption that $|H|^m \geq \lambda$, together with the assumption that $|\mathbb{F}| \geq |H|^3$, implies that

$$|\mathbb{F}|^{m-1} \geq \lambda^2,$$

which implies that

$$\Pr\left[|\{j : \neg E_j\}| \geq |\mathbb{F}| \right] \leq \binom{\lambda}{|\mathbb{F}|} \cdot \left(\frac{3}{|\mathbb{F}|^{m-1}} \right)^{|\mathbb{F}|} \leq \left(\frac{3e\lambda}{|\mathbb{F}|^m} \right)^{|\mathbb{F}|} \leq \left(\frac{3}{\lambda} \right)^{|\mathbb{F}|}.$$

Let $\mu_4 = \left(\frac{3}{\lambda}\right)^{|\mathbb{F}|}$. Adding up all this, by the union bound, we obtain that with probability $\geq 1 - \epsilon'$, where

$$\epsilon' = 6\ell|\mathbb{F}|\epsilon + \mu_1 + 3(2|\mathbb{F}|\epsilon + \mu_2) + |\mathbb{F}|(\epsilon + \mu_3) + \mu_4 \leq 7\ell|\mathbb{F}|\epsilon + \mu$$

(and where $\mu = \mu_1 + 3\mu_2 + |\mathbb{F}|\mu_3 + \mu_4$) there exist $v_1, v_2, v_3 \in \mathbb{F}$, such that, for at least $\lambda - r'$ of the indices $j \in [\lambda]$, where

$$r' = 8\ell|\mathbb{F}|r + 30|\mathbb{F}|r + r + |\mathbb{F}| \leq 9\ell|\mathbb{F}|r,$$

the following is satisfied (where the probability is over $L_1, \ldots, L_\lambda, A$):

1. $A_0^0 \circ L^j_{b_1,b_2,b_3} : \mathbb{F} \to \mathbb{F}$ is a univariate polynomial of degree $< 3\ell|H|$.
2. $A_X^{i_1 \to v_1} \circ L_{1,j} : \mathbb{F} \to \mathbb{F}$ is a univariate polynomial of degree $< m|H|$.
3. $A_X^{i_2 \to v_2} \circ L_{2,j} : \mathbb{F} \to \mathbb{F}$ is a univariate polynomial of degree $< m|H|$.
4. $A_X^{i_3 \to v_3} \circ L_{3,j} : \mathbb{F} \to \mathbb{F}$ is a univariate polynomial of degree $< m|H|$.
5. For every $t \in \mathbb{F} \setminus \{0\}$,

$$\begin{aligned} A_0^0\left(L^j_{b_1,b_2,b_3}(t)\right) = \hat{\phi}\left(L^j_{b_1,b_2,b_3}(t)\right) & \\ \cdot \left(A_X^{i_1 \to v_1}(L_{1,j}(t)) - \left(L^j_{b_1,b_2,b_3}(t)\right)_{\ell-2}\right) & \\ \cdot \left(A_X^{i_2 \to v_2}(L_{2,j}(t)) - \left(L^j_{b_1,b_2,b_3}(t)\right)_{\ell-1}\right) & \\ \cdot \left(A_X^{i_3 \to v_3}(L_{3,j}(t)) - \left(L^j_{b_1,b_2,b_3}(t)\right)_{\ell}\right) & \end{aligned}$$

Note that since both sides of the equation are polynomials of degree $< |\mathbb{F}|$ in the variable t, the equation must be satisfied for $t = 0$ as well. Substituting $t = 0$, since $L^j_{b_1,b_2,b_3}(0) = z = (i_1, i_2, i_3, b_1, b_2, b_3)$, we have

$$0 = \phi(z) \cdot (v_1 - b_1) \cdot (v_2 - b_2) \cdot (v_3 - b_3),$$

as desired.

□

ECCC ISSN 1433-8092

http://eccc.hpi-web.de