Verifiable Outsourcing of Computation

Ron D. Rothblum

Under the Supervision of Professor Oded Goldreich
Department of Computer Science and Applied Mathematics
Weizmann Institute of Science

Submitted for the degree of Doctor of Philosophy
to the Scientific Council of the Weizmann Institute of Science

March 2015

Dedicated to the memory of my father, Uri Rothblum

Abstract

We study the question of designing protocols for allowing a computationally weak client to outsource its computations to a powerful but untrusted server. Since the client does not trust the server, the latter must convince the client that the result of the computation is correct. The two crucial points are that (1) verifying the correctness must be much easier than directly performing the computation, and (2) the overhead on the server's side must be minimal. Our main results are:

1. **(Almost) Linear-Time Verification:** We show a general-purpose *single-round* protocol that allows the client to outsource the computation of *any* function computable in (arbitrarily large) polynomial-time such that the correctness of the computation can be verified in (almost) linear-time, with only a polynomial overhead for the server. The security of our protocol relies on the security of a standard cryptographic primitive.

2. **Sublinear-Time Verification:** In some settings even linear-time verification may not be feasible. We initiate the study of *non-interactive* proofs of proximity. These allow the client to verify a given statement in sub-linear time, using a short (sublinear length) explicitly given certificate (i.e., proof) from the server. Since the client cannot even read its entire input, following the property testing literature, we only guarantee an approximate answer. This notion can be viewed as the NP (or more accurately MA) analogue of property testing.

Acknowledgements

First and foremost, I would like to thank my advisor, Oded Goldreich, for his support and guidance. Working with Oded has been a life-changing experience and has shaped my view of theoretical computer science and beyond. Oded is an amazing mentor and I feel extremely fortunate to have had the opportunity to learn from him. I admire Oded for his extraordinary clarity of thought, and his unique ability to communicate these thoughts to others. What strikes me most about Oded, beyond his uncanny intelligence, are his friendliness and sense of humor. Our meetings in Dizi were filled with stories and jokes that I will greatly miss.

I would like to thank Moni Naor and Adi Shamir for serving as my PhD committee, and for their advice and insights throughout the past few years.

During graduate school, I spent two extremely enjoyable and fruitful summers as an intern, the first at MSR New England and the second at MSR Silicon Valley. I thank Yael Kalai and Omer Reingold for hosting me during these internships and for the friendly and fun atmosphere that they created. I thank Yael for introducing me to the exciting problem of delegating computation, which turned out to be at the core of my research and this thesis. I thank Omer for broadening my horizons beyond cryptography and working with me on problems in pseudorandomness and in data structures.

One of the great things about doing research is being able to collaborate with exceptionally talented scientists. I thank my co-authors, Gil Cohen, Ivan Damgård, Oded Goldreich, Tom Gur, Yuval Ishai, Yael Kalai, Jonas Kölker, Raghu Meka, Peter Bro Miltersen, Ran Raz, Omer Reingold, Guy Rothblum and Udi Wieder for sharing their knowledge with me and for their collaboration.

Although we Rothblums are geographically far apart, we somehow manage to remain extremely close. I thank Alex, Modi and little Uriel for the joy that they bring into my life with every video call and every visit. I thank my brother Guy for his constant support and encouragement. I also thank Guy for anticipating my research interests years in advance and developing fascinating lines of research that ended up being at the core of my research and this thesis.

To my mom and dad, Naomi and Uri, my gratitude is beyond expression. They have set an example for parenthood to which I can only aspire. Abba, it is inconceivable to me that you will never read these words. You are forever in my thoughts and in my heart. This thesis is lovingly dedicated to your memory.

Lastly, I thank my wife, Lucy, for always believing in me, for her boundless enthusiasm and energy and for her unwavering support even in the worst of times. Mostly, I thank her for being my best friend. Finally, I thank little Yotam who has recently entered our lives – he is undoubtedly my best result!

Contents

Chapter 1

Introduction

The past few years have seen a dramatic shift in the nature of computing. The classical paradigm of "personal computing" is being challenged by the fact that vast amounts of computation are being executed on massive datasets. Nowadays more and more computations, which used to be performed locally, are being outsourced to the "cloud" - an external service provider that has extremely powerful hardware and can be paid to run expensive computations. This exciting shift is coupled with the fact that personal computing devices (such as smartphones and tablets) have become (relatively) computationally weaker, and must use the power of the cloud to execute expensive computational tasks.[1]

This emerging paradigm of "cloud computing" gives rise to new security concerns, due to the fact that a potentially untrusted party (i.e., the cloud) is involved in our computations. Perhaps the most fundamental security concern, if we do not trust the cloud, is that we cannot just assume that the result of the outsourced computation is correct. For example, what is to stop the cloud service provider from neglecting to run the computation all-together and simply providing an arbitrary result (thereby saving itself a lot of resources)?[2] In addition, in some natural scenarios, the cloud service provider may an ulterior motive to convince the client of an incorrect result.

The focus of this thesis is introducing and exploring models and techniques that allow a client to securely outsource its computation to an untrusted server in such a way that the client can *efficiently* verify the correctness of the result.

[1] For example, "Siri ... uses a natural language user interface to answer questions, make recommendations, and perform actions by *delegating requests to a set of web services*" - Wikipedia entry for "Siri", the digital personal assistant software used by Apple.

[2] This attack is even more profound in case the computation has an a priori expected result. Consider for instance the SETI (search for extraterrestrial intelligence) program. Here, a huge amount of electromagnetic radiation data is processed for signs of transmission from civilizations from other worlds. A cloud service provider asked to process the data could save itself a lot of resources (without arousing too much suspicion) by just answering negatively.

1.1 Verifiably Outsourcing Computation

Suppose that a weak computational client as above wishes to outsource the computation of a function f, on a given input x, to a powerful server (i.e., the cloud). The client sends x to the server[3] and suppose that the server claims that $f(x) = y$. Since the client does not trust the server, in addition to sending y, the server must convince the client that indeed $y = f(x)$.

One naive solution to this problem is for the client to simply check that $y = f(x)$ by computing $f(x)$ directly. However, this solution defeats the entire purpose of outsourcing computation since the client's running time is equivalent to the time that it takes to directly compute f without any help from the server. Hence, we seek solutions in which the verification process is *much* faster than directly computing f.

On top of the efficient verification, an additional requirement, which seems necessary for any practical application, is that the server should not work too hard in order to *prove* that $y = f(x)$. That is, the running time of the server should not be much longer than the time that it takes to compute f.

To summarize, we seek a protocol for verifying the correctness of a computation that is *doubly efficient*. That is, the protocol should have both

1. *Super efficient verification* (i.e., the client's running time is much shorter than the time that it takes to directly compute f); and

2. *Efficient proving* (i.e., the server's running time is proportional to the time that it takes to compute f).

In addition to the double efficiency requirement, we will aim for solutions in which the number of rounds of communication is minimal. Ideally, the entire communication should consist of a single round - the client sends x (and possibly some additional short string) and the server sends back the result y and a "proof" π that convinces the client that indeed $y = f(x)$.

Comparison with Interactive Proofs. The problem of verifiably outsourcing computation is closely related to the notion of an *interactive proof*, introduced by Goldwasser, Micali and Rackoff [GMR89]. In contrast to our setting, in an interactive proof, the verifier is allowed to run in (arbitrarily large) polynomial-time and the prover's running time is typically not restricted. In contrast, the requirement in our setting is much stronger: the verifier should be *super* efficient (e.g., linear-time) and the prover should be relatively efficient (e.g., polynomial-time). The study of such strong interactive proofs (which are

[3]The fact that the client shares its input x with the server raises an additional security concern. Namely, the fact that the client's private information is revealed to the server. In this thesis our focus is only on the problem of guaranteeing *correctness* and we ignore the privacy aspect. Still, we note that solutions that guarantee *correctness* (as described in this thesis) can be combined with cryptographic tools such as *fully-homomorphic encryption* [RAD78, Gen09] to simultaneously yield both correctness and privacy.

inherently limited to tractable languages since the prover is efficient), originates in the work of Goldwasser, Kalai and Rothblum [GKR08].

1.2 Our Results

We start with a brief descriptions of our main results. More details will follow in subsequent subsections.

(Almost) Linear-Time Verification. Together with Kalai and Raz [KRR13a, KRR14], we designed a general-purpose *single-round* protocol that allows a verifier to outsource the computation of *any* function. For functions that can be computed in (arbitrarily large) polynomial-time, the verifier in our protocol runs in almost *linear-time* (i.e., essentially not doing much more than just reading its input) and the prover runs in polynomial-time. The verifier is guaranteed that the answer claimed by the prover is indeed correct (with overwhelming probability), unless the prover can break the security of a standard cryptographic primitive.

Sublinear-Time Verification. In some settings, when the input itself is huge, even linear-time verification may not be feasible for the verifier. Since the verifier cannot even read the entire input, following the property testing literature, we only require it to reject inputs that are far from being valid.

In [GR15b], together with Gur, we initiated the study of *non-interactive* proofs of proximity. Here, the verifier is able to verify that a given statement is *close* to a true statement, using a short (sublinear length) explicitly given certificate (i.e., proof), and a sublinear number of queries to its input. Such proof-systems can be viewed as the NP analogue of *property testing*. We explored both the power and the limitations of non-interactive proofs of proximity and, among other results, showed that such proof-systems can be exponentially stronger than property testers, but are exponentially weaker than *interactive* proofs of proximity, which were recently studied by Rothblum, Vadhan and Wigderson [RVW13].

1.2.1 Linear-Time Verification for P

Together with Kalai and Raz [KRR13a, KRR14], we constructed a one-round protocol for verifiably outsourcing the computation of every function computable in time $t = t(n)$, such that the running time of the verifier is $n \cdot \mathsf{polylog}(t)$ and the running time of the prover $\mathsf{poly}(t)$. In particular, for every language in P we obtain almost linear time verification. Our construction relies on the existence of a (sub-exponentially secure) computational private information retrieval (PIR) scheme.

Our proof is based on a curious connection between the problem of *verifiable outsourcing of computation* and a model of *multi-prover interactive proofs that are sound against no-signaling (cheating) strategies*. This model was studied in the context of multi-prover

interactive proofs with provers that share quantum entanglement, and is motivated by the physical principle that information cannot travel faster than light.

For any language computable in time $t = t(n)$, we constructed a multi-prover interactive proof (MIP) that is sound against no-signaling strategies, where the running time of the provers is $\mathsf{poly}(t)$, the number of provers is $\mathsf{polylog}(t)$, and the running time of the verifier is $n \cdot \mathsf{polylog}(t)$. In particular, this shows that the class of languages that have polynomial-time MIPs that are sound against no-signaling strategies, is exactly EXP. Previously, this class was only known to contain PSPACE.

To convert our MIP into a one-round verifiable delegation scheme, we used the method suggested by Aiello *et al.* [ABOR00], which makes use of a PIR scheme. This method lacked a proof of security. We proved that this method is secure assuming the underlying MIP is secure against no-signaling provers. See Chapter 2 for details.

1.2.2 Non-Interactive Proofs of Proximity

The protocol discussed above allows to verify the correctness of any polynomial-time computation in (almost) linear-time. In some settings, however, even linear-time verification may not be feasible. For example, consider a data analyst performing some statistical computation on a huge dataset x that is publicly available on the internet. A fundamental problem that arises when outsourcing computation in this setting is that the analyst cannot even read her entire input and therefore, the server can easily cheat by providing an answer corresponding to some dataset x' that only differs from x on a few bits.

Following the property testing literature, we address this problem by relaxing the requirement so that the analyst is only assured of the *proximity* of the statement to a correct one. Indeed, such an approximate answer can suffice for many applications (e.g., for statistical analysis). This model, known as *interactive proofs of proximity* (IPP) was introduced by Ergün, Kumar and Rubinfeld [EKR04] and was recently studied by Rothblum, Vadhan and Wigderson [RVW13], who showed a (multi-round) IPP for a large class of functions, in which the verifier runs in *sublinear-time*.

Together with Gur [GR15b], we initiated a study of *non-interactive* proofs of proximity. These proof-systems consist of a verifier that wishes to ascertain the validity of a given statement, using a short (sublinear length) explicitly[4] given proof, and a sublinear number of queries to its input. Since the verifier cannot even read the entire input, we only require it to reject inputs that are far from being valid. Thus, the verifier is only assured of the proximity of the statement to a correct one. Such proof-systems can be viewed as the NP (or more accurately MA) analogue of *property testing*.

We explored both the power and limitations of non-interactive proofs of proximity. We showed that such proof-systems can be exponentially stronger than property testers, but are exponentially weaker than *interactive* proofs of proximity. In addition, we showed a natural problem that has a full and (almost) tight multiplicative trade-off between the length of the proof and the verifier's query complexity. See Chapter 3 for details.

[4] In contrast, in the related model of *probabilistically checkable proofs of proximity (PCPPs)* the verifier is given *implicit* (i.e., oracle) access to the proof, and the length of the proof is typically super-linear.

1.2.3 Arguments of Proximity

Proofs of proximity guarantee the correctness of the result even if the prover deviates *arbitrarily* from the protocol. A natural relaxation to consider is limiting attention to adversaries that are *computationally bounded* - that is, soundness is only required to hold against *computationally bounded* cheating provers. Together with Kalai [KR14], we considered this natural relaxation, which we called *interactive arguments of proximity*.

We constructed such arguments of proximity that improve on the interactive proofs of proximity of [RVW13] by (1) capturing a richer class of functions and (2) minimizing the number of rounds in the interaction to a single round. More specifically, assuming the existence of a sub-exponentially secure FHE scheme, we constructed a *one-round* argument of proximity for *every language* computable in time t, where the running time of the verifier is $o(n) + \mathsf{polylog}(t)$ and the running time of the prover is $\mathsf{poly}(t)$.

In contrast, assuming sufficiently hard cryptographic PRGs, we showed a language in P for which the parameters of our argument-system are close to optimal. In addition, based on similar cryptographic assumptions, we also show a language in NC_1, for which the parameters of the IPPs of Rothblum *et al.* [RVW13] are close to optimal.

Finally, using adequate error correcting codes, we observed that any one-round argument of proximity immediately yields a one-round standard delegation scheme (i.e., the verifier rejects *every* false input) where the verifier runs in exact *linear* time. See Chapter 5 for details.

1.2.4 Proofs of Proximity for Context-Free Languages and Read-Once Branching Programs

To further demonstrate the usefulness of both interactive and non-interactive proofs of proximity, a basic goal is to design such protocols for natural languages or even classes of languages. Together with Goldreich and Gur [GGR15], we considered this questions and constructed proofs of proximity for two natural classes of properties: (1) context-free languages, and (2) languages accepted by small read-once branching programs. Our main results are:

1. *Non-interactive* proofs of proximity for these two classes, in which, for inputs of length n, both the verifier's query complexity and the length of the proof are $\tilde{O}(\sqrt{n})$.
2. *Interactive* proofs of proximity for the same two classes with constant query complexity, poly-logarithmic communication complexity, and logarithmically many rounds of interaction.

See Chapter 4 for details.

1.3 Organization

The thesis is organized in chapters, where each chapter contains a full version of a published paper or a paper that is currently in submission. Hence, each chapter may be

read independently of all other chapters. We note that (1) the first section of each chapter ends with an organization subsection that outlines the structure of that chapter and (2) appendices relevant to each chapter appear immediately at the end of the relevant chapter.

In Chapter 2 we present our construction of a delegation scheme for any language in P (based on [KRR13b]). In Chapter 3 we present our results on non-interactive proofs of proximity (based on [GR15b]). In Chapter 4 we show how to construct efficient proofs of proximity for context-free languages and languages accepted by small read-once branching programs (based on [GGR15]). In Chapter 5 we discuss our results for arguments of proximity (including also the aforementioned lower bound for interactive proofs of proximity) (based on [KR14]).

Finally, in Appendix A we give a high-level overview of results that were obtained during our doctoral studies but are not directly related to verifiable outsourcing of computation and were therefore not included in the thesis.

Chapter 2

Delegation for P

2.1 Introduction

The problem of delegating computation considers a setting where one party, the *delegator* (or *verifier*), wishes to delegate the computation of a function f to another party, the *worker* (or *prover*). The challenge is that the delegator may not trust the worker, and thus it is desirable to have the worker "prove" that the computation was done correctly. We require that verifying this proof is significantly easier than doing the computation itself; that is, the delegator's running time is significantly smaller than the time complexity of f. Moreover, we require that the running time of the worker is not much larger than the time complexity of f.

The problem of delegating computation became a central problem in cryptography, especially with the increasing popularity of cloud computing, where weak devices use cloud platforms to run their computations.

We focus on the problem of constructing *one-round* delegation protocols, where the delegator wants to verify a statement of the form $x \in \mathcal{L}$. The delegator sends x to the worker together with some query q; then the worker computes $b = \mathcal{L}(x)$, and based on the query q provides a *non-interactive* proof π for the fact that $b = \mathcal{L}(x)$. The delegator should be able to verify the correctness of the proof π very efficiently, and the worker should run in time polynomial in the time it takes to compute f. Throughout this work (similarly to all previous works that consider the problem of one-round delegation), the security requirement is against *computationally bounded* cheating workers. Namely, we consider the computational setting, where the security (i.e., soundness) of our scheme relies on a cryptographic assumption, and the guarantee is that any cheating worker, who cannot break the underlying assumption, cannot prove the correctness of an incorrect statement.

Previously, [GKR08, KR09] proved that (assuming the existence of a sub-exponentially secure computational PIR scheme) any function f that can be computed by a LOGSPACE-uniform circuit C of size $t = t(n)$ and depth $d = d(n)$, has a one-round delegation scheme where the running time of the verifier is $\tilde{O}(n + d)$, and the running time of the prover is $\mathsf{poly}(t)$.[1] Note however that for circuits with large depth d this delegation scheme does

[1] As is the case with all computationally sound delegation schemes, the runtime of both the prover

not satisfy the efficiency criterion.

A fundamental question is: Do there exist efficient 1-round delegation schemes for *all* deterministic computations? There are several works that (partially) answer this question in the preprocessing model, or under *non-falsifiable* assumptions.[2] We elaborate on these works in Section 2.1.4.

In this chapter, we answer the above question positively, by constructing a 1-round delegation scheme for *every* deterministic computation, assuming a sub-exponentially secure computational PIR scheme. More specifically, we show a delegation scheme for every language computable in time $t = t(n)$, where the running time of the verifier is $n \cdot \mathsf{polylog}(t)$, and the running time of the prover is $\mathsf{poly}(t)$. The underlying assumption is that there exists a computational PIR scheme (or an FHE scheme) that cannot be broken in time $t^{\mathsf{polylog}(t)}$ for security parameter $k \leq \mathsf{poly}(n)$.[3]

Our delegation scheme exploits a connection to the seemingly unrelated model of multi-prover interactive proof systems (MIP) in which soundness holds even against *no-signaling* cheating provers. Loosely speaking, no-signaling provers are allowed to use arbitrary strategies (as opposed to local ones, where the reply of each prover is a function only of her own input), as long as their strategies cannot be used for communication between any two disjoint sets of provers.

We show that any MIP that is sound against no-signaling cheating provers can be converted into a 1-round delegation scheme, using a fully-homomorphic encryption scheme (FHE), or alternatively, using a computational private information retrieval (PIR) scheme. We elaborate on this connection in Section 2.1.2.

We then construct a new MIP, for every deterministic language, with soundness against no-signaling cheating provers. This, together with the transformation above, gives us our 1-round delegation scheme.

2.1.1 Multi-Prover Interactive Proofs with No-Signaling Provers

The study of MIPs that are secure against no-signaling provers was motivated by the study of MIPs with provers that share entangled quantum states. Recall that no-signaling provers are allowed to use arbitrary strategies, as long as their strategies cannot be used for communication between any two disjoint sets of provers. By the physical principle that information cannot travel faster than light, a consequence of Einstein's special relativity theory, it follows that all the strategies that can be realized by provers that share entangled quantum states are no-signaling strategies.

Moreover, the principle that information cannot travel faster than light is a central principle in physics, and is likely to remain valid in any future ultimate theory of nature, since its violation means that information could be sent from future to past. Therefore,

and the verifier also grows polynomially with the security parameter. To avoid cluttering of notation, throughout this introduction, we omit this dependence on the security parameter.

[2] We note that under non-falsifiable assumptions, there are known positive results even for non-deterministic computations. The focus of this work is on deterministic computations.

[3] In particular, for languages in P we only require a PIR scheme with quasi-polynomial security.

soundness against no-signaling strategies is likely to ensure soundness against provers that obey a future ultimate theory of physics, and not only the current physical theories that we have, that are known to be incomplete.

The study of MIPs that are secure against no-signaling provers is very appealing also because no-signaling strategies have a simple mathematical characterization.

Loosely speaking, in a no-signaling strategy the answer given by each prover is allowed to depend on the queries to all other provers, as long as for any subset of provers S, and any queries given to the provers in S, the distribution of the answers given by the provers in S is independent of all the other queries. Thus, the answer of each prover can depend on the queries to all other provers as a function, but not as a random variable.

More formally, fix any MIP consisting of ℓ provers, and fix any set of cheating provers $\{P_1^*, \ldots, P_\ell^*\}$ who may see each other's queries (and thus each answer may depend on the queries sent to all the provers). The provers are said to be *no-signaling* if for every subset of provers $\{P_i^*\}_{i \in S}$, and for every two possible query sets $\{q_i\}_{i \in [\ell]}$ and $\{q'_i\}_{i \in [\ell]}$ such that $q_i = q'_i$ for every $i \in S$, it holds that the distributions of answers $\{a_i\}_{i \in S}$ and $\{a'_i\}_{i \in S}$ are *identical*, where $\{a_i\}_{i \in S}$ is the the answers of the provers in S corresponding to the queries $\{q_i\}_{i \in [\ell]}$, and $\{a'_i\}_{i \in S}$ is the answers of the provers in S corresponding to the queries $\{q'_i\}_{i \in [\ell]}$. If we have the slightly weaker guarantee that these two distributions are statistically close, then we say that the provers are *statistically no-signaling*. More specifically, if these two distributions are δ-close, then we say that the provers are δ-no-signaling. We refer the reader to Section 2.4.3 for details.

No-signaling strategies were first studied in physics in the context of Bell inequalities by Khalfin and Tsirelson [KT85] and Rastall [Ras85], and they gained much attention after they were reintroduced by Popescu and Rohrlich [PR94]. MIPs that are secure against no-signaling provers were extensively studied in the literature (see for example [Ton09, BLM+05, AII06, KKM+08, IKM09, Hol09, Ito10]). However, their precise power was unknown. It was known that they contain PSPACE [IKM09] and are contained in EXP.[4] For the case of *two* provers, Ito [Ito10] showed that the corresponding complexity class is contained in (and therefore equal to) PSPACE. Characterizing the exact power of MIPs (with more than two provers) that are secure against no-signaling provers remained an open problem.

In this chapter, we solve this open problem by constructing MIPs that are secure against no-signaling strategies (and more generally, statistically no-signaling strategies), for every language in EXP. Moreover, in our construction the provers are *efficient*; i.e., they run in time that is polynomial in the computation time. Specifically, for any language computable in time $t = t(n)$, we construct an MIP that is sound against no-signaling strategies, where the running time of the provers is $\mathsf{poly}(t)$, the number of provers is $\mathsf{polylog}(t)$, and the running time of the verifier is $n \cdot \mathsf{polylog}(t)$. The fact that our MIP is efficient implies that the resulting 1-round delegation scheme is efficient. We note that the previous construction of MIP that is sound against no-signaling strategies for PSPACE [IKM09] is inefficient (the provers run in time exponential in the space of the

[4] In a nutshell, one can find the best strategy for the provers by solving an exponential size linear program.

computation).

2.1.1.1 The Challenges in Proving Soundness Against No-Signaling Strategies

It is tempting to consider known constructions of MIPs and to try to prove their soundness against no-signaling strategies. However, known constructions of MIPs are usually for NEXP (or the scaled down version for NP). Since MIPs that are secure against no-signaling strategies are contained in EXP, there is no hope to construct such MIPs for NEXP. In particular, all known MIPs for NEXP (or the scaled down version for NP) are not sound against no-signaling strategies.

Indeed, often the trivial strategy, where the provers simply choose random answers that make the verifier accept, is no-signaling. For example, consider the trivial 2-prover interactive proof for graph 3-coloring , where the verifier sends each prover a vertex in the graph, where with probability 1/2 the vertices are the same and with probability 1/2 there is an edge between these vertices, and the provers reply with the color of these vertices. Suppose the graph is not 3-colorable. We argue that the "random accepting strategy" is a no-signaling strategy that is accepted with probability 1. More specifically, the cheating strategy is the following: If both vertices are the same, choose a random color from the set of three legal colors, and both provers send this color to the verifier. Otherwise, choose two different random colors from the set of three legal colors, and each prover sends one of these colors to the verifier. This strategy is clearly accepted with probability 1. Moreover, it is a no-signaling strategy, since the distribution of answers of each prover is uniform, independent of the query to the other prover.

This intuitive argument extends to more sophisticated MIPs and demonstrates the difficulty in proving soundness against no-signaling strategies.

2.1.2 From Multi-Prover Interactive Proofs to One-Round Delegation

Aiello *et al.* [ABOR00] suggested a method for converting a 1-round MIP into a 1-round delegation scheme, by using a PIR scheme (or an FHE scheme).[5] In this work, we choose to use the terminology of FHE schemes (as opposed to PIR schemes), because we find this terminology to be simpler. However, all our results hold with PIR schemes as well.

In the resulting delegation scheme, the verifier computes all the queries of the MIP verifier, and sends all these queries to the prover, each encrypted under a fresh key, using an FHE scheme. The prover then computes the MIP provers' responses homomorphically over the encrypted queries, that is, underneath the layer of the FHE scheme.

Unfortunately, shortly after this method was introduced, Dwork *et al.* [DLN+04] showed that it may, in general, be insecure. We elaborate further on the work of Dwork *et al.* and their connection to no-signaling soundness in Section 2.1.4.

[5] Actually, [ABOR00] suggested to use a PCP. However, as pointed out by [DLN+04] an MIP is more suitable.

Motivated by the work of Aiello *et al.*, Kalai and Raz [KR09] showed that a variant of this method can be used to securely convert any interactive proof into a one-round argument system. The idea is simply to have the verifier send all its (say t) messages in the first round, in the following redundant form: For every $i \in [t]$, all the first i messages are encrypted using a fresh FHE key.[6] The work of [KR09], together with the interactive delegation scheme of Goldwasser *et al.* [GKR08], gives rise to the 1-round delegation protocol for LOGSPACE-uniform low-depth circuits, mentioned above.

We show that the method of Aiello *et al.* [ABOR00] is secure if the underlying MIP is sound against statistically no-signaling strategies. Thus, we reduce the cryptographic problem of constructing secure one-round delegation schemes, to the information theoretical problem of constructing MIP schemes that are secure against statistically no-signaling provers. Such a reduction allows us to "strip off" the cryptography, and to focus on an information theoretic question of constructing an MIP that is secure against statistically no-signaling provers.

This result generalizes the work of [KR09], since any interactive proof can be seen as an MIP where the verifier sends his first i messages to prover i (it is quite easy to verify that the resulting MIP is secure against statistically no-signaling cheating provers). Moreover, our result significantly simplifies the one of [KR09], which implicitly converts the interactive proof into an MIP scheme and then applies the PIR to the resulting MIP scheme. We believe that due to the lack of the "correct" terminology, the result of [KR09] was relatively complicated, whereas this current result is significantly simpler and more general. We refer the reader to Section 2.16 for details.

2.1.3 Summary of Our Results

We show that when applying the method of Aiello *et al.* [ABOR00] to an MIP that is sound against statistically no-signaling cheating provers, then the resulting 1-round delegation protocol is secure (assuming that the underlying FHE is secure against attackers of sub-exponential size).

Informal Theorem 2.1 (See Theorem 2.11)**.** *Assuming the existence of an* FHE *scheme with sub-exponential security, there exists an efficient way to convert any* 1*-round* MIP *that is sound against statistically no-signaling cheating provers into a secure 1-round delegation scheme, where the running time of the prover and verifier in the delegation scheme are proportional to the running time of the provers and verifier in the* MIP*.*

Remark. More specifically, the precise assumption needed in Informal Theorem 2.1 is that there exists an FHE scheme that, for security parameter $k \leq \mathsf{poly}(n)$, is secure against adversaries running in time $2^{O(|a_1|+\ldots+|a_\ell|)}$, where $|a_i|$ is the answer size of the i'th prover in the underlying MIP scheme.

[6] The reason the i'th message is encrypted together with the preceding messages, is since the prover's reply may depend on all these messages.

Thus, we reduced the cryptographic problem of constructing secure one-round delegation schemes, to the information theoretical problem of constructing MIP schemes that are secure against statistically no-signaling provers.

We then construct an efficient MIP, that is sound against statistically no-signaling strategies, for every language in EXP.

Informal Theorem 2.2 (See Theorem 2.4)**.** *For any language L computable in time $t = t(n)$, there exists an* MIP *that is secure against statistically no-signaling adversaries. The (honest) provers in this* MIP *run in time* $\mathsf{poly}(t)$*, the number of provers and the communication complexity is* $\mathsf{polylog}(t)$*, and the verifier runs in time* $n \cdot \mathsf{polylog}(t)$*.*

We note that our MIP has the additional property that the verifier does not need to know the entire input, but rather only needs to access a few points in the low-degree extension of the input (we refer the reader to Section 2.4.6 for the definition of low-degree extension). This property, which was also a property of the [GKR08] protocol, is important for applications such as memory delegation [CKLR11].

The above theorem, together with Informal Theorem 2.1, immediately yields the following corollary:

Informal Theorem 2.3 (See Theorems 2.8-2.10)**.** *Assume the existence of an* FHE *scheme with sub-exponential security. Then, there exists a* 1*-round delegation scheme for any function computable in time $t = t(n)$. The prover in this delegation scheme runs in time* $\mathsf{poly}(t)$*, the verifier runs in time* $n \cdot \mathsf{polylog}(t)$*, and the communication complexity is* $\mathsf{polylog}(t)$*.*

Remark. As in Informal Theorem 2.1, the precise assumption needed for the above theorem is the existence of an FHE scheme that, for security parameter $k \leq \mathsf{poly}(n)$, is secure against adversaries running in time $2^{\mathsf{polylog}(t)}$.

As a special case, Informal Theorem 2.2 gives soundness against provers that share an entangled quantum state, since such provers are no-signaling. This gives a scheme for delegating computation to a group of workers that cannot communicate with each other (where the parameters are as in Theorem 2). The scheme is information theoretically secure even if the workers share an entangled quantum state. Moreover, the scheme remains secure in any future ultimate theory (that may extend quantum theory) as long as the no-signaling principle remains valid. We note, however, that recent breakthroughs by Ito and Vidick construct MIPs that are secure against provers that share entangled quantum states, for any language in NEXP [IV12, Vid13].

The bulk of technical contribution of this work is in proving Informal Theorem 2.2. As noted above, proving this theorem requires overcoming several technical hurdles that do not appear in the classical MIP (or PCP) setting. We refer the reader to Section 2.3 for an overview of our techniques for proving this theorem.

Informal Theorem 2.1 is mainly a conceptual contribution. Its proof is relatively straightforward, but we find the connection between the seemingly unrelated concepts of delegation and no-signaling soundness to be intriguing.

2.1.4 Related Work

Our work is greatly inspired by the work of Aiello *et al.* [ABOR00], who propose a general methodology of constructing 1-round delegation schemes, by combining an MIP (or a PCP) with a (computational) PIR scheme. Also very relevant to our work is the work of Dwork *et al.* [DLN+04], who proved that this method is not sound, by giving an example of a PCP for which the resulting one-round delegation scheme is not sound, no matter which PIR scheme (or FHE scheme) is used.

Moreover, [DLN+04] define the notion of a "spooky interaction" which is a behavior of the cheating prover, that on the one hand does not directly contradict the security of the PIR, yet on the other hand is not consistent with answers based on PIR databases. Using our terminology, a spooky behavior is exactly a no-signaling distribution on prover answers that are computed "homomorphically" under the "encrypted" PIR queries.

More importantly, Dwork *et al.* also argue that the soundness of the [ABOR00] technique cannot essentially be based on any MIP (or PCP). However, Dwork *et al.* (and [ABOR00]) were focused on constructing 1-round delegation schemes for non-deterministic languages (such as languages in NEXP or the scaled down version of NP). Indeed, it is implicitly shown in [DLN+04] that languages that can be proved by an MIP with soundness against no-signaling provers are in EXP (and the scaled down version of it is contained in P). Additionally, Gentry and Wichs [GW11] recently showed a negative result, proving that there does not exist a non-interactive delegation scheme for NP with a black-box proof of security under any falsifiable assumption.[7] However, these negative results do not apply to our setting as our delegation scheme is not for all of NP, but rather for languages in P (or, in the scaled up version, in EXP).

Thus, by focusing on deterministic classes (as opposed to non-deterministic ones), we manage to show that the [ABOR00] method is indeed sound in some cases.

Related work on computation delegation. Beyond the works of [GKR08, KR09] that were mentioned earlier, there are many other works on delegating computation that are less relevant to this work. Let us mention a few. In the *interactive* setting, Kilian [Kil92] constructed a 4-message delegation scheme for every function in NEXP. Micali [Mic94] showed that in the so called *random oracle model* this result can be made non-interactive, by relying on the Fiat-Shamir paradigm [FS86]. There are also several results that construct non-interactive delegation schemes under *non-falsifiable* assumptions (as defined by Naor [Nao03]). These works include [Gro10, Lip12, BCCT12a, DFH12, GLR11, BCCT12b, GGPR12] and more. Finally, we mention a series of results that construct non-interactive delegation scheme in the *preprocessing model*, where the verifier is efficient only in the amortized setting. These results include [GGP10, CKV10, AIK10, PRV12]. There are many other results that we do not mention, which consider various different models, or are concerned with practical efficiency.

[7] The model of [GW11] differs from our model in that they allow the prover the additional power of choosing the instance x after seeing the first message sent by the verifier.

2.1.5 Organization

In Section 2.2, we formally state our results. In Section 2.3, we provide a high-level overview of our techniques. In Section 2.4, we formally define the notions that we use throughout this work. In Sections 2.5 to 2.8, we construct a *base* PCP with soundness against no-signaling strategies for PSPACE. In Sections 2.9 to 2.11, we construct the *augmented* PCP for EXP. In Sections 2.12 to 2.14, we show how to transform this PCP into an MIP. In Section 2.15, we use the tools from all previous sections to prove the main information theoretic result. Finally, in Section 2.16 and Section 2.17, we show how to transform our MIP into an 1-round delegation scheme.

2.2 Our Results

We show a general result on MIP proof systems that are secure against no-signaling strategies and show how to use the latter to construct a new 1-round delegation scheme (a.k.a. 1-round argument-system).

Theorem 2.4. *Suppose that* $\mathcal{L} \in \mathsf{DTIME}(t(n))$*, where* $t = t(n)$ *satisfies* $\mathsf{poly}(n) \leq t \leq \mathsf{exp}(n)$*. Then, for any integer* $(\log t)^c \leq k \leq \mathsf{poly}(n)$*, where* c *is some (sufficiently large) universal constant, there exists an* MIP *for* $\mathcal{L}$ *with* $k \cdot \mathsf{polylog}(t)$ *provers and with soundness error* 2^{-k} *against* $2^{-k \cdot \mathsf{polylog}(t)}$*-no-signaling strategies.*

The verifier runs in time $n \cdot k^2 \cdot \mathsf{polylog}(t)$ *and the provers run in time* $\mathsf{poly}(t, k)$*. Each query and answer is of length* $k \cdot \mathsf{polylog}(t)$*.*

By setting the parameters $t = \mathsf{poly}(n)$ and $k = \mathsf{polylog}(n)$ we obtain the following corollary:

Corollary 2.5. *If* $\mathcal{L} \in \mathsf{P}$*, then there exists an* MIP *for* $\mathcal{L}$ *with* $\mathsf{polylog}(n)$ *provers, and with soundness error* $2^{-\mathsf{polylog}(n)}$ *against* $2^{-\mathsf{polylog}(n)}$*-no-signaling strategies. The verifier runs in time* $\tilde{O}(n)$ *and the provers run in time* $\mathsf{poly}(n)$*. Each query and answer is of length* $\mathsf{polylog}(n)$*.*

A scaled up result is obtained by setting $t = \mathsf{exp}(n)$ and $k = \mathsf{poly}(n)$:

Corollary 2.6. *If* $\mathcal{L} \in \mathsf{EXP}$*, then there exists an* MIP *for* $\mathcal{L}$ *with* $\mathsf{poly}(n)$ *provers and with soundness error* $2^{-\mathsf{poly}(n)}$ *against* $2^{-\mathsf{poly}(n)}$*-no-signaling strategies. The verifier runs in time* $\mathsf{poly}(n)$ *and the provers run in time* $\mathsf{exp}(n)$*. Each query and answer is of length* $\mathsf{poly}(n)$*.*

Having stated our main information-theoretic results, we proceed to state our main cryptographic results. The following theorems rely on the existence of an (S, δ)-secure FHE scheme, which is an FHE scheme where any $\mathsf{poly}(S)$-size adversary cannot distinguish between an encryption of any two messages with probability greater than δ (see Section 2.4.7 for a formal definition).[8]

[8] Alternatively, we can rely on the existence of a sufficiently strong cryptographic private information retrieval scheme (PIR), see remark at the end of Section 2.17.

We first state our general transformation from any MIP that has soundness against no-signaling strategies into a 1-round argument-system.

Theorem 2.7 (Simplified; for the full statement see Theorem 2.11)**.** *Suppose that the language $\mathcal{L}$ has an MIP with ϵ soundness against δ-no-signaling strategies and a total of λ communication (to all provers). Let $\tau = \tau(n) \geq \lambda$ be a security parameter, where n denotes the input length of the MIP. For every $S = S(\tau) \geq \tau$ such that $S \geq \max(n, 2^{\lambda})$ and $\delta' = \delta'(\tau)$ such that $\delta' \leq \delta/\lambda$, if there exists an (S, δ') secure FHE, then the language $\mathcal{L}$ has a 1-round argument system with soundness (S, ϵ).*

If the MIP verifier runs in time T_V, then the running time of the resulting verifier is $T_V + \mathsf{poly}(\tau)$. If the running time of each MIP prover is T_P, then the running time of the resulting prover is $\mathsf{poly}(T_P, \tau, n)$. The total communication in the resulting argument-system is of length $\mathsf{poly}(\tau)$.

By combining Theorem 2.4 with Theorem 2.7 we obtain the following argument-system:

Theorem 2.8. *Suppose that $\mathcal{L} \in \mathsf{DTIME}(t(n))$, where $t = t(n)$ satisfies $\mathsf{poly}(n) \leq t \leq \mathsf{exp}(n)$. Let $\tau = \tau(n)$ be a security parameter such that $\log(t) \leq \tau \leq \mathsf{poly}(t)$. Let $S = S(\tau) \geq \tau$ such that $2^{(\log(t))^c} \leq S \leq 2^{\mathsf{poly}(n)}$ and $S \leq 2^{\max(n,\tau)}$, where c is some sufficiently large universal constant. If there exists an $\left(S, 2^{-\sqrt{\log S}}\right)$-secure FHE, then $\mathcal{L}$ has a 1-round argument system with soundness $\left(S, 2^{-\frac{\sqrt{\log S}}{\mathsf{polylog}(t)}}\right)$. The verifier runs in time $n \cdot \log(S) \cdot \mathsf{polylog}(t) + \mathsf{poly}(\tau)$ and the prover runs in time $\mathsf{poly}(t)$. The total communication is of length $\mathsf{poly}(\tau)$.*

We stress that the running time of the verifier in Theorem 2.8 only depends *poly-logarithmically* on the time that it takes to compute $\mathcal{L}$. We proceed to describe two useful corollaries of Theorem 2.8.

By setting $t = \mathsf{poly}(n)$, $\tau = n^{\epsilon}$ and $S(\tau) = 2^{(\log(\tau))^c}$ where $\epsilon > 0$ (resp., $c > 0$) is a sufficiently small (resp., large) universal constant and assuming the existence of a quasi-polynomially secure FHE, we obtain a (cryptographic) delegation scheme for P with quasi-linear verification and sublinear communication.

Theorem 2.9. *Suppose that $\mathcal{L} \in \mathsf{P}$. If there exists a $\left(2^{\mathsf{polylog}(\tau)}, 2^{-\mathsf{polylog}(\tau)}\right)$-secure FHE, then, for every constant $\alpha > 0$, the language $\mathcal{L}$ has a 1-round argument system with soundness $\left(2^{\mathsf{polylog}(n)}, 2^{-\mathsf{polylog}(n)}\right)$. The verifier runs in time $\tilde{O}(n)$ and the prover runs in time $\mathsf{poly}(n)$. The total communication is of length $O(n^{\alpha})$.*

By setting $t = \mathsf{poly}(n)$, $\tau = (\log(n))^c$ and $S(\tau) = 2^{\tau^{\epsilon}}$ where $\epsilon > 0$ is a sufficiently small universal constant, $c > 0$ is a sufficiently large universal constant (that is chosen after ϵ) and assuming the existence of a *sub-exponentially* secure FHE (a stronger assumption than that in Theorem 2.9) we obtain a (cryptographic) delegation scheme for P with quasi-linear verification but only *poly-logarithmic* communication.

Theorem 2.10. *Suppose that* $\mathcal{L} \in \mathsf{P}$. *If there exists a* $\left(2^{\tau^\epsilon}, 2^{-\tau^{\epsilon/2}}\right)$*-secure* FHE*, where* $\epsilon > 0$ *is a sufficiently small universal constant, then* $\mathcal{L}$ *has a 1-round argument system with soundness* $\left(2^{\mathsf{polylog}(n)}, 2^{-\mathsf{polylog}(n)}\right)$. *The verifier runs in time* $\tilde{O}(n)$ *and the prover runs in time* $\mathsf{poly}(n)$. *The total communication is of length* $\mathsf{polylog}(n)$.

2.3 Our Techniques

Our techniques can be separated into two parts. The main technical contribution of this work is the construction of an MIP that is sound against statistically no-signaling cheating provers, for any function computable in time t. The number of provers is $\mathsf{polylog}(t)$, each prover runs in time at most $\mathsf{poly}(t)$, and the verifier runs in time $n \cdot \mathsf{polylog}(t)$. This construction, described in Section 2.3.1, is information theoretic, and does not rely on any cryptographic assumptions.

Then, in Section 2.3.2 we show how to convert a statistically no-signaling MIP into a one-round argument (while preserving the parameters, up to polynomial factors). The soundness of the resulting one-round argument assumes the existence of a fully homomorphic encryption (FHE) scheme with sub-exponential security.

2.3.1 Our Statistically No-Signaling MIP

We start by giving an overview of our MIP, and then give the high-level idea for why soundness holds against statistically no-signaling cheating provers. The proof of soundness requires a different approach than the ones taken to prove classical soundness. Indeed, all known MIP's for NEXP (or the scaled down version of NP) are not sound against no-signaling adversaries (see discussion in Section 2.1.1.1).

The main difference between a classical MIP and a no-signaling MIP is that in a classical MIP once a prover fixes his random tape (if at all he uses randomness), then his answer is a deterministic function of his query. This is not the case in the no-signaling setting, since a prover's answer can depend on the other queries. It is required that the answer of the prover is independent of the other queries *as a random variable* , but it may certainly depend on the other queries as a function. This makes the soundness proof significantly more challenging.

Before presenting the high level ideas of this proof, we first give a high level overview of our MIP.

As a first step in the construction of our MIP, we would like to assume for simplicity that any set of (possibly malicious) provers behave *symmetrically*; namely, any two subsets of provers, who are asked the same questions, answer similarly. Of course, we cannot ensure such a thing, since cheating provers may behave arbitrarily. Instead, this is ensured by defining a new model of no-signaling PCP, as oppose to no-signaling MIP.

Intuitively, a no-signaling PCP is defined like a classical PCP, but where soundness is required to hold also against a *no-signaling* prover, who can see all queries. Loosely speaking, a no-signaling prover, upon receiving any set of queries Q, may reply with

answers, where each answer may depend on all the queries in Q *as a function*, but not as a random variable. Namely, for any set of queries Q and for any subset $Q' \subseteq Q$, the *distribution* of the answers corresponding to the queries Q', should be independent of queries in $Q \setminus Q'$.

Formally, a no-signaling prover consists of a family of distributions $\{\mathcal{A}_Q\}$, where there is one distribution for every "sufficiently small" set of queries Q, and the requirement is that for every subset of queries $Q' \subseteq Q$, the distribution $(\mathcal{A}_Q)|_{Q'}$ (which is the distribution of answers $\mathcal{A}_Q$ restricted to queries in Q') is independent of queries in $Q \setminus Q'$. More generally, a δ-no-signaling family of distributions has the property that for every three sets of queries Q_1, Q_2, Q', such that $Q' \subseteq Q_1$ and $Q' \subseteq Q_2$, the distributions $(\mathcal{A}_{Q_1})|_{Q'}$ and $(\mathcal{A}_{Q_2})|_{Q'}$ are δ-close. We emphasize that in a δ-no-signaling PCP we think of a set of queries Q as an *unordered* set, thus achieving the desired *symmetry*; i.e., the answers do not depend on the order of the queries.

We note, however, that the definition of δ-no-signaling PCP given above, is not complete. One needs to define what is a "sufficiently small set of queries". We define it to be all the query sets with at most $k_{\max}$ queries. k_{max} is an important parameter. The larger k_{max} is, the more limited the cheating provers are.[9] We denote such a PCP by (k_{max}, δ) no-signaling PCP, and define it formally in Section 2.4.5. We devote most of the technical sections to constructing a (k_{max}, δ)-no-signaling PCP and proving its soundness.

Converting this PCP into a δ-no-signaling MIP is relatively straightforward. The basic idea is that the MIP verifier emulates the PCP verifier, and sends each query to a random prover (that was not yet asked any query). Each prover answers by simulating the (honest) PCP. The parameter k_{max} corresponds to the number of provers in the resulting MIP. In this work, $k_{max} = \mathsf{polylog}(t)$, and thus the number of provers in our MIP is $\mathsf{polylog}(t)$, and the verifier in our one-round argument runs in time $n \cdot \mathsf{polylog}(t)$.

Overview of our underlying PCP. We present our PCP in two steps. First, we construct a "base" PCP for languages in PSPACE. Then we show how to augment this PCP, and construct a PCP for EXP. We prove that both PCPs are sound against statistically no-signalling strategies.

2.3.1.1 Our Base PCP.

Let $\mathcal{L}$ be a language computable by a (deterministic) Turing machine running in time $t(n)$ and space $s(n)$ on instances of length n. Our base PCP for $\mathcal{L}$ has $k_{max} = \tilde{O}(s(n))$. This PCP is similar to known PCPs (in particular, to the PCP of [Sud00]). The main points of distinction are that in our base PCP each test is repeated k times, where k is a security parameter, and that our PCP is applied for deterministic computations, rather than for non-deterministic computations.

Suppose that the prover needs to prove that $x \in \mathcal{L}$, where x is an instance of length n. The underlying PCP consists of several low degree multi-variate polynomials. The first

[9] Jumping ahead, we note that in this work $k_{max} = \mathsf{polylog}(t)$.

polynomial is the low-degree extension (defined in Section 2.4.6) of the entire computation. More specifically, let $\mathcal{C}_n$ be a circuit of size $N = O(t(n)s(n))$ that computes $\mathcal{L}$ on inputs of length n. It is known that the circuit $\mathcal{C}_n$ can be made layered, with $O(s(n))$ gates in each layer, and $O(t(n))$ layers.

Assume that the wires of the circuit are indexed by the numbers $1, \ldots, N$, in an order that agrees with the layers of the circuit. In particular, the indexes of wires at layer i are larger than the indexes of wires at layer $i-1$. We assume that $1, \ldots, n$ are the indexes of the n input variables and that N is the index of the output wire. Let $x_1, \ldots, x_N$ be the values of the N wires of the circuit $\mathcal{C}_n$ when computed with input $x = (x_1, \ldots, x_n)$.

The entire computation $x_1, \ldots, x_N$ appears in the PCP encoded using an error correcting code (specifically, using the low-degree extension encoding), so that if a single bit in the computation is incorrect it causes a global affect on the encoding.

In addition, the PCP contains several other low-degree multi-variate polynomials, denoted by $P_0, P_1, \ldots, P_\ell$, which are defined in Section 2.5. In this overview we ignore these polynomials.

The analysis of our base PCP. The analysis of our base PCP begins with an *error amplification* step, where (loosely speaking) we prove that if there exists a (statistically) no-signaling prover (which is a family of distributions, one for every possible set of queries), that convinces the PCP verifier to accept a statement of the form $\mathcal{C}_n(x) = b$ with some non-negligible probability, then there exists a (statistically) no-signaling prover that convinces a different verifier, called, the *relaxed verifier*, to accept the same statement with probability close to 1 (i.e., with probability $1 - \frac{1}{\mathsf{poly}(t)}$ for any polynomial poly).

This error amplification step, which is a crucial step in our proof, is achieved as follows: Recall that the verifier V repeats each test k times, and accepts if and only if all tests accept. We define a "relaxed" verifier V' that makes the exact same queries as V, but accepts if and only if for each (repeated) test, at least r of the k repetitions are accepting, where r is a parameter. Loosely speaking, we prove that if the verifier V accepts with probability ϵ then the relaxed verifier accepts with probability $1 - \frac{\tilde{O}(2^{-r})}{\epsilon}$, where $\tilde{O}$ hides $\mathsf{polylog}(N)$ factors.

To prove this we argue that if V and V' choose their queries independently then the probability that V accepts and V' rejects is very small. This is true because for each group of k tests we can first choose the $2k$ tests for both V and V', and only then decide which tests go to V and which ones go to V'. Consider the answers for these $2k$ tests. (It is important here that k_{max} is greater than the total number of queries in these $2k$ tests, so that all these queries can be asked simultaneously.) If among the $2k$ tests many are rejected then V rejects with high probability. On the other hand, if among the $2k$ tests only few are rejected (say, less than r) then V' always accepts. We refer the reader to Section 2.6 for details.

In this overview, we ignore the fact that the relaxed verifier is different than the actual verifier, and assume for simplicity that there is a (statistically) no-signaling prover that convinces the actual verifier to accept with probability $1 - \frac{1}{\mathsf{poly}(t)}$. We will prove that in that case the statement $\mathcal{C}_n(x) = b$ must be correct.

To this end, we first prove that for every (statistically) no-signaling prover, if the PCP verifier accepts with probability $1 - \frac{1}{\mathsf{poly}(t)}$, then it must be the case that the distributions corresponding to queries in $\{x_1, \ldots, x_N\}$ are *locally consistent*. More specifically, we prove that for every gate in the circuit $\mathcal{C}_n$, and for every set of queries that include the two input wires and the output wire of the gate, the answers of values of the inputs and output wires are consistent with the gate, with very high probability (say, higher than $1 - \frac{1}{t^3}$). We note that this guarantee only requires $k_{max} = \mathsf{polylog}(t)$, and in particular the dependence on the space s is not needed to obtain this local consistency guarantee.[10]

We note that the local consistency guarantee is only true if variables in $\{x_1, \ldots, x_N\}$ are read in a certain way, which uses interpolation and the local decoding properties of the low-degree extension encoding. In this overview, for simplicity, we completely ignore this extra complication and assume that the local consistency guarantee holds as stated above.

From a classical perspective, it seems that the local consistency guarantee should immediately imply global consistency, and thus correctness, by applying a straightforward union bound. However, in the no-signaling setting, this intuition is misleading. The reason is that in order to apply the union bound we need to consider the probability that *all* the local consistency conditions are met *simultaneously*, and make the following argument:

$$
\begin{aligned}
&\Pr[\text{correctness}] \geq \\
&\Pr[\text{all local consistency conditions hold}] = \\
&1 - \Pr[\exists \text{ local consistency condition that does not hold}] \geq \\
&1 - O(t \cdot s) \cdot \Pr[\text{a single local consistency does not hold}] \geq \\
&1 - O(t \cdot s) \cdot \frac{1}{t^3} \geq \\
&1 - O\left(\frac{1}{t}\right).
\end{aligned}
$$

Unfortunately, in the no-signaling setting, this type of calculation is not correct, since it is not clear what it means for *all* the local consistency conditions to hold simultaneously. Recall that there is no PCP in the sky, but rather a set of distributions for each set of queries of size at most k_{max}. Thus, we can check whether at most k_{max} local consistency conditions hold simultaneously, but not more than that, as the relevant random variables are not even defined simultaneously.

We can still use the local consistency guarantee to argue that up to a small probability of error, the probability that the output is correct is at least the probability that *both* children of the output gate are correct (using the local consistency condition). We can then proceed by induction, towards the base of the tableau. However, this will incur an exponential (in t) blowup in the error.

[10] Jumping ahead, we note that in the base PCP, $k_{max} = \tilde{\Theta}(s)$ is required to go from local consistency to global consistency.

Generally, we cannot afford this exponential blowup in error. Jumping ahead, we note that curiously, in one of the lemmas for our augmented PCP, we do use this analysis (and guarantee) for a specific computation for which the depth of the tableau is relatively small ($O(\log s)$). We elaborate on this point below.

In the analysis of our base PCP, we solve this problem by taking $k_{max} = \tilde{\Theta}(s)$, which enables us to check the correctness of an entire layer of the tableau simultaneously. More specifically, we first check the correctness of the input. The local consistency condition implies that this check passes with probability $1 - \frac{1}{\mathsf{poly}(t)}$. Then we check the first two level simultaneously. This could be done since $k_{max} = \tilde{\Theta}(s)$. The local consistency, together with the correctness of the first level, implies that the second level is correct with probability at least $\left(1 - \frac{1}{\mathsf{poly}(t)}\right)^2$. Then, we check consistency of the second and third levels, and deduce that the third level is correct with probability $\left(1 - \frac{1}{\mathsf{poly}(t)}\right)^3$. This argument continues by induction until the top layer is reached, and the conclusion is that the computation is correct with probability $\left(1 - \frac{1}{\mathsf{poly}(t)}\right)^t$, which is very close to 1, as desired.

This idea indeed works, however, it results with an MIP with $\tilde{\Theta}(s)$ provers, and thus with a one-round argument where the running time of the verifier grows linearly with s. We refer the reader to Section 2.7 for the formal analysis.

Our goal is to make k_{max} independent of s, and thus eliminate the dependency on the space. Indeed, we manage to "augment" this base PCP, and to prove that our augmented PCP is secure against statistically no-signaling distributions with $k_{max} = \mathsf{polylog}(t)$. This gives rise to an MIP where the verifier runs in time $n \cdot \mathsf{polylog}(t)$, and where the provers run in time $\mathsf{poly}(t)$. This, in turn, gives rise to our one-round delegation scheme, that achieves similar parameters.

2.3.1.2 Our Augmented PCP.

Recall that our base PCP is a proof that the computation of $\mathcal{C}_n$ was performed correctly, where $\mathcal{C}_n$ is a layered circuit of size $N = O(t(n)s(n))$ (consisting of $O(t(n))$ layers each of size $O(s(n))$), that computes $\mathcal{L}$ on inputs of length n.

The basic idea behind our *augmented* PCP is to run the same base PCP on an *augmented* circuit, denoted by $\mathcal{C}'_n$. Loosely speaking, the circuit $\mathcal{C}'_n$ computes the same function as $\mathcal{C}_n$, but in $\mathcal{C}'_n$ each layer of the circuit is augmented with the low-degree extension of the layer, and with all the low-degree tests corresponding to lines of the low-degree extension. Namely, the circuit $\mathcal{C}'_n$ is the same as $\mathcal{C}_n$, but where after each layer we insert another circuit, denoted by $\mathcal{C}_{\mathsf{LDE}}$, which takes as input the entire layer, and computes the low-degree extension of the layer, and performs *all* the low-degree tests; i.e., for every line in the low-degree extension it checks that the values on that line correspond to a low-degree univariate polynomial.[11] It is known that $\mathcal{C}_{\mathsf{LDE}}$ can be made a circuit of size

[11] The low-degree tests are seemingly redundant as the values of the low-degree extension were computed by the circuit. However, since we don't know that the values are computed correctly, the low-degree

$\mathsf{poly}(s)$ and depth $O(\log s)$. We refer the reader to Section 2.9 for a formal description of our augmented PCP.

The basic idea behind adding the computation of $\mathcal{C}_{\mathsf{LDE}}$ after each layer, is that now the PCP verifier can read a single point in the low-degree extension of a layer, and in some sense, this point contains information about the entire layer. As we argue below, if a random point in the low-degree extension is correct with high probability, then *each* value in the layer is correct with almost the same probability. We elaborate below.

Our analysis. Since our augmented PCP is identical to our base PCP, applied to the augmented circuit $\mathcal{C}'_n$ (as opposed to $\mathcal{C}_n$), the analysis of our base PCP implies that if there exists a (statistically) no-signaling prover that convinces the verifier to accept with probability close to 1, then local consistency holds with probability close to 1. Namely, for any set of queries Q of size at most $k_{max} = \mathsf{polylog}(t)$ and for any subset $Q' \subseteq Q$ of queries corresponding to variables in the tableau of $\mathcal{C}'_n$, the answers to these queries are locally consistent (i.e., they satisfy the constraints imposed by the gates and they satisfy the low-degree tests) with probability close to 1.

We next show how we go from local consistency to global consistency, without increasing the size of k_{max}. To this end, we use the special structure of $\mathcal{C}'_n$; i.e., the fact that it includes all the low-degree extensions and low degree tests.

Fix any layer in $\mathcal{C}_n$. We first argue that if a random point in the low-degree extension of this layer has the correct value with high probability, then the value of every point in the layer is correct with high probability. The idea is the following: Fix any point z in the layer. Consider the line connecting this point to the random point in the low-degree extension. The local consistency condition implies that with high probability the values on this line correspond to a low degree polynomial. Thus, if the value of the point z is incorrect (with significant probability) then most of the points on the line are incorrect (with significant probability), and in particular a random point on the line is incorrect (with significant probability), contradicting our assumption that a random point in the low-degree extension is correct with high probability.

We use the argument above to prove the correctness of the entire computation. The proof is by induction on the depth of the tableau of $\mathcal{C}_n$. In what follows, we denote the probability that local consistency holds by $(1-\epsilon)$, and the reader should think of $\epsilon = \frac{1}{\mathsf{poly}(t)}$. We start with the base case, and claim that each element in the base of the tableau (where the input lies) is correct with probability $1-\epsilon$. This follows from the local consistency condition. Next we claim that if each element in a layer is correct with high probability $(1-\epsilon)$, then any point in the low-degree extension of the layer is correct with probability $(1-\epsilon)^{\mathsf{poly}(s)}$. To this end, we use the analysis where the error increases exponentially with the depth, and we use the fact that the depth of $\mathcal{C}_{\mathsf{LDE}}$ is $O(\log s)$.

For the induction step, we claim that if at layer i a random point in the low-degree extension is correct with some probability p, then a random point in the low-degree extension of layer $i+1$ is correct with probability $\approx p(1-\epsilon)^{\mathsf{poly}(s)}$. Note that this guarantee

tests will be very important in our analysis.

is strong enough for correctness, since by induction we get that a random point in the low-degree extension of the top layer is correct with probability $\approx (1-\epsilon)^{t\cdot\mathsf{poly}(s)}$ which is close to 1 for $\epsilon = \frac{1}{t^2\cdot\mathsf{poly}(s)}$.

To prove the induction step we use conditional probabilities, and condition on the event that the value of a random point in the low-degree extension of layer i is indeed correct. Conditioned on this event, each element in the i-th layer of $\mathcal{C}_n$ is correct with probability $(1-\epsilon)$ (which is the probability in which local consistency holds). Therefore, conditioned on this event, each element in the $i+1$-th layer of $\mathcal{C}_n$ is correct with probability $(1-\epsilon)^3$. This implies that, conditioned on this event, each element in the low-degree extension of the $i+1$-th layer is correct with probability $(1-\epsilon)^{\mathsf{poly}(s)}$. Therefore, without conditioning, the probability that an element in the low-degree extension of the $i+1$-th layer is correct is $p(1-\epsilon)^{\mathsf{poly}(s)}$.

This analysis does not quite work as is, since there is too big of a loss in the correctness probability when going from a random point in the low-degree extension to a point in the layer of $\mathcal{C}_n$. We fix this by reading several random points in the low-degree extension (rather than just one), and we claim that if all of them are correct with some probability then each point in the layer is correct with essentially the same probability (where the loss here is exponentially small). We refer the reader to Section 2.10 for details.

2.3.2 From No-Signaling MIP to a Delegation Scheme

In this section we show that the method of Aiello *et al.* [ABOR00], of using a fully homomorphic (FHE) scheme to convert a 1-round MIP into a 1-round delegation scheme, is *sound* if the underlying MIP is secure against δ-no-signaling provers, where the value of δ affects the security requirement of the FHE scheme.[12]

Let us start by recalling their method. Aiello *et al.* proposed to take any MIP and convert it into the following 1-round delegation scheme: The verifier computes all the queries that the MIP verifier would send to the MIP provers, and sends all of these queries to the prover, each encrypted under a fresh and independent key, using an FHE scheme. The prover then answers on behalf of each MIP prover, where each answer is computed *homomorphically* on the corresponding encrypted query.

As mentioned in the introduction, shortly after this method was introduced, Dwork *et al.* [DLN+04] showed that it may, in general, be insecure. In this work, we show that this method in fact is *secure* if the underlying MIP is sound against δ-no-signaling provers.

In a nutshell, our result is obtained by proving that if there exists a cheating prover P^* that breaks the soundness of the 1-round argument, then this prover can be used to construct a δ-no-signaling prover P^{NS} that breaks the soundness of the MIP scheme.

The prover P^{NS} uses P^* in the obvious way: Given a set of queries $(q_1,\ldots,q_\ell)$ it encrypts these queries using fresh and independent keys, and sends the encrypted queries to P^*; upon receiving encrypted answers, it decrypts these answers and sends the decrypted answers $(a_1,\ldots,a_\ell)$ to the MIP verifier.

[12] Aiello *et al.* originally suggested to use a PCP together with a private information retrieval (PIR) scheme to construct a 1-round delegation scheme.

Clearly this strategy breaks the soundness of the MIP verifier, but we need to argue that it is δ-no-signaling. Indeed, we argue that if P^{NS} is *not* δ-no-signaling then the prover P^* can be used to break the underlying FHE scheme. Loosely speaking, by the definition of δ-no-signaling (see Section 2.4.3), if P^{NS} is *not* δ-no-signaling then there is a subset $S \subset [\ell]$ such that the distribution of the answers $(a_i)_{i\in S}$, conditioned on the corresponding queries $(q_i)_{i\in S}$, depends on the other queries $(q_i)_{i\notin S}$. In other words, these answers give information on the other queries. If this is the case, then indeed one can use P^* to break the FHE scheme.

We note that the above break may take time exponential in the communication complexity of the underlying MIP scheme, since the information obtained from the answers $(a_i)_{i\in S}$, is not necessarily efficiently computable. Therefore, we need to assume that the underlying FHE scheme is secure against adversaries of size $2^{|a_1|+\ldots+|a_\ell|}$. Thus, if we choose the security parameter of the FHE scheme to be polynomially related to the communication complexity, then we need to assume sub-exponential security of the underlying FHE scheme. But one can choose a larger security parameter (resulting in larger communication complexity in the 1-round delegation scheme), and thus relax the security requirement of the FHE scheme. We refer the reader to Section 2.16 for details.

2.4 Preliminaries

2.4.1 Notation

For a vector $a = (a_1, \ldots, a_k)$ and a subset $S \subseteq [k]$, we denote by a_S the sequence of elements of a that are indexed by indices in S, that is, $a_S = (a_i)_{i\in S}$. In general, we denote by a_S a sequence of elements indexed by S, and we denote by a_i the i^{th} coordinate of a vector a.

For a distribution $\mathcal{A}$, we denote by $a \in_R \mathcal{A}$ a random variable distributed according to $\mathcal{A}$ (independently of all other random variables).

We will measure the distance between two distributions by their *statistical distance*, defined as half the l_1-distance between the distributions. We will say that two distributions are δ-*close* if their statistical distance is at most δ.

For a field $\mathbb{F}$ and an integer ℓ, a line L in $\mathbb{F}^\ell$ is an affine function $L : \mathbb{F} \to \mathbb{F}^\ell$. A plain M in $\mathbb{F}^\ell$ is an affine function $M : \mathbb{F}^2 \to \mathbb{F}^\ell$. We say that the line L is orthogonal to the i^{th} coordinate if for every $t_1, t_2 \in \mathbb{F}$, we have $L(t_1)_i = L(t_2)_i$, where $L(t_1)_i, L(t_2)_i$ denote the i^{th} coordinate of the points $L(t_1), L(t_2)$ respectively.

We will sometimes confuse between a set and a multiset. In particular, many times we will refer to a multiset as a set. For example, when we choose a (multi)set of k elements in a certain domain.

We will sometimes write $\Pr_x \Pr_y$ instead of $\Pr_{x,y}$.

2.4.2 Multi-Prover Interactive Proofs

Let $\mathcal{L}$ be a language and let x be an input of length n. In a one-round k-prover interactive proof, k computationally unbounded provers, $P_1, \ldots, P_k$, try to convince a (probabilistic) $\mathsf{poly}(n)$-time verifier, V, that $x \in \mathcal{L}$. The input x is known to all parties.

The proof consists of only one round. Given x and her random string, the verifier generates k queries, $q_1, \ldots, q_k$, one for each prover, and sends them to the k provers. Each prover responds with an answer that depends only on her own individual query. That is, the provers respond with answers $a_1, \ldots, a_k$, where for every i we have $a_i = P_i(q_i)$. Finally, the verifier decides wether to accept or reject based on the answers that she receives (as well as the input x and her random string).

We say that $(V, P_1, \ldots, P_k)$ is a one-round multi-prover interactive proof system (MIP) for $\mathcal{L}$ if the following two properties are satisfied:

1. **Completeness:** For every $x \in \mathcal{L}$, the verifier V accepts with probability 1, after interacting with $P_1, \ldots, P_k$.

2. **Soundness:** For every $x \notin \mathcal{L}$, and any (computationally unbounded, possibly cheating) provers $P_1^*, \ldots, P_k^*$, the verifier V rejects with probability $\geq 1 - \epsilon$, after interacting with $P_1^*, \ldots, P_k^*$, where ϵ is a parameter referred to as the *error* or *soundness* of the proof system.

Important parameters of an MIP are the number of provers, the length of queries, the length of answers, and the error.

2.4.2.1 MIPs with Oracle

We will also consider the model of *one-round k-prover interactive proofs with oracle*, where the verifier V is given access to an oracle that computes some fixed function (that may depend on the language $\mathcal{L}$). We require that all queries, to the oracle and the provers, are done simultaneously.

For every n, let $\phi_n : \{0,1\}^{n'} \to \{0,1\}^{n''}$ be a function (where n', n'' depend on n). We allow the functions ϕ_n to depend on the language $\mathcal{L}$ (but not on the input x).

We define a *one-round multi-prover interactive proof system for $\mathcal{L}$, relative to the oracle $\{\phi_n\}$*, exactly as before, except that now the verifier V is a (probabilistic, $\mathsf{poly}(n)$-time) oracle machine that on input x of length n has free oracle access to the function ϕ_n. The verifier may base her accept/reject decision on queries to the oracle, but the oracle queries are not adaptive, and we do not allow the queries to the provers to depend on the answers of the oracle or the queries to the oracle to depend on the answers of the provers. In other words, we require that all queries, to the oracle and to the provers, are done simultaneously.

We require the same completeness and soundness properties as before.

2.4.3 No-Signaling MIPs

We will consider a variant of the MIP model, where the cheating provers are more powerful. In the MIP model, each prover answers her own query locally, without knowing the queries that were sent to the other provers. The no-signaling model allows each answer to depend on all the queries, as long as for any subset $S \subset [k]$, and any queries q_S for the provers in S, the distribution of the answers a_S, conditioned on the queries q_S, is independent of all the other queries.

Intuitively, this means that the answers a_S do not give the provers in S information about the queries of the provers outside S, except for information that they already have by seeing the queries q_S.

Formally, denote by D the alphabet of the queries and denote by Σ the alphabet of the answers. For every $q = (q_1, \ldots, q_k) \in D^k$, let $\mathcal{A}_q$ be a distribution over Σ^k. We think of $\mathcal{A}_q$ as the distribution of the answers for queries q.

We say that the family of distributions $\{\mathcal{A}_q\}_{q \in D^k}$ is *no-signaling* if for every subset $S \subset [k]$ and every two sequences of queries $q, q' \in D^k$, such that $q_S = q'_S$, the following two random variables are identically distributed:

- a_S, where $a \in_R \mathcal{A}_q$
- a'_S where $a' \in_R \mathcal{A}_{q'}$

If the two distributions are δ-close, rather than identical, we say that the family of distributions $\{\mathcal{A}_q\}_{q \in D^k}$ is *δ-no-signaling*.

An MIP, $(V, P_1, \ldots, P_k)$ for a language $\mathcal{L}$ (possibly, relative to an oracle $\{\phi_n\}$) is said to have soundness ϵ against no-signaling strategies (or provers) if the following (more general) soundness property is satisfied:

2. **Soundness:** For every $x \notin \mathcal{L}$, and any no-signaling family of distributions $\{\mathcal{A}_q\}_{q \in D^k}$, the verifier V rejects with probability $\geq 1 - \epsilon$, where on queries $q = (q_1, \ldots, q_k)$ the answers are given by $(a_1, \ldots, a_k) \in_R \mathcal{A}_q$, and ϵ is the error parameter.

If the property is satisfied for any δ-no-signaling family of distributions $\{\mathcal{A}_q\}_{q \in D^k}$, we say that the MIP has soundness ϵ against δ-no-signaling strategies (or provers).

2.4.4 Probabilistically Checkable Proofs

Let $\mathcal{L}$ be a language and let x be an input of length n. Intuitively, a probabilistically checkable proof (PCP) is a proof for $x \in \mathcal{L}$ that can be verified by reading only a small number of its symbols.

Formally, a proof is a vector of symbols $P \in \Sigma^D$, where Σ denotes the alphabet of symbols and D denotes the set of indices. We think of P also as a function $P : D \to \Sigma$ and hence we think of D as a set of possible queries and we think of Σ as a set of possible answers.

A PCP verifier V is a probabilistic $\mathsf{poly}(n)$-time Turing machine that is given access to the input x, as well as an oracle access to the proof $P : D \to \Sigma$.

Given x and her random string, the verifier generates k queries, $q_1, \ldots, q_k \in D$, and queries the proof P in all these places to get answers $a_1 = P(q_1), \ldots, a_k = P(q_k)$. Finally, the verifier decides wether to accept or reject based on the answers that she receives (as well as the input x and her random string).

We say that V is a PCP verifier for $\mathcal{L}$ if the following two properties are satisfied:

1. **Completeness:** For every $x \in \mathcal{L}$, there exists a proof P, such that, the verifier V accepts with probability 1, after querying P.
2. **Soundness:** For every $x \notin \mathcal{L}$, and any proof $P^* : D \to \Sigma$, the verifier V rejects with probability $\geq 1 - \epsilon$, after querying P^*, where ϵ is a parameter referred to as the *error* or *soundness* of the proof system.

Important parameters of a PCP are the length of proof, the number of queries, the length of answers, and the error.

2.4.4.1 PCPs with Oracle

We will also consider the model of *probabilistically checkable proofs with oracle*, where the verifier V is given access to an additional oracle that computes some fixed function (that may depend on the language $\mathcal{L}$). We require that all queries, to the oracle and to the PCP proof, are done simultaneously.

For every n, let $\phi_n : \{0,1\}^{n'} \to \{0,1\}^{n''}$ be a function (where n', n'' depend on n). We allow the functions ϕ_n to depend on the language $\mathcal{L}$ (but not on the input x).

We define a *probabilistically checkable proof for $\mathcal{L}$, relative to the oracle $\{\phi_n\}$*, exactly as before, except that now the verifier V is a (probabilistic, $\mathsf{poly}(n)$-time) machine with access to two oracles. The first oracle is the PCP proof P, and in addition, on input x of length n, the verifier has free oracle access to the function ϕ_n. The verifier may base her accept/reject decision on queries to the oracle ϕ_n, but the oracle queries are not adaptive, and we do not allow the queries to P to depend on the answers of the oracle ϕ_n or the queries to the oracle ϕ_n to depend on the answers of P. In other words, we require that all queries, to the oracle ϕ_n and to the PCP proof P, are done simultaneously.

We require the same completeness and soundness properties as before.

2.4.5 No-Signaling PCPs

We will now define the new notion of PCP with no-signaling soundness, in analogy to MIP with no-signaling soundness.

We will consider a variant of the PCP model, where the cheating proof is more powerful. In the PCP model, each query is answered locally, without knowing the other queries. In the no-signaling model, we allow each answer to depend on all the queries, as long as for any subset $\{q_1, \ldots, q_d\}$ of queries, the distribution of the answers $(a_1, \ldots, a_d)$, conditioned on the queries $\{q_1, \ldots, q_d\}$, is independent of all the other queries.

Formally, denote by D the alphabet of the queries and denote by Σ the alphabet of the answers. Let k_{max} be some parameter, which is at least the maximal number of

queries made by the verifier to the proof. For every subset $Q = \{q_1, \ldots, q_d\} \subset D$, of size $|Q| = d \leq k_{max}$, let $\mathcal{A}_Q$ be a distribution over Σ^Q. We think of $\mathcal{A}_Q$ as the distribution of the answers for the queries $\{q_1, \ldots, q_d\}$.

We say that the family of distributions $\{\mathcal{A}_Q\}_{Q \subset D, |Q| \leq k_{max}}$ is *no-signaling* if for every $Q \subset D$ of size at most k_{max}, and every subset $S \subset Q$, the following two random variables are identically distributed:

- $a \in_R \mathcal{A}_S$
- a'_S, where $a' \in_R \mathcal{A}_Q$

If the two distributions are δ-close, rather than identical, we say that the family of distributions $\{\mathcal{A}_Q\}_{Q \subset D, |Q| \leq k_{max}}$ is *δ-no-signaling*.

A PCP verifier V for a language $\mathcal{L}$ (possibly, relative to an oracle $\{\phi_n\}$) is said to have soundness ϵ against k_{max}-no-signaling strategies (or proofs) if the following (more general) soundness property is satisfied:

2. **Soundness:** For every $x \notin \mathcal{L}$, and any no-signaling family of distributions $\{\mathcal{A}_Q\}_{Q \subset D, |Q| \leq k_{max}}$, the verifier V rejects with probability $\geq 1 - \epsilon$, where on queries $Q = \{q_1, \ldots, q_k\}$, the answers are given by $a_Q \in_R \mathcal{A}_Q$, and ϵ is the error parameter.

If the property is satisfied for any δ-no-signaling family of distributions $\{\mathcal{A}_Q\}_{Q \subset D, |Q| \leq k_{max}}$, we say that the PCP has soundness ϵ against (k_{max}, δ)-no-signaling strategies (or proofs).

Note that k_{max} is an important parameter. The larger k_{max} is, the more limited the cheating proofs are. We will typically take k_{max} to be significantly larger than the maximal number of queries made by the verifier.

2.4.5.1 A Note on Ordered versus Unordered Sets

The families of distributions $\{\mathcal{A}_Q\}_{Q \subset D, |Q| \leq k_{max}}$ that we consider are defined with *unordered* sets $Q \subset D$. However, sometimes we will have *ordered* sets Q (that is, Q will be a vector of elements); for example, when we need to know which test to apply on which subset of queries, it is important that the set of queries is ordered by the order that the queries were chosen. In these cases, we will abuse notation and denote by Q both the ordered set and the unordered set that corresponds to it. Thus, we will use the notation $\mathcal{A}_Q$ to denote the distribution that corresponds to the unordered set that corresponds to Q. In general, we will sometimes abuse notation between an ordered set and the unordered set that corresponds to it.

2.4.6 Low Degree Extension

Let $\mathbb{F}$ be a field and $H \subset \mathbb{F}$ a subset of the field. Fix an integer $m \in \mathbb{N}$. A basic fact is that for every function $\phi : H^m \to \mathbb{F}$, there exists a unique extension of ϕ into a function $\hat{\phi} : \mathbb{F}^m \to \mathbb{F}$ (which agrees with ϕ on H^m; i.e., $\hat{\phi}|_{H^m} \equiv \phi$), such that $\hat{\phi}$ is an m-variate polynomial of degree at most $|H| - 1$ in each variable. Moreover, for every $x \in H^m$, there

exists a unique m-variate polynomial $\hat{\beta}_x : \mathbb{F}^m \to \mathbb{F}$ of degree $|H| - 1$ in each variable, such that for every function $\phi : H^m \to \mathbb{F}$ it holds that

$$\hat{\phi}(z_1, \ldots, z_m) = \sum_{x \in H^m} \hat{\beta}_x(z_1, \ldots, z_m) \cdot \phi(x).$$

The function $\hat{\phi}$ is called the *low degree extension* of ϕ (with respect to $\mathbb{F}, H, m$).

In the following we assume that all algorithms have access to m, the set H and the field $\mathbb{F}$. We assume that field operations over $\mathbb{F}$ can be computed in time poly-logarithmic in the field size and space that is logarithmic in the field size.

Proposition 2.1 (Cf., e.g., [Rot09, Proposition 3.2.1])**.** *There exists a Turing machine that on input $x \in H^m$, runs in time $\mathsf{poly}(|H|, m, \log|\mathbb{F}|)$ and space $O(\log(|\mathbb{F}|) + \log(m))$, and outputs the polynomial $\hat{\beta}_x : \mathbb{F}^m \to \mathbb{F}$ defined above, represented as an arithmetic circuit over $\mathbb{F}$.*

Moreover, the arithmetic circuit $\hat{\beta}_x$ can be evaluated in time $\mathsf{poly}(|H|, m, \log(|\mathbb{F}|))$ and space $O(\log(|\mathbb{F}|) + \log(m))$. Namely, there exists a Turing machine with the above time and space bounds that given an input pair $(x, z) \in H^m \times \mathbb{F}^m$ outputs $\hat{\beta}_x(z)$.

Proof. Consider the function $\hat{\beta}_x : \mathbb{F}^m \to \mathbb{F}$ defined as:

$$\hat{\beta}_x(z) \stackrel{\text{def}}{=} \prod_{i \in [m]} \prod_{h \in H \setminus \{x_i\}} \frac{z_i - h}{x_i - h}.$$

For every $z \in H^m$ it holds that $\hat{\beta}_x(z) = 1$ if $z = x$ and $\hat{\beta}_x(z) = 0$ otherwise. Thus, for every function $\phi : H^m \to \mathbb{F}$ it holds that $\sum_{x \in H^m} \hat{\beta}_x \cdot \phi(x)$ agrees with ϕ on H^m. Hence, since $\hat{\beta}_x$ has degree $|H| - 1$ in each variable, $\sum_{x \in H^m} \hat{\beta}_x \cdot \phi(x)$ is the (unique) low degree extension of ϕ. □

Proposition 2.2. *Let $\phi : H^m \to \mathbb{F}$ and suppose that ϕ can be evaluated by a Turing Machine in time t and space s. Then, there exists a Turing machine that, given as an input a point $z \in \mathbb{F}^m$, runs in time $|H|^m \left(\mathsf{poly}(|H|, m, \log(|\mathbb{F}|)) + O(t)\right)$ and space $O(m \log(|H|) + s + \log(|\mathbb{F}|))$ and outputs the value $\hat{\phi}(z)$ where $\hat{\phi}$ is the unique low degree extension of ϕ (with respect to $H, \mathbb{F}, m$).*

Proof. The Turing machine computes

$$\hat{\phi}(z) = \sum_{x \in H^m} \hat{\beta}_x(z) \cdot \phi(x)$$

by generating and evaluating $\hat{\beta}_x(z)$ as in Proposition 2.1. □

2.4.7 Public-Key Encryption and Fully Homomorphic Encryption (FHE)

A *public-key encryption* scheme consists of three probabilistic polynomial-time algorithms $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$. The key generation algorithm Gen, when given as input a security parameter 1^τ, outputs a pair $(\mathsf{pk}, \mathsf{sk})$ of public and secret keys. The encryption algorithm, Enc, on input a public key pk and a message $m \in \{0,1\}^{\mathsf{poly}(\tau)}$, outputs a ciphertext $\hat{m}$, and the decryption algorithm, Dec, when given the ciphertext $\hat{m}$ and the secret key sk, outputs the original message m (with overwhelming probability). We allow the decryption process to fail with negligible probability (over the randomness of all algorithms).

Let $S : \mathbb{N} \to \mathbb{N}$ and $\delta : \mathbb{N} \to [0,1]$ be parameters. A public-key encryption scheme has security (S, δ) if for every family of circuits $\{C_\tau\}_{\tau \in \mathbb{N}}$ of size $\mathsf{poly}(S(\tau))$, for all sufficiently large τ and for any two messages $m, m' \in \{0,1\}^{\mathsf{poly}(\tau)}$ such that $|m| = |m'|$,

$$\left| \Pr_{(\mathsf{pk},\mathsf{sk}) \in_R \mathsf{Gen}(1^\tau)} [C_\tau(\mathsf{pk}, \mathsf{Enc}_{\mathsf{pk}}(m)) = 1] - \Pr_{(\mathsf{pk},\mathsf{sk}) \in_R \mathsf{Gen}(1^\tau)} [C_\tau(\mathsf{pk}, \mathsf{Enc}_{\mathsf{pk}}(m')) = 1] \right| < \delta(\tau)$$

where the probability is also over the random coin tosses of Enc.

Fully homomorphic encryption. The tuple $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval})$ is a *fully-homomorphic encryption scheme* if (1) $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ is a public-key encryption scheme, and (2) for every key-pair (pk, sk), the probabilistic polynomial-time algorithm Eval, on input the public-key pk, a circuit $C : \{0,1\}^k \to \{0,1\}^\ell$, where $k, \ell \leq \mathsf{poly}(\tau)$ (and τ is the security parameter), and a ciphertext $\hat{m}$ that is an encryption of a message $m \in \{0,1\}^k$ with respect to pk, outputs a string ψ such that the following two conditions hold:

- **Homomorphic Evaluation:** $\mathsf{Dec}_{\mathsf{sk}}(\psi) = C(m)$, except with negligible probability (over the coins of all algorithms).
- **Compactness:** The length of ψ is polynomial in τ, k and ℓ (and is independent of the size of C).

2.4.8 Interactive Argument Systems

An interactive argument for a language $\mathcal{L}$ consists of a polynomial-time verifier that wishes to verify a statement of the form $x \in \mathcal{L}$, and a prover that helps the verifier to decide. The two parties, given as input $x \in \{0,1\}^n$, interact and at the end of the interaction the verifier either accepts or rejects. We require that if $x \in \mathcal{L}$ then the verifier accepts with high probability but if $x \notin \mathcal{L}$, then no *computationally bounded* prover can convince the verifier to accept with non-negligible (in n) probability.

We focus on 1-round argument systems. Such an argument-system consists of a single message sent from the verifier V to the prover P, followed by a single message sent from the prover to the verifier.

Let $S : \mathbb{N} \to \mathbb{N}$ and $\epsilon : \mathbb{N} \to [0,1]$ be parameters. We say that (V, P) is a one-round argument-system with soundness (S, ϵ) for $\mathcal{L}$ if the following two properties are satisfied:

1. **Completeness:** For every $x \in \mathcal{L}$, the verifier $V(x)$ accepts with overwhelming probability, after interacting with $P(x)$.

2. **Soundness:** For every family of circuits $\{P_n^*\}_{n\in\mathbb{N}}$ of size $\mathsf{poly}(S(n))$, for all sufficiently large $x \notin \mathcal{L}$, the verifier V rejects with probability $\geq 1 - \epsilon(|x|)$, after interacting with $P_{|x|}^*$ on common input x.

2.5 The Base PCP

2.5.1 The PCP Proof

Let $\mathcal{L}$ be a language in $\mathsf{DTISP}(t(n), s(n))$, where $\mathsf{poly}(n) \leq t(n) \leq \mathsf{exp}(n)$ and $\log(n) \leq s(n) \leq \mathsf{poly}(n)$. Let x be an input of length n. Since $\mathcal{L} \in \mathsf{DTISP}(t(n), s(n))$, for any n there is a (fanin 2) Boolean circuit $\mathcal{C}_n$ of size $N = O(t(n)s(n))$ that computes $\mathcal{L}$ on inputs of length n. Moreover, the circuit $\mathcal{C}_n$ is layered, with $O(s(n))$ gates in each layer, such that a child of a gate in layer $i+1$ is either an input variable (or a negation of an input variable) or a gate in layer i. Moreover, there is a deterministic Turing machine of space $O(\log N)$ that on input n outputs the (description of the) circuit $\mathcal{C}_n$.

Without loss of generality, we assume that in the circuit $\mathcal{C}_n$ all negations are on input variables, and that the two children of any gate in the circuit are different (this property can be achieved by duplicating each gate in the circuit twice, increasing the number of gates in each layer by a factor of 2).

Also, we assume that the gates of the circuit are indexed by the numbers $1, \ldots, N$, in an order that agrees with the layers of the circuit. In particular, for every gate, the index of the gate is larger than the indexes of its children. We assume that $1, \ldots, n$ are the indexes of the n input variables and $n+1, \ldots, 2n$ are the indexes of their negations. We assume that the circuit has a special output gate indexed by N whose value represents the decision of whether $x \in \mathcal{L}$ (we do not assume that there are no other output gates). We assume that the Turing machine that outputs the (description of the) circuit $\mathcal{C}_n$ outputs the vertices in the order of their index.

Let $w_1, \ldots, w_N$ be variables in $\{0, 1\}$ that represent the N wires of the circuit $\mathcal{C}_n$, in the order of their index. In particular, for every gate, the variable that represents the output of the gate appears after the variables that represent the inputs for the gate. Also, $w_1, \ldots, w_n$ represent the n input bits, $w_{n+1}, \ldots, w_{2n}$ represent the negations of the n input bits, and w_N represents the output of the circuit.

Let $\varphi_{\mathcal{C}}(w_1, \ldots, w_N)$ be a 3-CNF Boolean formula that checks that $w_1, \ldots, w_N$ is a correct computation of the circuit $\mathcal{C}_n$ (given the input variables and their negations), by checking that the computation of every gate in the circuit is performed correctly (except for negation gates on input variables - and recall that we assume that these are the only negation gates in the circuit). More precisely, for every (non-negation) gate in the circuit, the formula $\varphi_{\mathcal{C}}$ contains four clauses that check that the computation of that gate is performed correctly for every possibility for the inputs for the gate. For example, if w_i represents the output of a conjunction gate with inputs that are represented by w_{i_1} and

w_{i_2}, we will have the following four clauses in $\varphi_{\mathcal{C}}$ (and note that indeed each of them can be written as a clause):

$$\begin{aligned}
&(w_{i_1} = 0) \wedge (w_{i_2} = 0) \rightarrow (w_i = 0),\\
&(w_{i_1} = 0) \wedge (w_{i_2} = 1) \rightarrow (w_i = 0),\\
&(w_{i_1} = 1) \wedge (w_{i_2} = 0) \rightarrow (w_i = 0),\\
&(w_{i_1} = 1) \wedge (w_{i_2} = 1) \rightarrow (w_i = 1).
\end{aligned}$$

We have $\varphi_{\mathcal{C}}(w_1, \ldots, w_N) = 1$ if and only if $w_1, \ldots, w_N$ is a correct computation of the circuit $\mathcal{C}_n$ (assuming that the input variables are given in $w_1, \ldots, w_n$, and their negations are given in $w_{n+1}, \ldots, w_{2n}$).

For a fixed input $x = (x_1, \ldots, x_n)$, let $\varphi_x(w_1, \ldots, w_{2n}, w_N)$ be a 3-CNF Boolean formula that checks that $(w_1, \ldots, w_n) = (x_1, \ldots, x_n)$, $(w_{n+1}, \ldots, w_{2n}) = (\neg x_1, \ldots, \neg x_n)$ and that $w_N = 1$. More precisely, for every $i \in [n]$, the formula φ_x contains a clause that checks that $w_i = x_i$. For example, if $x_i = 0$, we will have the clause $(w_i = 0) \vee (w_i = 0) \vee (w_i = 0)$ that ensures that $w_i = 0$. In the same way, the formula φ_x contains clauses that check that $w_{n+i} = \neg x_i$, and a clause that checks that $w_N = 1$. We have $\varphi_x(w_1, \ldots, w_{2n}, w_N) = 1$ if and only if $(w_1, \ldots, w_n) = (x_1, \ldots, x_n)$, $(w_{n+1}, \ldots, w_{2n}) = (\neg x_1, \ldots, \neg x_n)$, and $w_N = 1$.

Let $\varphi(w_1, \ldots, w_N)$ be the 3-CNF Boolean formula $\varphi_{\mathcal{C}}(w_1, \ldots, w_N) \wedge \varphi_x(w_1, \ldots, w_{2n}, w_N)$. Thus, $\varphi(w_1, \ldots, w_N) = 1$ if and only if $w_1, \ldots, w_N$ is the computation of the circuit $\mathcal{C}_n$ on the input $x = (x_1, \ldots, x_n)$, and $w_N = 1$. Denote by $x_1, \ldots, x_N$ the computation of the circuit $\mathcal{C}_n$ on the input $x = (x_1, \ldots, x_n)$. Thus, $\varphi(w_1, \ldots, w_N) = 1$ if and only if $(w_1, \ldots, w_N) = (x_1, \ldots, x_N)$, and $x_N = 1$.

Note also that since there is a deterministic Turing machine of space $O(\log N)$ that on input n outputs the description of the circuit $\mathcal{C}_n$, there is a deterministic Turing machine of space $O(\log N)$ that on input n outputs the formula $\varphi_{\mathcal{C}}$.

Let $H = \{0, 1, \ldots, \log N - 1\}$ 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, larger than 100). Let $\ell = 3m + 3$. Let $\mathbb{F}$ be a field, such that $4|H|^{10} \leq |\mathbb{F}| \leq 8(\log N)^{10}$.

Since $N = |H|^m$, we can identify $[N]$ and H^m (say, by the lexicographic order on H^m). In what follows we will abuse notation and view $w_1, \ldots, w_N$ and $x_1, \ldots, x_N$ as indexed by $i \in H^m$ (rather than $i \in [N]$). We can hence view $x = (x_1, \ldots, x_N)$ as a function $x : H^m \rightarrow \{0, 1\}$ (given by $x(i) = x_i$, where we identify $[N]$ and H^m).

Define the multi-variate polynomial $X : \mathbb{F}^m \rightarrow \mathbb{F}$ to be the low-degree extension of $x : H^m \rightarrow \{0, 1\}$.

Let $\phi : (H^m)^3 \times \{0, 1\}^3 \rightarrow \{0, 1\}$ be the function where $\phi(i_1, i_2, i_3, b_1, b_2, b_3) = 1$ if and only if the clause $(w_{i_1} = b_1) \vee (w_{i_2} = b_2) \vee (w_{i_3} = b_3)$ appears in φ. Extend ϕ to be a function $\phi : H^{3m+3} \rightarrow \{0, 1\}$ by setting it to be 0 for inputs outside of $H^{3m} \times \{0, 1\}^3$. Let $\hat{\phi} : \mathbb{F}^\ell \rightarrow \mathbb{F}$ be the low-degree extension of ϕ.

Let $\phi_{\mathcal{C}} : (H^m)^3 \times \{0, 1\}^3 \rightarrow \{0, 1\}$ be the function where $\phi_{\mathcal{C}}(i_1, i_2, i_3, b_1, b_2, b_3) = 1$ if and only if the clause $(w_{i_1} = b_1) \vee (w_{i_2} = b_2) \vee (w_{i_3} = b_3)$ appears in $\varphi_{\mathcal{C}}$. Extend $\phi_{\mathcal{C}}$ to be

a function $\phi_{\mathcal{C}} : H^{3m+3} \to \{0,1\}$ by setting it to be 0 for inputs outside of $H^{3m} \times \{0,1\}^3$. Let $\hat{\phi}_{\mathcal{C}} : \mathbb{F}^\ell \to \mathbb{F}$ be the low-degree extension of $\phi_{\mathcal{C}}$.

Let $\phi_x : (H^m)^3 \times \{0,1\}^3 \to \{0,1\}$ be the function where $\phi_x(i_1, i_2, i_3, b_1, b_2, b_3) = 1$ if and only if the clause $(w_{i_1} = b_1) \vee (w_{i_2} = b_2) \vee (w_{i_3} = b_3)$ appears in φ_x. Extend ϕ_x to be a function $\phi_x : H^{3m+3} \to \{0,1\}$ by setting it to be 0 for inputs outside of $H^{3m} \times \{0,1\}^3$. Let $\hat{\phi}_x : \mathbb{F}^\ell \to \mathbb{F}$ be the low-degree extension of ϕ_x.

Since the sets of clauses of $\varphi_{\mathcal{C}}$ and φ_x are disjoint, we have $\hat{\phi} = \hat{\phi}_x + \hat{\phi}_{\mathcal{C}}$.

Recall that there is a deterministic Turing machine of space $O(\log N)$ that on input n outputs the formula $\varphi_{\mathcal{C}}$. Hence, by Proposition 2.2, there is a deterministic Turing machine of space $O(\log N)$ that on input $z \in \mathbb{F}^\ell$ outputs $\hat{\phi}_{\mathcal{C}}(z)$. Since ϕ_x is Boolean valued and is zero on all but a fixed set of $2n+1$ points (specifically, the clauses that verify the correctness of the inputs and output), its low degree extension $\hat{\phi}_x$ can be evaluated on a point $z \in \mathbb{F}^\ell$ in time $n \cdot \mathsf{polylog} N$ (by using Proposition 2.1 and iterating only over the set of $O(n)$ potentially non-zero points).

Since for $x \in \mathcal{L}$ we have $\varphi(x_1, \ldots, x_N) = 1$, every clause that appears in φ is satisfied by $(x_1, \ldots, x_N)$. Therefore, if $x \in \mathcal{L}$, for every $z = (i_1, i_2, i_3, b_1, b_2, b_3) \in (H^m)^3 \times H^3 = H^\ell$, we have

$$\hat{\phi}(z) \cdot (X(i_1) - b_1) \cdot (X(i_2) - b_2) \cdot (X(i_3) - b_3) = 0 \tag{2.1}$$

Let $P_0 : \mathbb{F}^\ell \to \mathbb{F}$ be the multivariate 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 (2.1) implies that if $x \in \mathcal{L}$ then $P_0|_{H^\ell} \equiv 0$. Moreover, the fact that X and $\hat{\phi}$ have 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 $z = (z_1, \ldots, z_\ell) \in \mathbb{F}^\ell$, let

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

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

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

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 if $x \in \mathcal{L}$ then $P_i|_{\mathbb{F}^i \times H^{\ell-i}} \equiv 0$, and in particular $P_\ell \equiv 0$.

The PCP proof for $x \in \mathcal{L}$ consists of $\ell + 1$ multivariate polynomials: The polynomial $X : \mathbb{F}^m \to \mathbb{F}$ 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 is not part of 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.

2.5.1.1 Complexity of the Prover

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

2.5.2 The PCP Verifier, V

The verifier knows the language $\mathcal{L}$, or more precisely, she knows the Turing machine of space $O(\log N)$ that on input n outputs the description of the circuit $\mathcal{C}_n$. The verifier gets an input x of length n and she wants to verify that $x \in \mathcal{L}$ by querying the PCP proof $X, P_0, P_1, \ldots, P_\ell$.

We will first assume that the verifier has access to the correct values of the function $\hat{\phi} : \mathbb{F}^\ell \to \mathbb{F}$. That is, the verifier can get the correct value of $\hat{\phi}(z)$ for free, for as many points $z \in \mathbb{F}^\ell$ as she wants.

Let $k \leq \mathsf{poly}(n)$, such that $4|\mathbb{F}|^4 \leq k \leq N$, be a security parameter. (The restriction $k \leq \mathsf{poly}(n)$ is because we would like the running time of the verifier to be at most $\mathsf{poly}(n)$).

Recall that 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 $L(t) \in \mathbb{F}^\ell$. Recall that 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: a **Low Degree Test for** X; four types of **Low Degree Tests for** P_i; a **Sum Check for** P_i; and a test of **Consistency of** X **and** P_0 (the exact tests are described below). We note that we have four types of low degree tests for P_i, rather than one, just 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 $O(k \cdot |\mathbb{F}|^2)$ random lines, rather than k random lines), 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 the PCP proof passes all of them:

1. **Low Degree Test for** X**:** Choose k random lines $L_1, \ldots, L_k : \mathbb{F} \to \mathbb{F}^m$. For every $L \in \{L_1, \ldots, L_k\}$, query X on all the points $\{L(t)\}_{t \in \mathbb{F}}$, and 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 k random lines $L_1, \ldots, L_k : \mathbb{F} \to \mathbb{F}^\ell$, such that, every line

$L \in \{L_1, \ldots, L_k\}$ satisfies $L(0)_{i+1} = u$. For every $L \in \{L_1, \ldots, L_k\}$, 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 $< 2\ell|H|$.

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 k random lines $L_1, \ldots, L_k : \mathbb{F} \to \mathbb{F}^\ell$, such that, every line $L \in \{L_1, \ldots, L_k\}$ is orthogonal to the $(i+1)^{th}$ coordinate. For every $L \in \{L_1, \ldots, L_k\}$, 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 $< 2\ell|H|$.

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 k random lines $L_1, \ldots, L_k : \mathbb{F} \to \mathbb{F}^\ell$, such that, every line $L \in \{L_1, \ldots, L_k\}$ is orthogonal to the i^{th} coordinate, and satisfies $L(0)_{i+1} = u$. For every $L \in \{L_1, \ldots, L_k\}$, 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 $< 2\ell|H|$.

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 k random lines $L_1, \ldots, L_k : \mathbb{F} \to \mathbb{F}^\ell$, such that, every line $L \in \{L_1, \ldots, L_k\}$ is orthogonal to the $(i+1)^{th}$ coordinate, and satisfies $L(0)_i = u$. For every $L \in \{L_1, \ldots, L_k\}$, 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 $< 2\ell|H|$.

6. **Sum Check for P_i:** For every $i \in \{1, \ldots, \ell\}$, choose k random points in $\mathbb{F}^\ell$. For each of these points, $z = (z_1, \ldots, z_\ell) \in \mathbb{F}^\ell$, 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 k random points in $\mathbb{F}^\ell$. For each of these points, $z = (i_1, i_2, i_3, b_1, b_2, b_3) \in (\mathbb{F}^m)^3 \times \mathbb{F}^3 = \mathbb{F}^\ell$, query P_0 on the point z and X on the points i_1, i_2, i_3, and check that

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

2.5.2.1 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 $6k\ell|\mathbb{F}|^2$. The time complexity of V is $k \cdot \mathsf{polylog}(N) = k \cdot \mathsf{polylog}(t(n))$.

2.5.3 The Relaxed Verifier, V'

We will now define another verifier for the PCP proof $X, P_0, P_1, \ldots, P_\ell$. We will call the new verifier, the *relaxed verifier* with parameter r, such that $1 \leq r < k$, and denote it by V'. As before, V' knows the language $\mathcal{L}$ and the input x, and we assume that she has access to the correct values of the function $\hat{\phi} : \mathbb{F}^\ell \to \mathbb{F}$.

The verifier V' makes the exact same queries as V, but she accepts in some cases where V rejects.

Recall that V repeated every test k times: The Low Degree Test for X was repeated on k different lines in $\mathbb{F}^m$. The four types of Low Degree Tests for each P_i (and for three of these types, for each $u \in \mathbb{F}$), were each repeated on k different lines in $\mathbb{F}^\ell$. The Sum Check for each P_i (for $i \in \{1, \ldots, \ell\}$) was repeated on k different points in $\mathbb{F}^\ell$. The Consistency of X and P_0 was repeated on k different points in $\mathbb{F}^\ell$.

This gives a partition of all the tests made by V into groups, with exactly k tests in each group. The verifier V accepted if all the tests in all the groups passed. The relaxed verifier, V', accepts if in each group of k tests at least $k - r$ of the tests pass, that is, at most r tests fail.

2.6 Soundness of V' versus Soundness of V

In this section we will show that if the verifier V can be fooled to accept $x \notin \mathcal{L}$, with very small probability, then the verifier V' can be fooled to accept $x \notin \mathcal{L}$ with probability very close to 1. Intuitively, this makes sense because the relaxed verifier V' accepts even if she rejects some of the tests, as long as the number of tests rejected in each group of k tests is at most r.

Recall that $k \leq \mathsf{poly}(n)$, such that $4|\mathbb{F}|^4 \leq k \leq N$, is the security parameter of the PCP, and that $1 \leq r < k$ is the parameter of the relaxed verifier V'. Recall that ℓ and $|\mathbb{F}|$ are bounded by $\mathsf{polylog}(N)$.

We will prove the following lemma.

Lemma 2.3. *Assume that V doesn't have soundness ϵ against (k_{max}, δ)-no-signaling strategies, where $\delta < \frac{\epsilon}{8 \cdot |\mathbb{F}|^{6k\ell|\mathbb{F}|^2}}$. Then, V' doesn't have soundness $1 - (10\ell|\mathbb{F}|2^{-r} + 2\delta)/\epsilon$ against (k'_{max}, δ')-no-signaling strategies, where $k'_{max} = k_{max} - 6k\ell|\mathbb{F}|^2$, and $\delta' = 8\delta|\mathbb{F}|^{6k\ell|\mathbb{F}|^2}/\epsilon$.*

Let us first sketch the main techniques that we will use in the proof of the lemma:

The main claim that we will need in order to prove the lemma (Claim 2.3.1), shows that if V, V' choose their queries independently then the probability that V accepts and V' rejects, when all answers are given by a δ-no-signaling family of distributions $\{\mathcal{A}_S\}_{S \subset D, |S| \leq k_{max}}$, is very small. This will be true because for each group of k tests we can first choose the $2k$ tests for both V, V' and only then decide which tests go to V and which ones go to V'. If among the $2k$ tests many are rejected then V rejects with high probability. On the other hand, if among the $2k$ tests only few are rejected then V' always accepts on that group.

We will assume that there exists a δ-no-signaling family of distributions $\{\mathcal{A}_S\}_{S\subset D,|S|\leq k_{max}}$ that fools V with probability larger than ϵ. That is, the verifier V accepts with probability $> \epsilon$, where on queries Q, the answers are given (probabilistically) by $A_Q \in_R \mathcal{A}_Q$. We will construct a δ'-no-signaling family of distributions $\{\mathcal{A}'_S\}_{S\subset D,|S|\leq k'_{max}}$ that fools V' with probability close to 1.

This will be done by fixing a set of queries q for V and answers a_q for the queries in q, such that V accepts on queries q and answers a_q. The queries q will be chosen randomly by the distribution of V, and the answers a_q will be chosen randomly by the distribution $\mathcal{A}_q$, conditioned on the event that V accepts on queries q and answers a_q. The family $\{\mathcal{A}'_S\}$ will be the family $\{\mathcal{A}_S\}$ conditioned on the event that on queries q the answers are a_q.

Formally, for a set S, we denote by $\mathcal{A}_{q\cup S}|_{a_q}$ the distribution of the random element $A \in_R \mathcal{A}_{q\cup S}$, conditioned on the event $A_q = a_q$ (where we assume that the event $A_q = a_q$ occurs with non-zero probability). Since in $\mathcal{A}_{q\cup S}|_{a_q}$ we have that the coordinates indexed by q are fixed to a_q, we think of $\mathcal{A}_{q\cup S}|_{a_q}$, for simplicity of the notations, as a distribution over Σ^S, rather than over $\Sigma^{q\cup S}$, where $\Sigma = \mathbb{F}$ is the alphabet of the answers, (and note that in this distribution the coordinates indexed by $q \cap S$ are fixed to $a_{q\cap S}$). We will define the family of distributions $\{\mathcal{A}'_S\}$ by $\mathcal{A}'_S = \mathcal{A}_{q\cup S}|_{a_q}$.

We assume that for every distribution $\mathcal{A}_S$ (or $\mathcal{A}'_S$) in the family $\{\mathcal{A}_S\}$ (or $\{\mathcal{A}'_S\}$), 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).

2.6.1 Proof of Lemma 2.3

Proof. Assume that V doesn't have soundness ϵ against (k_{max}, δ)-no-signaling strategies.

Thus, for some $x \notin \mathcal{L}$, there exists a δ-no-signaling family of distributions $\{\mathcal{A}_S\}_{S\subset D,|S|\leq k_{max}}$ that fools V with probability larger than ϵ. That is, the verifier V accepts with probability $> \epsilon$, where on queries Q, the answers are given (probabilistically) by $A_Q \in_R \mathcal{A}_Q$.

Let Q be the set of queries chosen randomly by the verifier V and let Q' be the set of queries chosen independently by the verifier V'. Thus, Q, Q' are independent random variables. Let $A \in_R \mathcal{A}_{Q\cup Q'}$ be the (probabilistic) answers for the queries $Q \cup Q'$.

Let $V(Q, A_Q)$ be 1 if V accepts on queries Q and answers A_Q, and 0 otherwise. Let $V'(Q', A_{Q'})$ be 1 if V' accepts on queries Q' and answers $A_{Q'}$, and 0 otherwise. (We assume here that the sets of queries Q, Q' are ordered by the order that the queries were chosen by the verifiers, so that the sets of queries also define which tests are performed on which queries). We denote by $V(Q, A_Q)$ also the event $V(Q, A_Q) = 1$, and in the same way we denote by $V'(Q', A_{Q'})$ also the event $V'(Q', A_{Q'}) = 1$.

Claim 2.3.1.

$$\Pr_{Q,Q'} \Pr_{A\in_R\mathcal{A}_{Q\cup Q'}} [V(Q, A_Q) \wedge \neg V'(Q', A_{Q'})] \leq 5\ell|\mathbb{F}| \cdot 2^{-r}$$

(where r is the parameter of the relaxed verifier V').

Proof. Recall that V and V' repeated every test k times, and that this gives a partition of all the tests performed by V and V' into groups, with exactly k tests in each group (see Section 2.5.3), and the number of groups for each verifier is smaller than $5\ell|\mathbb{F}|$.

Let $Q_{i,j}$ be the set of queries chosen randomly by V in order to perform the j^{th} test in the i^{th} group. Let $Q'_{i,j}$ be the set of queries chosen independently by V' in order to perform the j^{th} test in the i^{th} group. We think of $Q_{i,j}, Q'_{i,j}$ also as tests, rather than just sets of queries. All these tests are performed independently. That is, all the sets in $\{Q_{i,j}\}_{i,j} \cup \{Q'_{i,j}\}_{i,j}$ are independent, as random variables.

Let Q_i be the multiset of tests $\{Q_{i,j}\}_{j\in[k]}$ and let Q'_i be the multiset of tests $\{Q'_{i,j}\}_{j\in[k]}$.

Let $V(Q_i, A_{Q_i})$ be 1 if V accepts all the tests in Q_i (with answers A_{Q_i}), and 0 otherwise. Let $V'(Q'_i, A_{Q'_i})$ be 0 if V' rejects more than r tests in Q'_i (with answers $A_{Q'_i}$), and 1 otherwise. As before, we denote by $V(Q_i, A_{Q_i})$ also the event $V(Q_i, A_{Q_i}) = 1$, and in the same way we denote by $V'(Q'_i, A_{Q'_i})$ also as the event $V'(Q'_i, A_{Q'_i}) = 1$.

Note that if both $V(Q, A_Q)$ and $\neg V'(Q', A_{Q'})$ occur, then there exists i such that V accepts all the tests in Q_i while V' rejects more than r tests in Q'_i. Hence,

$$\Pr_{Q,Q',A}[V(Q, A_Q) \wedge \neg V'(Q', A_{Q'})] \leq \sum_i \Pr_{Q,Q',A}[V(Q_i, A_{Q_i}) \wedge \neg V'(Q'_i, A_{Q'_i})]$$

Thus, it remains to bound $\Pr[V(Q_i, A_{Q_i}) \wedge \neg V'(Q'_i, A_{Q'_i})]$ by 2^{-r}, for every i.

Fix i. Let $W_i = \{\{Q_{i,j}, Q'_{i,j}\}\}_{j\in[k]}$. That is, W_i is the partition of the multiset $Q_i \cup Q'_i$ into pairs $\{Q_{i,j}, Q'_{i,j}\}$, without specifying for each pair which test is $Q_{i,j}$ and which one is $Q'_{i,j}$. Note that we could have chosen Q_i, Q'_i by first choosing W_i and only then specifying which test in each pair is $Q_{i,j}$ and which one is $Q'_{i,j}$.

Let $r(W_i)$ be the number of pairs in W_i with at least one test that is rejected by the verifiers. Note that $r(W_i)$ is a random variable that depends on Q, Q', A, but conditioned on W_i it is independent of Q_i, Q'_i (that is, $r(W_i)$ is independent of the specification which test in each pair is $Q_{i,j}$ and which one is $Q'_{i,j}$).

We can now bound

$$\Pr_{Q,Q',A}[V(Q_i, A_{Q_i}) \wedge \neg V'(Q'_i, A_{Q'_i})] = \mathop{\mathbf{E}}_{W_i, r(W_i)}\left[\Pr_{Q,Q',A}[V(Q_i, A_{Q_i}) \wedge \neg V'(Q'_i, A_{Q'_i}) \mid W_i, r(W_i)]\right]$$

$\Pr[V(Q_i, A_{Q_i}) \wedge \neg V'(Q'_i, A_{Q'_i}) \mid W_i, r(W_i)]$ is bounded as follows:

If $r(W_i) < r$ then $[V(Q_i, A_{Q_i}) \wedge \neg V'(Q'_i, A_{Q'_i})]$ doesn't occur, because $\neg V'(Q'_i, A_{Q'_i})$ doesn't occur (because in order for $\neg V'(Q'_i, A_{Q'_i})$ to occur V' needs to reject at least r tests in Q'_i, which is impossible when $r(W_i) < r$). Hence,

$$\Pr[V(Q_i, A_{Q_i}) \wedge \neg V'(Q'_i, A_{Q'_i}) \mid W_i, (r(W_i) < r)] = 0$$

For $r(W_i) \geq r$,

$$\Pr[V(Q_i, A_{Q_i}) \wedge \neg V'(Q'_i, A_{Q'_i}) \mid W_i, (r(W_i) \geq r)] \leq \Pr[V(Q_i, A_{Q_i}) \mid W_i, (r(W_i) \geq r)] \leq 2^{-r},$$

where the second inequality follows because for each pair $\{Q_{i,j}, Q'_{i,j}\} \in W_i$, each test goes to Q_i with probability $1/2$ (independently at random), so the probability that Q_i gets

none of the rejected tests is $\leq 2^{-r}$ (because when $r(W_i) \geq r$, there are at least r pairs with at least one rejected test in each pair).

We hence have

$$\Pr_{Q,Q',A}[V(Q_i, A_{Q_i}) \wedge \neg V'(Q'_i, A_{Q'_i})] = \mathop{\mathbf{E}}_{W_i, r(W_i)}\left[\Pr_{Q,Q',A}[V(Q_i, A_{Q_i}) \wedge \neg V'(Q'_i, A_{Q'_i}) \mid W_i, r(W_i)]\right]$$

$$\leq 2^{-r}$$

□

We will now proceed with the proof of Lemma 2.3. Recall that we assume that the δ-no-signaling family of distributions $\{\mathcal{A}_S\}_{S \subset D, |S| \leq k_{max}}$ fools V with probability larger than ϵ.

Recall that $\Sigma = \mathbb{F}$ is the alphabet of the answers. Recall that for a vector $a_Q \in \Sigma^Q$, we denote by $\mathcal{A}_{Q \cup Q'}|_{a_Q}$ the distribution of the random element $A \in_R \mathcal{A}_{Q \cup Q'}$, conditioned on the event $A_Q = a_Q$, (where we assume that the event $A_Q = a_Q$ is obtained with non-zero probability (otherwise we define $\mathcal{A}_{Q \cup Q'}|_{a_Q}$ to be an arbitrary fixed distribution)). Since in $\mathcal{A}_{Q \cup Q'}|_{a_Q}$ we have that the coordinates indexed by Q are fixed to a_Q, we think of $\mathcal{A}_{Q \cup Q'}|_{a_Q}$, for simplicity of the notations, as a distribution over $\Sigma^{Q'}$, rather than over $\Sigma^{Q \cup Q'}$ (and note that in this distribution the coordinates indexed by $Q \cap Q'$ are fixed to $a_{Q \cap Q'}$).

Since $\{\mathcal{A}_S\}$ is a δ-no-signaling family of distributions, the distributions of the following two random variables are δ-close:

- $A \in_R \mathcal{A}_{Q \cup Q'}$
- $\tilde{A}$, where the coordinates indexed by Q of $\tilde{A}$ are chosen by $\tilde{A}_Q \in_R \mathcal{A}_Q$, and the coordinates indexed by Q' of $\tilde{A}$ are chosen by $\tilde{A}_{Q'} \in_R \mathcal{A}_{Q \cup Q'}|_{\tilde{A}_Q}$ (and note that on $Q \cap Q'$ the vectors $\tilde{A}_Q, \tilde{A}_{Q'}$ always agree).

Therefore, by Claim 2.3.1,

$$5\ell|\mathbb{F}| \cdot 2^{-r} \geq$$

$$\Pr_{Q,Q'} \Pr_{A \in_R \mathcal{A}_{Q \cup Q'}}[V(Q, A_Q) \wedge \neg V'(Q', A_{Q'})] =$$

$$\mathop{\mathbf{E}}_{Q,Q'} \mathop{\mathbf{E}}_{A \in_R \mathcal{A}_{Q \cup Q'}}[V(Q, A_Q) \cdot (1 - V'(Q', A_{Q'}))] \geq$$

$$\mathop{\mathbf{E}}_{Q} \mathop{\mathbf{E}}_{Q'} \mathop{\mathbf{E}}_{\tilde{A}_Q \in_R \mathcal{A}_Q} \mathop{\mathbf{E}}_{\tilde{A}_{Q'} \in_R \mathcal{A}_{Q \cup Q'}|_{\tilde{A}_Q}}[V(Q, \tilde{A}_Q) \cdot (1 - V'(Q', \tilde{A}_{Q'}))] - \delta =$$

$$\mathop{\mathbf{E}}_{Q} \mathop{\mathbf{E}}_{\tilde{A}_Q \in_R \mathcal{A}_Q}\left[V(Q, \tilde{A}_Q) \cdot \mathop{\mathbf{E}}_{Q'} \mathop{\mathbf{E}}_{\tilde{A}_{Q'} \in_R \mathcal{A}_{Q \cup Q'}|_{\tilde{A}_Q}}[1 - V'(Q', \tilde{A}_{Q'})]\right] - \delta$$

That is,

$$\mathop{\mathbf{E}}_{Q}\mathop{\mathbf{E}}_{\tilde{A}_Q\in_R\mathcal{A}_Q}\left[V(Q,\tilde{A}_Q)\cdot\mathop{\mathbf{E}}_{Q'}\mathop{\mathbf{E}}_{\tilde{A}_{Q'}\in_R\mathcal{A}_{Q\cup Q'}|_{\tilde{A}_Q}}[1-V'(Q',\tilde{A}_{Q'})]\right]\leq 5\ell|\mathbb{F}|\cdot 2^{-r}+\delta \tag{2.2}$$

The following claim shows that we can fix the values of Q and $\tilde{A}_Q$ to specific values q and $\tilde{a}_q$ that satisfy two desired properties. The first property will be used to show that V' is fooled with high probability. The second one will be used to show that the new family of distributions that we will construct is δ'-no-signaling.

Claim 2.3.2. *We can fix a set of queries q, and answers $\tilde{a}_q\in\Sigma^q$, such that:*

1. $$\mathop{\mathbf{E}}_{Q'}\mathop{\mathbf{E}}_{\tilde{A}_{Q'}\in_R\mathcal{A}_{q\cup Q'}|_{\tilde{a}_q}}[1-V'(Q',\tilde{A}_{Q'})]\leq(5\ell|\mathbb{F}|\cdot 2^{-r}+\delta)\cdot\frac{2}{\epsilon}$$

2. $$\Pr_{\tilde{A}_q\in_R\mathcal{A}_q}(\tilde{A}_q=\tilde{a}_q)\geq\frac{\epsilon}{2\cdot|\Sigma|^{|q|}}$$

Proof. Consider the conditional distribution of $(Q,\tilde{A}_Q)\mid V(Q,\tilde{A}_Q)$, that is, the distribution of $(Q,\tilde{A}_Q)$, where $\tilde{A}_Q\in_R\mathcal{A}_Q$, conditioned on the event $V(Q,\tilde{A}_Q)$. Fix $(q,\tilde{a}_q)$ randomly according to this distribution.

By Equation (2.2) and Markov inequality, and since

$$\Pr_{Q}\Pr_{\tilde{A}_Q\in_R\mathcal{A}_Q}V(Q,\tilde{A}_Q)>\epsilon,$$

the first part of the claim occurs with probability larger than $1/2$.

Since $\Pr_Q\Pr_{\tilde{A}_Q\in_R\mathcal{A}_Q}V(Q,\tilde{A}_Q)>\epsilon$ and since the number of possibilities for each $\tilde{a}_q$ is $|\Sigma|^{|q|}$, the second part of the claim occurs with probability larger than $1/2$. □

Fix $q,\tilde{a}_q$ from Claim 2.3.2. Define the family of distributions $\{\mathcal{A}'_S\}_{S\subset D,|S|\leq k'_{max}}$ by

$$\mathcal{A}'_S=\mathcal{A}_{q\cup S}|_{\tilde{a}_q}$$

(where, as before, $\mathcal{A}_{q\cup S}|_{\tilde{a}_q}$ is viewed as a distribution over Σ^S). Note also that $|q|\leq 6k\ell|\mathbb{F}|^2=k_{max}-k'_{max}$.

By the first part of Claim 2.3.2,

$$\mathop{\mathbf{E}}_{Q'}\mathop{\mathbf{E}}_{A'_{Q'}\in_R\mathcal{A}'_{Q'}}V'(Q',A'_{Q'})\geq 1-(10\ell|\mathbb{F}|\cdot 2^{-r}+2\delta)/\epsilon$$

That is, V' is fooled with probability of at least $1-(10\ell|\mathbb{F}|\cdot 2^{-r}+2\delta)/\epsilon$.

It remains to prove that $\{\mathcal{A}'_S\}$ is a δ'-no-signaling family of distributions.

Claim 2.3.3. *$\{\mathcal{A}'_S\}$ is a δ'-no-signaling family of distributions, where $\delta'=8\delta|\Sigma|^{6k\ell|\mathbb{F}|^2}/\epsilon$.*

Proof. Let $S_1 \subset S_2 \subset D$, be such that $|S_2| \leq k'_{max}$. Denote by $(\mathcal{A}'_{S_2})_{S_1}$ and $(\mathcal{A}_{S_2})_{S_1}$ the projections of the distributions $\mathcal{A}'_{S_2}, \mathcal{A}_{S_2}$, respectively, on the coordinates in S_1.

We need to prove that the distributions $\mathcal{A}'_{S_1}$ and $(\mathcal{A}'_{S_2})_{S_1}$ are δ'-close. Without loss of generality, assume that $q \subseteq S_1$. Otherwise, just add q to both S_1, S_2 (this doesn't change the distance between the two distributions because, by the definition of $\mathcal{A}'_{S_1}, \mathcal{A}'_{S_2}$, we just added fixed coordinates to each of the two distribution).

Denote by $\mathcal{A}_{S_1}|_{\tilde{a}_q}$ the distribution of $A \in_R \mathcal{A}_{S_1}$ conditioned on the event $A_q = \tilde{a}_q$, and, in the same way, denote by $\mathcal{A}_{S_2}|_{\tilde{a}_q}$ the distribution of $A \in_R \mathcal{A}_{S_2}$ conditioned on the event $A_q = \tilde{a}_q$, and by $(\mathcal{A}_{S_2})_{S_1}|_{\tilde{a}_q}$ the distribution of $A \in_R (\mathcal{A}_{S_2})_{S_1}$ conditioned on the event $A_q = \tilde{a}_q$.

By the definitions, $\mathcal{A}'_{S_1} = \mathcal{A}_{S_1}|_{\tilde{a}_q}$, and $\mathcal{A}'_{S_2} = \mathcal{A}_{S_2}|_{\tilde{a}_q}$. Thus, we need to prove that $\mathcal{A}_{S_1}|_{\tilde{a}_q}$ and $(\mathcal{A}_{S_2}|_{\tilde{a}_q})_{S_1}$ are δ'-close. Since $(\mathcal{A}_{S_2}|_{\tilde{a}_q})_{S_1} = (\mathcal{A}_{S_2})_{S_1}|_{\tilde{a}_q}$, we need to prove that $\mathcal{A}_{S_1}|_{\tilde{a}_q}$ and $(\mathcal{A}_{S_2})_{S_1}|_{\tilde{a}_q}$ are δ'-close.

Since $\mathcal{A}$ is a δ-no-signaling family, $\mathcal{A}_{S_1}$ and $(\mathcal{A}_{S_2})_{S_1}$ are δ-close.

By the second part of Claim 2.3.2, and since $\mathcal{A}$ is a δ-no-signaling family, we have that

$$\Pr_{A \in_R \mathcal{A}_{S_1}}(A_q = \tilde{a}_q) \geq \frac{\epsilon}{2 \cdot |\Sigma|^{|q|}} - \delta$$

and

$$\Pr_{A \in_R (\mathcal{A}_{S_2})_{S_1}}(A_q = \tilde{a}_q) \geq \frac{\epsilon}{2 \cdot |\Sigma|^{|q|}} - \delta$$

The proof of the claim thus follows by Proposition 2.4, with $\mu = \mathcal{A}_{S_1}$, $\psi = (\mathcal{A}_{S_2})_{S_1}$, and

$$\alpha = \frac{\epsilon}{2 \cdot |\Sigma|^{|q|}} - \delta \geq \frac{\epsilon}{4 \cdot |\Sigma|^{|q|}} \geq \frac{\epsilon}{4 \cdot |\Sigma|^{6k\ell|\mathbb{F}|^2}}$$

□

Proposition 2.4. *Let δ, α be such that $0 < 2\delta < \alpha \leq 1$. Let $\mu, \psi : \Omega \to \mathbb{R}$ be two probability distributions over a finite set Ω, and assume that μ, ψ are δ-close. Let $E \subset \Omega$ be an event, such that, $\mu(E), \psi(E) \geq \alpha$. Denote by μ_E, ψ_E the conditional distributions μ, ψ, conditioned on the event E. Thus, $\mu_E, \psi_E : E \to \mathbb{R}$ are probability distributions over E.*

Then, μ_E, ψ_E are δ'-close, where $\delta' = 2\delta/\alpha$.

Proof. Denote by $\mu' : E \to \mathbb{R}$ and $\psi' : E \to \mathbb{R}$ the restrictions of μ, ψ to E. That is, for every $e \in E$, we have $\mu'(e) = \mu(e)$ and $\psi'(e) = \psi(e)$. Since μ, ψ are δ-close, $\|\mu' - \psi'\|_1 \leq \|\mu - \psi\|_1 \leq 2\delta$, where $\|\cdot\|_1$ denotes the l_1-norm.

Assume without loss of generality $\mu(E) \geq \psi(E)$. That is, $\frac{\psi(E)}{\mu(E)} \leq 1$.

Assume for a contradiction $\|\mu_E - \psi_E\|_1 > 2\delta'$. Then

$$\delta' < \frac{1}{2}\sum_{e \in E} |\mu_E(e) - \psi_E(e)| = \sum_{\{e | \mu_E(e) \geq \psi_E(e)\}} |\mu_E(e) - \psi_E(e)| \leq$$

$$\sum_{\{e | \mu_E(e) \geq \psi_E(e)\}} \left| \mu_E(e) - \frac{\psi(E)}{\mu(E)} \psi_E(e) \right| \leq \sum_{e \in E} \frac{1}{\mu(E)} |\mu(E) \cdot \mu_E(e) - \psi(E) \cdot \psi_E(e)|$$

$$\leq \frac{1}{\alpha} \sum_{e \in E} |\mu(E) \cdot \mu_E(e) - \psi(E) \cdot \psi_E(e)|$$

Since $\mu' = \mu(E) \cdot \mu_E$, and $\psi' = \psi(E) \cdot \psi_E$, we get $\delta' < 2\delta/\alpha$. □

This concludes the proof of Lemma 2.3. □

2.7 Soundness of V' in the Base PCP

In this section we will show that the verifier V' cannot be fooled to accept $x \notin \mathcal{L}$, with probability close to 1.

Recall that $k \leq \mathsf{poly}(n)$, such that $4|\mathbb{F}|^4 \leq k \leq N$, is the security parameter of the PCP, and that $1 \leq r < k$ is the parameter of the relaxed verifier V'. Recall that ℓ and $|\mathbb{F}|$ are bounded by $\mathsf{polylog}(N)$.

We will prove the following lemma.

Lemma 2.5. *Assume that $k_{max} \geq 4sk|\mathbb{F}| + 6k\ell|\mathbb{F}|^2$, where $s = O(s(n))$ is the maximal number of gates in a layer of the circuit $\mathcal{C}_n$. Assume that $\delta < \frac{1}{1000N\ell|\mathbb{F}|}$. Fix $\epsilon = \frac{1}{100N\ell|\mathbb{F}|}$, and note that $\epsilon > 10 \max\left(\delta, \frac{2k}{|\mathbb{F}|^{m-2}}\right)$. Assume $r < \frac{k}{20\ell|\mathbb{F}|}$. Then, V' has soundness $1-\epsilon$ against (k_{max}, δ)-no-signaling strategies.*

The rest of the section is devoted for the proof of Lemma 2.5. From now on, through Section 2.7, fix s, δ, ϵ, r to be as in the statement of Lemma 2.5.

As for the parameter k_{max}, for the proof of Lemma 2.5, we will assume that $k_{max} \geq 4sk|\mathbb{F}| + 6k\ell|\mathbb{F}|^2$. We will assume for a contradiction that for some $x \notin \mathcal{L}$, there exists a δ-no-signaling family of distributions $\{\mathcal{A}_S\}_{S \subset D, |S| \leq k_{max}}$ that fools V' with probability larger than $1-\epsilon$. That is, the verifier V' accepts with probability $> 1-\epsilon$, where on queries Q, the answers are given (probabilistically) by $A \in_R \mathcal{A}_Q$ (see Section 2.7.7).

However, in most parts of Section 2.7, a much weaker requirement $k_{max} \geq 6k\ell|\mathbb{F}|^2$ will suffice. Hence, for the rest of the section we fix $k_{max} \geq 6k\ell|\mathbb{F}|^2$ and denote by $\{\mathcal{A}_S\}_{S \subset D, |S| \leq k_{max}}$ a δ-no-signaling family of distributions that makes V' accept x with probability $> 1-\epsilon$. The requirement that $k_{max} \geq 4sk|\mathbb{F}| + 6k\ell|\mathbb{F}|^2$ will only be used in Section 2.7.7, by Lemma 2.28, Lemma 2.29 and by the proof of Lemma 2.5 (and the requirement will be noted therein).

Recall that we denote by D the domain of the PCP (that is, the alphabet of queries in the PCP). Recall that

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

where $D_X = \mathbb{F}^m$ is viewed as the domain of X, and $D_0, \ldots, D_\ell$ are $\ell + 1$ copies of $\mathbb{F}^\ell$, viewed as the domains of $P_0, \ldots, P_\ell$, respectively.

For a set $S \subset D, |S| \leq k_{max}$, we will view the answers $A \in_R \mathcal{A}_S$ as a function $A : S \to \mathbb{F}$. We can view A also as a partial function $A : D \to \mathbb{F}$, and we denote by $A_X, A_0, \ldots, A_\ell$ the restriction of that partial function to $D_X, D_0, \ldots, D_\ell$, respectively.

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).

2.7.1 Some Immediate Claims

Fix $k_{max}, s, \delta, \epsilon, r$ to be as in the statement of Lemma 2.5. Assume for a contradiction that for some $x \notin \mathcal{L}$, there exists a δ-no-signaling family of distributions $\{\mathcal{A}_S\}_{S \subset D, |S| \leq k_{max}}$ that fools V' with probability larger than $1 - \epsilon$.

We will start by stating an immediate corollary of the fact that $\{\mathcal{A}_S\}$ is a δ-no-signaling family.

Claim 2.5.1. *Let $S \subset D, |S| \leq k_{max}$ be a set generated by some random process. Let $A \in_R \mathcal{A}_S$. Let $f(S, A)$ be a predicate that is satisfied with probability p (where the probability is over S, A). Let S', Q, such that $S' \subseteq Q \subset D, |Q| \leq k_{max}$, be two sets generated by some random process, such that the distribution of S' is identical to the distribution of S. Let $A' \in_R \mathcal{A}_Q$. Then the probability that $f(S', A'_{S'})$ is satisfied is between $p - \delta$ and $p + \delta$, (where the probability is over S', Q, A').*

Proof. Since $\{\mathcal{A}_S\}$ is a δ-no-signaling family, for every fixed sets $s = s' \subseteq q$,

$$\Pr_{A \in_R \mathcal{A}_s}(f(s, A)) = \Pr_{A' \in_R \mathcal{A}_q}(f(s', A'_{s'})) \mp \delta$$

The claim follows by taking expectation over S on the left hand side and expectation over S', Q on the right hand side. □

Next we will state seven immediate corollaries of the fact that V' accepts with probability $> 1-\epsilon$. The following seven claims correspond to the seven different tests performed by the verifier V'. Each claim states that the corresponding test is satisfied with high probability.

Claim 2.5.2. Low Degree Test for X:
Let $L_1, \ldots, L_k : \mathbb{F} \to D_X$ be k random lines. Let $S = \{L_j(t)\}_{j \in [k], t \in \mathbb{F}} \subset D_X$. Let $A \in_R \mathcal{A}_S$. Then, with probability $> 1-\epsilon-\delta$, for at least $k-r$ of the lines $L \in \{L_1, \ldots, L_k\}$, we have that $A \circ L : \mathbb{F} \to \mathbb{F}$ is a univariate polynomial of degree $< m|H|$ (where the probability is over $L_1, \ldots, L_k, A$).

Proof. Note that the set S can be extended to a set Q of queries of the verifier V', where Q is generated by the correct distribution of V', and $S \subset Q$ is the set of queries for the first test performed by V' (that is, the low degree test for X). Let $A' \in_R \mathcal{A}_Q$. Since V' accepts with probability $> 1 - \epsilon$ on queries Q and answers A', and in particular this means that the first test of V' passes with probability $> 1 - \epsilon$, we have that A' satisfies the claim with probability $> 1 - \epsilon$. Formally:

With probability $> 1 - \epsilon$, for at least $k - r$ of the lines $L \in \{L_1, \ldots, L_k\}$, we have that $A' \circ L : \mathbb{F} \to \mathbb{F}$ is a univariate polynomial of degree $< m|H|$ (where the probability is over $L_1, \ldots, L_k, Q, A'$).

Since $\{\mathcal{A}_S\}$ is a δ-no-signaling family, by Claim 2.5.1, the same is satisfied for A, rather than A', with probability $> 1-\epsilon-\delta$, rather than $> 1-\epsilon$. □

Claim 2.5.3. Low Degree Test for P_i (fixed $L(0)_{i+1}$):
Let $i \in \{0,\ldots,\ell-1\}$. Let $u \in \mathbb{F}$. Let $L_1,\ldots,L_k : \mathbb{F} \to D_i$ be k random lines, such that, every line $L \in \{L_1,\ldots,L_k\}$ satisfies $L(0)_{i+1} = u$. Let $S = \{L_j(t)\}_{j\in[k],t\in\mathbb{F}} \subset D_i$. Let $A \in_R \mathcal{A}_S$. Then, with probability $> 1-\epsilon-\delta$, for at least $k-r$ of the lines $L \in \{L_1,\ldots,L_k\}$, we have that $A \circ L : \mathbb{F} \to \mathbb{F}$ is a univariate polynomial of degree $< 2\ell|H|$ (where the probability is over $L_1,\ldots,L_k,A$).

Proof. Similar to the proof of Claim 2.5.2, using the second test performed by V' (the low degree test for P_i, type 1), rather than the first one. □

Claim 2.5.4. Low Degree Test for P_i (orthogonal to the $(i+1)^{th}$ coordinate):
Let $i \in \{0,\ldots,\ell-1\}$. Let $L_1,\ldots,L_k : \mathbb{F} \to D_i$ be k random lines, such that, every line $L \in \{L_1,\ldots,L_k\}$ is orthogonal to the $(i+1)^{th}$ coordinate. Let $S = \{L_j(t)\}_{j\in[k],t\in\mathbb{F}} \subset D_i$. Let $A \in_R \mathcal{A}_S$. Then, with probability $> 1-\epsilon-\delta$, for at least $k-r$ of the lines $L \in \{L_1,\ldots,L_k\}$, we have that $A \circ L : \mathbb{F} \to \mathbb{F}$ is a univariate polynomial of degree $< 2\ell|H|$ (where the probability is over $L_1,\ldots,L_k,A$).

Proof. Similar to the proof of Claim 2.5.2, using the third test performed by V' (the low degree test for P_i, type 2), rather than the first one. □

Claim 2.5.5. Low Degree Test for P_i (fixed $L(0)_{i+1}$; orthogonal to the i^{th} coordinate):
Let $i \in \{1,\ldots,\ell-1\}$. Let $u \in \mathbb{F}$. Let $L_1,\ldots,L_k : \mathbb{F} \to D_i$ be k random lines, such that, every line $L \in \{L_1,\ldots,L_k\}$ is orthogonal to the i^{th} coordinate, and satisfies $L(0)_{i+1} = u$. Let $S = \{L_j(t)\}_{j\in[k],t\in\mathbb{F}} \subset D_i$. Let $A \in_R \mathcal{A}_S$. Then, with probability $> 1-\epsilon-\delta$, for at least $k-r$ of the lines $L \in \{L_1,\ldots,L_k\}$, we have that $A \circ L : \mathbb{F} \to \mathbb{F}$ is a univariate polynomial of degree $< 2\ell|H|$ (where the probability is over $L_1,\ldots,L_k,A$).

Proof. Similar to the proof of Claim 2.5.2, using the fourth test performed by V' (the low degree test for P_i, type 3), rather than the first one. □

Claim 2.5.6. Low Degree Test for P_i (fixed $L(0)_i$; orthogonal to the $(i+1)^{th}$ coordinate):
Let $i \in \{1,\ldots,\ell-1\}$. Let $u \in \mathbb{F}$. Let $L_1,\ldots,L_k : \mathbb{F} \to D_i$ be k random lines, such that, every line $L \in \{L_1,\ldots,L_k\}$ is orthogonal to the $(i+1)^{th}$ coordinate, and satisfies $L(0)_i = u$. Let $S = \{L_j(t)\}_{j\in[k],t\in\mathbb{F}} \subset D_i$. Let $A \in_R \mathcal{A}_S$. Then, with probability $> 1-\epsilon-\delta$, for at least $k-r$ of the lines $L \in \{L_1,\ldots,L_k\}$, we have that $A \circ L : \mathbb{F} \to \mathbb{F}$ is a univariate polynomial of degree $< 2\ell|H|$ (where the probability is over $L_1,\ldots,L_k,A$).

Proof. Similar to the proof of Claim 2.5.2, using the fifth test performed by V' (the low degree test for P_i, type 4), rather than the first one. □

Claim 2.5.7. Sum Check for P_i:
Let $i \in \{1, \ldots, \ell\}$. Let $z_1, \ldots, z_k \in \mathbb{F}^\ell$ be k random points, where $z_j = (z_{j,1}, \ldots, z_{j,\ell}) \in \mathbb{F}^\ell$. Let S^i and S^{i-1} be two copies of the set of points $\{(z_{j,1}, \ldots, z_{j,i-1}, t, z_{j,i+1}, \ldots, z_{j,\ell})\}_{j\in[k], t\in\mathbb{F}} \subset \mathbb{F}^\ell$, 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 $A \in_R \mathcal{A}_S$. Then, with probability $> 1 - \epsilon - \delta$, for at least $k - r$ of the indices $j \in [k]$, the following is satisfied for every $t \in \mathbb{F}$:

$$A_i(z_{j,1}, \ldots, z_{j,i-1}, t, z_{j,i+1}, \ldots, z_{j,\ell}) = \sum_{h\in H} A_{i-1}(z_{j,1}, \ldots, z_{j,i-1}, h, z_{j,i+1}, \ldots, z_{j,\ell}) t^h$$

(where the probability is over $z_1, \ldots, z_k, A$).

Proof. Similar to the proof of Claim 2.5.2, using the sixth test performed by V' (the sum check for P_i), rather than the first one. □

Claim 2.5.8. Consistency of X and P_0:
Let $z_1, \ldots, z_k \in \mathbb{F}^\ell$ be k random points, where $z_j = (i_{j,1}, i_{j,2}, i_{j,3}, b_{j,1}, b_{j,2}, b_{j,3}) \in (\mathbb{F}^m)^3 \times \mathbb{F}^3 = \mathbb{F}^\ell$. Let $S^0 = \{z_j\}_{j\in[k]}$, viewed as a subset of D_0, and let $S^X = \{i_{j,1}, i_{j,2}, i_{j,3}\}_{j\in[k]}$, viewed as a subset of D_X. Let $S = S^0 \cup S^X \subset D$. Let $A \in_R \mathcal{A}_S$. Then, with probability $> 1 - \epsilon - \delta$, for at least $k - r$ of the points $z_j \in \{z_1, \ldots, z_k\}$, the following is satisfied:

$$A_0(z_j) = \hat{\phi}(z_j) \cdot (A_X(i_{j,1}) - b_{j,1}) \cdot (A_X(i_{j,2}) - b_{j,2}) \cdot (A_X(i_{j,3}) - b_{j,3})$$

(where the probability is over $z_1, \ldots, z_k, A$).

Proof. Similar to the proof of Claim 2.5.2, using the seventh test performed by V' (the consistency of X and P_0), rather than the first one. □

2.7.2 Additional Notation

Let $\ell' \geq 0$ be an integer. Let $M : \mathbb{F}^2 \to \mathbb{F}^{\ell'}$ be a plain. For every $t_1 \in \mathbb{F}$, denote by $M(t_1, *) : \mathbb{F} \to \mathbb{F}^{\ell'}$ the line $L : \mathbb{F} \to \mathbb{F}^{\ell'}$ defined by $L(t) = M(t_1, t)$. For every $t_2 \in \mathbb{F}$, denote by $M(*, t_2) : \mathbb{F} \to \mathbb{F}^{\ell'}$ the line $L : \mathbb{F} \to \mathbb{F}^{\ell'}$ defined by $L(t) = M(t, t_2)$.

Let $f : \mathbb{F}^2 \to \mathbb{F}$ be a function. For every $t_1 \in \mathbb{F}$, define $f_{(t_1,*)} : \mathbb{F} \to \mathbb{F}$ by $f_{(t_1,*)}(t) = f(t_1, t)$. For every $t_2 \in \mathbb{F}$, define $f_{(*,t_2)} : \mathbb{F} \to \mathbb{F}$ by $f_{(*,t_2)}(t) = f(t, t_2)$.

2.7.3 Consistency of P_0

We will now give a definition that will be central in the rest of the section. Intuitively, a point z satisfies property $\mathbb{Z}(\epsilon', r')$ if when taking k lines through it, with high probability, for most of these lines, the answers correspond to low degree polynomials that "evaluate" the point z to 0.

Definition 2.6. Property $\mathbb{Z}(\epsilon', r')$:
Let $\epsilon' \geq 0$ and $r' \geq 0$. Let $i \in \{0, \ldots, \ell\}$. Let $z \in D_i$.

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

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 $\mathbb{Z}(\epsilon', r')$ (also denoted $z \in \mathbb{Z}(\epsilon', r')$) if with probability $\geq 1 - \epsilon'$, for at least $k - r'$ of the lines $L \in \{L_1, \ldots, L_k\}$, 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 $L_1, \ldots, L_k, A$).

Our main lemma about property $\mathbb{Z}(\epsilon', r')$ 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 2.7. *For any $z = (z_1, \ldots, z_\ell) \in D_0$, such that, $z_1, \ldots, z_\ell \in H$, we have $z \in \mathbb{Z}(\epsilon', r')$, where $\epsilon' = 8\ell|\mathbb{F}|\epsilon$, and $r' = 8\ell|\mathbb{F}|r$.*

The rest of Subsection 2.7.3 is devoted for the proof of Lemma 2.7.

2.7.3.1 Proof of Lemma 2.7

First, we define a variant of property $\mathbb{Z}(\epsilon', r')$, where the random lines are restricted to be orthogonal to the $(i')^{th}$ coordinate. (We will use this property only for $i' \in \{i, i+1\}$).

Definition 2.8. Property $\mathbb{Z}^{i'}(\epsilon', r')$:
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 $L_1, \ldots, L_k : \mathbb{F} \to D_i$ be k random lines, such that for every $L \in \{L_1, \ldots, L_k\}$, we have $L(0) = z$, and L is orthogonal to the $(i')^{th}$ coordinate. Let $S = \{L_j(t)\}_{j \in [k], t \in \mathbb{F}} \subset D_i$. Let $A \in_R \mathcal{A}_S$.

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 $\mathbb{Z}^{i'}(\epsilon', r')$ (also denoted $z \in \mathbb{Z}^{i'}(\epsilon', r')$) if with probability $\geq 1 - \epsilon'$, for at least $k - r'$ of the lines $L \in \{L_1, \ldots, L_k\}$, 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 $L_1, \ldots, L_k, A$).

Lemma 2.7 will follow easily by Lemma 2.9, Lemma 2.11 and Lemma 2.12.

Lemma 2.9. *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 $z \in \mathbb{Z}^{i+1}(\epsilon_1, r_1)$ then $z \in \mathbb{Z}^i(\epsilon_2, r_2)$, where $\epsilon_2 = \epsilon_1 + 3|\mathbb{F}|\epsilon$, and $r_2 = r_1 + 2|\mathbb{F}|r$.*

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

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

Denote by E the event that for every $j \in [k]$, 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 the event E occurs with probability of at least $1 - \frac{k \cdot 2}{|\mathbb{F}|^{\ell-2}}$.

Let $M_1, \ldots, M_k : \mathbb{F}^2 \to D_i$ be k plains, 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 $A \in_R \mathcal{A}_S$. 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 Proposition 2.10, (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 $< 2\ell|H|$.

Proposition 2.10. *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 $a_1, \ldots, a_d \in \mathbb{F}$, (where $a_1, \ldots, a_d$ are the Lagrange interpolation coefficients), such that for every $t_2 \in \mathbb{F}$, 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 $f_{(1,*)}, \ldots, f_{(d,*)}$ 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 $k - r_2$ of the plains $M \in \{M_1, \ldots, M_k\}$ are good (where the probability is over $L_1, \ldots, L_k, L'_1, \ldots, L'_k, A$). By Proposition 2.10, this implies that with high probability, at least $k - r_2$ of the lines $L' \in \{L'_1, \ldots, L'_k\}$ 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 $L_1, \ldots, L_k, L'_1, \ldots, L'_k, A$).

Claim 2.10.1. *With probability $\geq 1 - \epsilon_1 - 2|\mathbb{F}|\epsilon - 4|\mathbb{F}|\delta - \frac{2k}{|\mathbb{F}|^{\ell-2}}$, for at least $k - r_1 - 2|\mathbb{F}|r$ of the indices $j \in [k]$, we have that M_j is good.*

Proof. For every $t_1 \in \mathbb{F} \setminus \{0\}$, consider the set of lines $\{M_j(t_1, *)\}_{j\in[k]}$ and note that this is a set of k random lines in D_i, such that, every line $L \in \{M_j(t_1, *)\}_{j\in[k]}$ is orthogonal to the i^{th} coordinate, and satisfies $L(0)_{i+1} = z_{i+1}$. Hence, by Claim 2.5.5, using also Claim 2.5.1, with probability $> 1 - \epsilon - 2\delta$, for at least $k - r$ of the indices $j \in [k]$, we have that $A \circ M_j(t_1, *) : \mathbb{F} \to \mathbb{F}$ is a univariate polynomial of degree $< 2\ell|H|$. If, in addition, the event E occurs, then $A^0 \circ M_j(t_1, *) = A \circ M_j(t_1, *)$ and hence $A^0 \circ M_j(t_1, *) : \mathbb{F} \to \mathbb{F}$ is also a univariate polynomial of degree $< 2\ell|H|$.

For every $t_2 \in \mathbb{F} \setminus \{0\}$, consider the set of lines $\{M_j(*, t_2)\}_{j\in[k]}$ and note that this is a set of k random lines in D_i, such that, every line $L \in \{M_j(*, t_2)\}_{j\in[k]}$ is orthogonal to the $(i+1)^{th}$ coordinate, and satisfies $L(0)_i = z_i$. Hence, by Claim 2.5.6, using also Claim 2.5.1, with probability $> 1 - \epsilon - 2\delta$, for at least $k - r$ of the indices $j \in [k]$, we have

that $A \circ M_j(*, t_2) : \mathbb{F} \to \mathbb{F}$ is a univariate polynomial of degree $< 2\ell|H|$. If, in addition, the event E occurs, then $A^0 \circ M_j(*, t_2) = A \circ M_j(*, t_2)$ and hence $A^0 \circ M_j(*, t_2) : \mathbb{F} \to \mathbb{F}$ is also a univariate polynomial of degree $< 2\ell|H|$.

Consider the set of lines $\{M_j(*, 0)\}_{j \in [k]}$ and note that $M_j(*, 0) = L_j$. Since $z \in \mathbb{Z}^{i+1}(\epsilon_1, r_1)$, and using also Claim 2.5.1, with probability $\geq 1 - \epsilon_1 - \delta$, for at least $k - r_1$ of the indices $j \in [k]$, we have that $A^0 \circ M_j(*, 0) : \mathbb{F} \to \mathbb{F}$ is a univariate polynomial of degree $< 2\ell|H|$.

Recall also that the event E occurs with probability of at least $1 - \frac{2k}{|\mathbb{F}|^{\ell-2}}$.

Adding up all this, by the union bound, we obtain that with probability $\geq 1 - \epsilon_1 - 2|\mathbb{F}|\epsilon - 4|\mathbb{F}|\delta - \frac{2k}{|\mathbb{F}|^{\ell-2}}$, for at least $k - r_1 - 2|\mathbb{F}|r$ of the indices $j \in [k]$, 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 $\geq 1 - \epsilon_1 - 2|\mathbb{F}|\epsilon - 4|\mathbb{F}|\delta - \frac{2k}{|\mathbb{F}|^{\ell-2}}$, for at least $k - r_1 - 2|\mathbb{F}|r$ of the indices $j \in [k]$, we have that M_j is good. □

By Proposition 2.10, Claim 2.10.1 implies that with probability $\geq 1 - \epsilon_1 - 2|\mathbb{F}|\epsilon - 4|\mathbb{F}|\delta - \frac{2k}{|\mathbb{F}|^{\ell-2}} > 1 - \epsilon_2 + \delta$, at least $k - r_2$ of the lines $L' \in \{L'_1, \ldots, L'_k\}$ 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 $L_1, \ldots, L_k, L'_1, \ldots, L'_k, A$). Thus, using Claim 2.5.1, $z \in \mathbb{Z}^i(\epsilon_2, r_2)$.

This concludes the proof of Lemma 2.9. □

Lemma 2.11. *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 $z \in \mathbb{Z}^{i+1}(\epsilon_1, r_1)$ then $z \in \mathbb{Z}(\epsilon_2, r_2)$, where $\epsilon_2 = \epsilon_1 + 3|\mathbb{F}|\epsilon$, and $r_2 = r_1 + 2|\mathbb{F}|r$.*

Proof. Similar to the proof of Lemma 2.9, except that we let $L'_1, \ldots, L'_k : \mathbb{F} \to D_i$ be k random lines, such that for every $L' \in \{L'_1, \ldots, L'_k\}$, we have $L'(0) = z$, (without the requirement that L' is orthogonal to the i^{th} coordinate), and we use Claim 2.5.3 and Claim 2.5.4, rather than Claim 2.5.5 and Claim 2.5.6, in the proof for the equivalent of Claim 2.10.1. □

Lemma 2.12. *Let $\epsilon_1 \geq 0$. Let $r_1 \geq 0$. Let $i \in \{1, \ldots, \ell\}$. 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 $\mathbb{Z}^i(\epsilon_1, r_1)$. Then the point z, viewed as a point in D_{i-1}, satisfies property $\mathbb{Z}^i(\epsilon_2, r_2)$, where $\epsilon_2 = \frac{\epsilon_1}{1-\gamma} + 2|\mathbb{F}|\epsilon$, and $r_2 = \frac{r_1}{1-\gamma} + |\mathbb{F}|r$, and $\gamma = \sqrt{\frac{|H|}{|\mathbb{F}|}}$.*

Proof. Assume that for every $t \in \mathbb{F}$, the point $z(t)$, viewed as a point in D_i, satisfies property $\mathbb{Z}^i(\epsilon_1, r_1)$.

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

Denote by E the event that for every $j \in [k]$, the line L_j is in a general position (as a line in $\mathbb{F}^\ell$), that is, it's image is not a single point. Note that the event E occurs with probability of at least $1 - \frac{k}{|\mathbb{F}|^{\ell-1}}$.

Let $M_1, \ldots, M_k : \mathbb{F}^2 \to \mathbb{F}^\ell$ be k plains, 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 $\{M_j(t_1, t_2)\}_{j\in[k],t_1,t_2\in\mathbb{F}} \subset \mathbb{F}^\ell$, 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 $A \in_R \mathcal{A}_S$. 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 $t \in \mathbb{F}$,
$$A^0_i(M_j(t_1, t)) = \sum_{h\in H} A^0_{i-1}(M_j(t_1, h))t^h$$
2. For at least $|H|$ values $t_2 \in \mathbb{F}$, the function $A^0_i \circ M_j(*, t_2) : \mathbb{F} \to \mathbb{F}$ is a univariate polynomial of degree $< 2\ell|H|$.

By Proposition 2.13, (applied with $f = A^0_i \circ M_j$, $f' = A^0_{i-1} \circ M_j$ and $d = 2\ell|H|$), if M_j is good then for every $t_2 \in H$, the function $A^0_{i-1} \circ M_j(*, t_2) : \mathbb{F} \to \mathbb{F}$ is a univariate polynomial of degree $< 2\ell|H|$.

Proposition 2.13. *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 $k-r_2$ of the plains $M\in\{M_1,\ldots,M_k\}$ are good (where the probability is over $L_1,\ldots,L_k,A$).

Claim 2.13.1. *With probability* $\geq 1-|\mathbb{F}|\epsilon-2|\mathbb{F}|\delta-\frac{k}{|\mathbb{F}|^{\ell-1}}-\frac{\epsilon_1+\delta}{1-\gamma}$, *for at least* $k-|\mathbb{F}|r-\frac{r_1}{1-\gamma}$ *of the indices* $j\in[k]$, *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 $j\in[k]$, 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 $\{M_j(t_1,0)\}_{j\in[k]}$ and note that this is a set of k 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). Note that in Claim 2.5.7, the i^{th} coordinate of each random point is ignored. Hence, by Claim 2.5.7, using also Claim 2.5.1, with probability $>1-\epsilon-2\delta$, for at least $k-r$ of the indices $j\in[k]$, the following is satisfied for every $t\in\mathbb{F}$:

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

If, in addition, the event E occurs, then for every $t\in\mathbb{F}$, we have that, $A_i^0(M_j(t_1,t))=A_i(M_j(t_1,t))$ and $A_{i-1}^0(M_j(t_1,t))=A_{i-1}(M_j(t_1,t))$ and hence

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

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

Recall that the event E occurs with probability of at least $1 - \frac{k}{|\mathbb{F}|^{\ell-1}}$.

Thus, by the union bound, with probability $> 1 - |\mathbb{F}|\epsilon - 2|\mathbb{F}|\delta - \frac{k}{|\mathbb{F}|^{\ell-1}}$, for at least $k - |\mathbb{F}|r$ of the indices $j \in [k]$, the following is satisfied for every $t_1 \in \mathbb{F}$ and every $t \in \mathbb{F}$:

$$A_i^0(M_j(t_1, t)) = \sum_{h \in H} A_{i-1}^0(M_j(t_1, h))t^h \tag{2.3}$$

For every $t_2 \in \mathbb{F}$, consider the set of lines $\{M_j(*, t_2)\}_{j\in[k]}$ and note that this is a set of k random lines, such that for every $L \in \{M_j(*, t_2)\}_{j\in[k]}$, we have $L(0) = z(t_2)$, and L is orthogonal to the i^{th} coordinate. Since $z(t_2)$, viewed as a point in D_i, satisfies property $\mathbb{Z}^i(\epsilon_1, r_1)$, and using also Claim 2.5.1, with probability $\geq 1 - \epsilon_1 - \delta$, for at least $k - r_1$ of the indices $j \in [k]$, 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 Proposition 2.14, applied with $\alpha = \epsilon_1 + \delta$, we obtain the following for any $\gamma < 1$:

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

Proposition 2.14. *Let $\{E_t\}_{t\in\mathbb{F}}$ be a set of events, such that, for every $t \in \mathbb{F}$, $\Pr(E_t) \geq 1 - \alpha$. Then, for any $\gamma < 1$, with probability of at least $1 - \frac{\alpha}{1-\gamma}$, at least $\gamma|\mathbb{F}|$ events in $\{E_t\}_{t\in\mathbb{F}}$ occur.*

Proof. Let I_t be the characteristic function of the event $\neg E_t$. 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 $\{E_t\}_{t\in\mathbb{F}}$ occur. □

Thus, with probability $\geq 1 - \frac{\epsilon_1+\delta}{1-\gamma}$, for at least $\gamma|\mathbb{F}|$ values $t_2 \in \mathbb{F}$ we have that for at most r_1 of the indices $j \in [k]$, 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 $[k]$ 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 $\geq 1 - \frac{\epsilon_1+\delta}{1-\gamma}$, for at most $\frac{\gamma|\mathbb{F}|r_1}{\gamma|\mathbb{F}|-|H|}$ indices $j \in [k]$ we have that for less than $|H|$ 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 (2.3), by the union bound, with probability $> 1 - |\mathbb{F}|\epsilon - 2|\mathbb{F}|\delta - \frac{k}{|\mathbb{F}|^{\ell-1}} - \frac{\epsilon_1+\delta}{1-\gamma}$, for at least $k - |\mathbb{F}|r - \frac{\gamma|\mathbb{F}|r_1}{\gamma|\mathbb{F}|-|H|}$ of the indices $j \in [k]$, 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 $\geq 1-|\mathbb{F}|\epsilon-2|\mathbb{F}|\delta-\frac{k}{|\mathbb{F}|^{\ell-1}}-\frac{\epsilon_1+\delta}{1-\gamma}$, for at least $k-|\mathbb{F}|r-\frac{\gamma|\mathbb{F}|r_1}{\gamma|\mathbb{F}|-|H|}$ of the indices $j\in[k]$, we have that M_j is good. In particular, for $\gamma=\sqrt{\frac{|H|}{|\mathbb{F}|}}$, we have that with probability $\geq 1-|\mathbb{F}|\epsilon-2|\mathbb{F}|\delta-\frac{k}{|\mathbb{F}|^{\ell-1}}-\frac{\epsilon_1+\delta}{1-\gamma}$, for at least $k-|\mathbb{F}|r-\frac{r_1}{1-\gamma}$ of the indices $j\in[k]$, we have that M_j is good. □

By Proposition 2.13, Claim 2.13.1 implies that with probability $\geq 1-|\mathbb{F}|\epsilon-2|\mathbb{F}|\delta-\frac{k}{|\mathbb{F}|^{\ell-1}}-\frac{\epsilon_1+\delta}{1-\gamma}>1-\epsilon_2+\delta$, at least $k-r_2$ of the indices $j\in[k]$ satisfy that for every $t_2\in H$, the function $A^0_{i-1}\circ M_j(*,t_2):\mathbb{F}\to\mathbb{F}$ is a univariate polynomial of degree $<2\ell|H|$, (where the probability is over $L_1,\ldots,L_k,A$).

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

With probability $>1-\epsilon_2+\delta$, at least $k-r_2$ of the indices $j\in[k]$ satisfy that the function $A^0_{i-1}\circ M_j(*,t_2):\mathbb{F}\to\mathbb{F}$ is a univariate polynomial of degree $<2\ell|H|$. Thus, using Claim 2.5.1, the point z, viewed as a point in D_{i-1}, satisfies property $\mathbb{Z}^i(\epsilon_2,r_2)$.

This concludes the proof of Lemma 2.12. □

Combining Lemma 2.12 and Lemma 2.9, we obtain the following lemma.

Lemma 2.15. *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 $\mathbb{Z}^i(\epsilon_1,r_1)$. Then the point z, viewed as a point in D_{i-1}, satisfies property $\mathbb{Z}^{i-1}(\epsilon_2,r_2)$, where $\epsilon_2=\frac{\epsilon_1}{1-\gamma}+5|\mathbb{F}|\epsilon$, and $r_2=\frac{r_1}{1-\gamma}+3|\mathbb{F}|r$, and $\gamma=\sqrt{\frac{|H|}{|\mathbb{F}|}}$.*

Proof. Follows by applying Lemma 2.12 and then Lemma 2.9. □

We can now prove Lemma 2.7.

Proof. 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 $\mathbb{Z}^\ell(\epsilon_\ell,r_\ell)$, where $\epsilon_\ell=0$ and $r_\ell=0$.

By inductive application of Lemma 2.15, for any $i\in\{1,\ldots,\ell-1\}$, any point $z=(z_1,\ldots,z_\ell)\in D_i$, such that, $z_{i+1},\ldots,z_\ell\in H$, satisfies property $\mathbb{Z}^i(\epsilon_i,r_i)$, where $\epsilon_i=\frac{\epsilon_{i+1}}{1-\gamma}+5|\mathbb{F}|\epsilon$ and $r_i=\frac{r_{i+1}}{1-\gamma}+3|\mathbb{F}|r$, and $\gamma=\sqrt{\frac{|H|}{|\mathbb{F}|}}$.

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

Hence, by Lemma 2.12, any point $z=(z_1,\ldots,z_\ell)\in D_0$, such that, $z_1,\ldots,z_\ell\in H$, satisfies property $\mathbb{Z}^1(\epsilon_0,r_0)$, where $\epsilon_0=\frac{\epsilon_1}{1-\gamma}+2|\mathbb{F}|\epsilon<7\ell|\mathbb{F}|\epsilon$, and $r_0=\frac{r_1}{1-\gamma}+|\mathbb{F}|r<7\ell|\mathbb{F}|r$.

Finally, by Lemma 2.11, any point $z=(z_1,\ldots,z_\ell)\in D_0$, such that, $z_1,\ldots,z_\ell\in H$, satisfies property $\mathbb{Z}(\epsilon',r')$, where $\epsilon'<8\ell|\mathbb{F}|\epsilon$, and $r'<8\ell|\mathbb{F}|r$. □

2.7.4 Consistency of X

In this subsection we will show that, intuitively, 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 correspond to low degree polynomials that "evaluate" the point z to v.

This is stated formally in Lemma 2.18, Lemma 2.19 and Lemma 2.20. The main goal of the subsection is to prove Lemma 2.19 and Lemma 2.20 (their statements could be read before reading the rest of the subsection). To prove these lemmas, we will need to first prove Lemma 2.16 and Lemma 2.18.

Lemma 2.16. *Let $\epsilon' = 3|\mathbb{F}|\epsilon$. Let $r' = 2|\mathbb{F}|r$. Let $z \in D_X$. Let $L_1, \ldots, L_k, L'_1, \ldots, L'_k : \mathbb{F} \to D_X$ be $2k$ random lines, such that for every $L \in \{L_1, \ldots, L_k, L'_1, \ldots, L'_k\}$, we have $L(0) = z$. Let $S' = \{L_j(t)\}_{j\in[k],t\in\mathbb{F}} \cup \{L'_j(t)\}_{j\in[k],t\in\mathbb{F}} \subset D_X$. Let $A \in_R \mathcal{A}_{S'}$.*

For any $v \in \mathbb{F}$, define $A^v : S' \to \mathbb{F}$ by $A^v(z') = A(z')$ for $z' \neq z$ and $A^v(z) = v$.

Then, with probability $\geq 1 - \epsilon'$, for at least $k - r'$ of the indices $j \in [k]$, there exists $v \in \mathbb{F}$, 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 $< m|H|$ (where the probability is over $L_1, \ldots, L_k, L'_1, \ldots, L'_k, A$).

Proof. Let $L_1, \ldots, L_k, L'_1, \ldots, L'_k : \mathbb{F} \to D_X$ be $2k$ random lines, such that for every $L \in \{L_1, \ldots, L_k, L'_1, \ldots, L'_k\}$, we have $L(0) = z$.

Denote by E the event that for every $j \in [k]$, 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 $D_X = \mathbb{F}^m$). Note that the event E occurs with probability of at least $1 - \frac{k\cdot 2}{|\mathbb{F}|^{m-2}}$.

Let $M_1, \ldots, M_k : \mathbb{F}^2 \to D_X$ be k plains, 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 $S = \{M_j(t_1, t_2)\}_{j\in[k],t_1,t_2\in\mathbb{F}} \subset D_X$. Let $A \in_R \mathcal{A}_S$. 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 $< m|H|$.

By Proposition 2.17, (applied with $f = A \circ M_j$ and $d = m|H|$), if the event E occurs and M_j is good 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 $< m|H|$.

Proposition 2.17. *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 Proposition 2.10, applied on the function f^v (with variables t_1, t_2 switched). □

We will show that with high probability, at least $k - r'$ of the plains $M \in \{M_1, \ldots, M_k\}$ are good (where the probability is over $L_1, \ldots, L_k, L'_1, \ldots, L'_k, A$). By Proposition 2.17, this implies that with high probability, for at least $k - r'$ of the indices $j \in [k]$, 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 $< m|H|$ (where the probability is over $L_1, \ldots, L_k, L'_1, \ldots, L'_k, A$).

Claim 2.17.1. *With probability $\geq 1 - 2|\mathbb{F}|\epsilon - 4|\mathbb{F}|\delta$, for at least $k - 2|\mathbb{F}|r$ of the indices $j \in [k]$, we have that M_j is good.*

Proof. For every $t_1 \in \mathbb{F} \setminus \{0\}$, consider the set of lines $\{M_j(t_1, *)\}_{j \in [k]}$ and note that this is a set of k random lines in D_X. Hence, by Claim 2.5.2, using also Claim 2.5.1, with probability $> 1 - \epsilon - 2\delta$, for at least $k - r$ of the indices $j \in [k]$, 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 $\{M_j(*, t_2)\}_{j \in [k]}$ and note that this is a set of k random lines in D_X. Hence, by Claim 2.5.2, using also Claim 2.5.1, with probability $> 1 - \epsilon - 2\delta$, for at least $k - r$ of the indices $j \in [k]$, 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 $\geq 1 - 2|\mathbb{F}|\epsilon - 4|\mathbb{F}|\delta$, for at least $k - 2|\mathbb{F}|r$ of the indices $j \in [k]$, 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 $\geq 1 - 2|\mathbb{F}|\epsilon - 4|\mathbb{F}|\delta$, for at least $k - 2|\mathbb{F}|r$ of the indices $j \in [k]$, we have that M_j is good. □

By Proposition 2.17, and since the event E occurs with probability of at least $1 - \frac{k \cdot 2}{|\mathbb{F}|^{m-2}}$ Claim 2.17.1 implies that with probability $\geq 1 - 2|\mathbb{F}|\epsilon - 4|\mathbb{F}|\delta - \frac{2k}{|\mathbb{F}|^{m-2}} > 1 - \epsilon' + \delta$, for at least $k - r'$ of the indices $j \in [k]$, 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 $< m|H|$ (where the probability is over $L_1, \ldots, L_k, L'_1, \ldots, L'_k, A$). Thus, using Claim 2.5.1, Lemma 2.16 follows. □

Lemma 2.18. *Let $r' = 20|\mathbb{F}|r$. Let $\epsilon' = 4|\mathbb{F}|\epsilon$. Let $z \in D_X$. Let $L_1, \ldots, L_{2k} : \mathbb{F} \to D_X$ be $2k$ random lines, such that for every $L \in \{L_1, \ldots, L_{2k}\}$, we have $L(0) = z$. Let $S = \{L_j(t)\}_{j\in[2k],t\in\mathbb{F}} \subset D_X$. Let $A \in_R \mathcal{A}_S$.*

For any $v \in \mathbb{F}$, define $A^v : S \to \mathbb{F}$ by $A^v(z') = A(z')$ for $z' \neq z$ and $A^v(z) = v$.

Then, with probability $\geq 1 - \epsilon'$, there exists $v \in \mathbb{F}$, such that, for at least $2k - r'$ of the indices $j \in [2k]$, $A^v \circ L_j : \mathbb{F} \to \mathbb{F}$ is a univariate polynomial of degree $< m|H|$ (where the probability is over $L_1, \ldots, L_{2k}, A$).

Proof. Let $L_1, \ldots, L_{2k} : \mathbb{F} \to D_X$ be $2k$ random lines, such that for every $L \in \{L_1, \ldots, L_{2k}\}$, we have $L(0) = z$. Let $S = \{L_j(t)\}_{j\in[2k],t\in\mathbb{F}} \subset D_X$. Let $A \in_R \mathcal{A}_S$. For any $v \in \mathbb{F}$, define $A^v : S \to \mathbb{F}$ by $A^v(z') = A(z')$ for $z' \neq z$ and $A^v(z) = v$.

Denote by E the event that there exists $v \in \mathbb{F}$, such that, for at least $2k - r'$ of the indices $j \in [2k]$, $A^v \circ L_j : \mathbb{F} \to \mathbb{F}$ is a univariate polynomial of degree $< m|H|$. We will show that $\Pr(E) \geq 1 - \epsilon'$, as needed (where the probability is over $L_1, \ldots, L_{2k}, A$).

Partition $L_1, \ldots, L_{2k}$ randomly into $L'_1, \ldots, L'_k$ and $L''_1, \ldots, L''_k$. Denote by E' the event that for at least $k - 2|\mathbb{F}|r$ of the indices $j \in [k]$, there exists $v \in \mathbb{F}$, 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 Lemma 2.16, $\Pr(E') \geq 1 - 3|\mathbb{F}|\epsilon$.

Claim 2.18.1. $\Pr(E' \mid \neg E) \leq 2^{-|\mathbb{F}|r/4}$

Proof. For every $v \in \mathbb{F}$, let J_v be the set of indices $j \in [2k]$, 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 the event $\neg E$ occurs then for every $v \in \mathbb{F}$, $|J_v| < 2k - r'$. Denote by J the largest set J_v and by $\bar{J}$ the complement of J in $[2k]$. Thus, if the event $\neg E$ occurs then $|\bar{J}| > r'$.

Given the sets $\{J_v\}_{v\in\mathbb{F}}$, the probability that E' occurs is the probability that when partitioning $[2k]$ randomly into k pairs, for at least $k - 2|\mathbb{F}|r$ 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 $k' = r'/2$ different indices $j_1, \ldots, j_{k'}$ in $\bar{J}$. Match the indices $j_1, \ldots, j_{k'}$ one by one, each to a random index in $[2k]$ that was still not chosen. Say that $j_t \in \{j_1, \ldots, j_{k'}\}$ 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 $[2k]$ into k pairs. Note that the probability for an index j_t to be good is at most $\frac{k}{2k-r'} < 0.51$, independently of all previous choices of indices. Thus, the probability that at least $k' - 2|\mathbb{F}|r$ indices $j_t \in \{j_1, \ldots, j_{k'}\}$ are good is at most $k' \cdot \binom{k'}{2|\mathbb{F}|r} \cdot 0.51^{k'-2|\mathbb{F}|r} < 2^{-|\mathbb{F}|r/4}$.

Therefore, $\Pr(E' \mid \neg E) < 2^{-|\mathbb{F}|r/4}$. □

We can now bound,

$$1 - 3|\mathbb{F}|\epsilon \leq \Pr(E') \leq \Pr(E' \mid \neg E) + \Pr(E) < \Pr(E) + 2^{-|\mathbb{F}|r/4}$$

Thus,

$$\Pr(E) > 1 - 3|\mathbb{F}|\epsilon - 2^{-|\mathbb{F}|r/4} > 1 - 4|\mathbb{F}|\epsilon$$

□

Lemma 2.19. *Let $r' = 20|\mathbb{F}|r$. Let $\epsilon' = 5|\mathbb{F}|\epsilon$. Let $z \in D_X$. Let $L_1, \ldots, L_k : \mathbb{F} \to D_X$ be k random lines, such that for every $L \in \{L_1, \ldots, L_k\}$, we have $L(0) = z$. Let $S = \{L_j(t)\}_{j\in[k],t\in\mathbb{F}} \subset D_X$. Let $A \in_R \mathcal{A}_S$.*

For any $v \in \mathbb{F}$, define $A^v : S \to \mathbb{F}$ by $A^v(z') = A(z')$ for $z' \neq z$ and $A^v(z) = v$.

Then, with probability $\geq 1 - \epsilon'$, there exists $v \in \mathbb{F}$, such that, for at least $k - r'$ of the indices $j \in [k]$, $A^v \circ L_j : \mathbb{F} \to \mathbb{F}$ is a univariate polynomial of degree $< m|H|$ (where the probability is over $L_1, \ldots, L_k, A$).

Proof. Follows immediately by Lemma 2.18 and Claim 2.5.1. □

Lemma 2.20. *Let $r' = 40|\mathbb{F}|r$. Let $\epsilon' = 10|\mathbb{F}|\epsilon$. Let $z \in D_X$. Let $L_1, \ldots, L_{3k} : \mathbb{F} \to D_X$ be $3k$ random lines, such that for every $L \in \{L_1, \ldots, L_{3k}\}$, we have $L(0) = z$. Let $S = \{L_j(t)\}_{j\in[3k],t\in\mathbb{F}} \subset D_X$. Let $A \in_R \mathcal{A}_S$.*

For any $v \in \mathbb{F}$, define $A^v : S \to \mathbb{F}$ by $A^v(z') = A(z')$ for $z' \neq z$ and $A^v(z) = v$.

Then, with probability $\geq 1 - \epsilon'$, there exists $v \in \mathbb{F}$, such that, for at least $3k - r'$ of the indices $j \in [3k]$, $A^v \circ L_j : \mathbb{F} \to \mathbb{F}$ is a univariate polynomial of degree $< m|H|$ (where the probability is over $L_1, \ldots, L_{3k}, A$).

Proof. Let $L_1, \ldots, L_{3k} : \mathbb{F} \to D_X$ be $3k$ random lines, such that for every $L \in \{L_1, \ldots, L_{3k}\}$, we have $L(0) = z$. Let $S = \{L_j(t)\}_{j\in[3k],t\in\mathbb{F}} \subset D_X$. Let $A \in_R \mathcal{A}_S$. For any $v \in \mathbb{F}$, define $A^v : S \to \mathbb{F}$ by $A^v(z') = A(z')$ for $z' \neq z$ and $A^v(z) = v$.

Apply Lemma 2.18 twice: once on the set of lines $\{L_1, \ldots, L_{2k}\}$, and once on the set of lines $\{L_{k+1}, \ldots, L_{3k}\}$. By applying Lemma 2.18 twice, and using also Claim 2.5.1, we know that with probability $\geq 1 - \epsilon'$, both of the following are satisfied:

1. There exists $v_1 \in \mathbb{F}$, such that, for at least $2k - 20|\mathbb{F}|r$ of the indices $j \in \{1, \ldots, 2k\}$, $A^v \circ L_j : \mathbb{F} \to \mathbb{F}$ is a univariate polynomial of degree $< m|H|$.
2. There exists $v_2 \in \mathbb{F}$, such that, for at least $2k - 20|\mathbb{F}|r$ of the indices $j \in \{k + 1, \ldots, 3k\}$, $A^v \circ L_j : \mathbb{F} \to \mathbb{F}$ is a univariate polynomial of degree $< m|H|$.

Note that if both of the above are satisfied then $v_1 = v_2$, since $2k - 20|\mathbb{F}|r > 3k/2$. Therefore, with probability $\geq 1 - \epsilon'$, there exists $v \in \mathbb{F}$, such that, for at least $3k - 40|\mathbb{F}|r$ of the indices $j \in \{1, \ldots, 3k\}$, $A^v \circ L_j : \mathbb{F} \to \mathbb{F}$ is a univariate polynomial of degree $< m|H|$. □

2.7.5 Consistency of X and P_0

Let $i_1, i_2, i_3 \in H^m$. Let $b_1, b_2, b_3 \in \{0, 1\}$ be such that $\phi(i_1, i_2, i_3, b_1, b_2, b_3) = 1$, that is, the clause $(w_{i_1} = b_1) \vee (w_{i_2} = b_2) \vee (w_{i_3} = b_3)$ appears in the 3-CNF formula φ.

In this subsection we will show that, intuitively, when taking a large number of lines through each of the points $i_1, i_2, i_3 \in D_X$, with high probability, there exist values $v_1, v_2, v_3 \in \mathbb{F}$, that satisfy $(v_1 - b_1) \cdot (v_2 - b_2) \cdot (v_3 - b_3) = 0$, and such that:

1. For most of the lines through i_1, the answers correspond to low degree polynomials that "evaluate" the point i_1 to v_1.

2. For most of the lines through i_2, the answers correspond to low degree polynomials that "evaluate" the point i_2 to v_2.

3. For most of the lines through i_3, the answers correspond to low degree polynomials that "evaluate" the point i_3 to v_3.

This is stated formally in Lemma 2.22. To prove this lemma, we will need to first prove Lemma 2.21.

Lemma 2.21. *Let $r' = 9\ell|\mathbb{F}|r$. Let $\epsilon' = 9\ell|\mathbb{F}|\epsilon$. Let $z = (i_1, i_2, i_3, b_1, b_2, b_3) \in (H^m)^3 \times H^3 = H^\ell \subset \mathbb{F}^\ell$. We view z as a point in D_0. We view $i_1, i_2, i_3 \in H^m \subset \mathbb{F}^m$ as points in D_X.*

Let $L_1, \ldots, L_k : \mathbb{F} \to D_0$ be k random lines, such that for every $L_j \in \{L_1, \ldots, L_k\}$, we have $L_j(0) = z$. For every $L_j \in \{L_1, \ldots, L_k\}$, the line L_j is a function $L_j : \mathbb{F} \to \mathbb{F}^\ell$. Let $L_j^1 : \mathbb{F} \to \mathbb{F}^m$ be L_j, restricted to coordinates $\{1, \ldots, m\}$. Let $L_j^2 : \mathbb{F} \to \mathbb{F}^m$ be L_j, restricted to coordinates $\{m+1, \ldots, 2m\}$. Let $L_j^3 : \mathbb{F} \to \mathbb{F}^m$ be L_j, restricted to coordinates $\{2m+1, \ldots, 3m\}$. We think of L_j^1, L_j^2, L_j^3 as lines $L_j^1, L_j^2, L_j^3 : \mathbb{F} \to D_X$, and note that $L_j^1(0) = i_1, L_j^2(0) = i_2, L_j^3(0) = i_3$.

Let $S^0 = \{L_j(t)\}_{j\in[k],t\in\mathbb{F}} \subset D_0$. Let $S^X = \{L_j^1(t), L_j^2(t), L_j^3(t)\}_{j\in[k],t\in\mathbb{F}} \subset D_X$. Let $S = S^0 \cup S^X \subset D$. Let $A \in_R \mathcal{A}_S$.

Define $A_0^0 : S^0 \to \mathbb{F}$ by $A_0^0(z') = A_0(z')$ for $z' \neq z$ and $A_0^0(z) = 0$. For any $i \in D_X$ and $v \in \mathbb{F}$, 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 $A_X^{i\to v}(i) = v$.

Then, with probability $\geq 1-\epsilon'$, there exist $v_1, v_2, v_3 \in \mathbb{F}$, such that, for at least $k - r'$ of the indices $j \in [k]$, the following is satisfied (where the probability is over $L_1, \ldots, L_k, A$):

1. *$A_0^0 \circ L_j : \mathbb{F} \to \mathbb{F}$ is a univariate polynomial of degree $< 2\ell|H|$.*

2. *$A_X^{i_1\to v_1} \circ L_j^1 : \mathbb{F} \to \mathbb{F}$ is a univariate polynomial of degree $< m|H|$.*

3. *$A_X^{i_2\to v_2} \circ L_j^2 : \mathbb{F} \to \mathbb{F}$ is a univariate polynomial of degree $< m|H|$.*

4. *$A_X^{i_3\to v_3} \circ L_j^3 : \mathbb{F} \to \mathbb{F}$ is a univariate polynomial of degree $< m|H|$.*

5. *$\hat{\phi}(z) \cdot (v_1 - b_1) \cdot (v_2 - b_2) \cdot (v_3 - b_3) = 0$*

Proof. Let $L_1, \ldots, L_k : \mathbb{F} \to D_0$ be k random lines, such that for every $L_j \in \{L_1, \ldots, L_k\}$, we have $L_j(0) = z$. For every $j \in [k]$: Let $L_j^1 : \mathbb{F} \to \mathbb{F}^m$ be L_j, restricted to coordinates $\{1, \ldots, m\}$. Let $L_j^2 : \mathbb{F} \to \mathbb{F}^m$ be L_j, restricted to coordinates $\{m+1, \ldots, 2m\}$. Let $L_j^3 : \mathbb{F} \to \mathbb{F}^m$ be L_j, restricted to coordinates $\{2m+1, \ldots, 3m\}$. We view L_j^1, L_j^2, L_j^3 as $L_j^1, L_j^2, L_j^3 : \mathbb{F} \to D_X$.

Let $S^0 = \{L_j(t)\}_{j\in[k],t\in\mathbb{F}} \subset D_0$. Let $S^X = \{L_j^1(t), L_j^2(t), L_j^3(t)\}_{j\in[k],t\in\mathbb{F}} \subset D_X$. Let $S = S^0 \cup S^X \subset D$. Let $A \in_R \mathcal{A}_S$.

Define $A_0^0 : S^0 \to \mathbb{F}$ by $A_0^0(z') = A_0(z')$ for $z' \neq z$ and $A_0^0(z) = 0$. For any $i \in D_X$ and $v \in \mathbb{F}$, 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 $A_X^{i \to v}(i) = v$.

Denote by E the event that for every $j \in [k]$, and every $w \in \{1, 2, 3\}$ the line L_j^w is in a general position (as a line in $\mathbb{F}^m$), that is, it's image is not a single point. Note that the event E occurs with probability of at least $1 - \frac{3k}{|\mathbb{F}|^{m-1}}$.

By Lemma 2.7, using Claim 2.5.1, with probability $\geq 1 - 8\ell|\mathbb{F}|\epsilon - \delta$, for at least $k - 8\ell|\mathbb{F}|r$ of the indices $j \in [k]$, we have that $A_0^0 \circ L_j : \mathbb{F} \to \mathbb{F}$ is a univariate polynomial of degree $< 2\ell|H|$.

By Lemma 2.19, using Claim 2.5.1, with probability $\geq 1 - 5|\mathbb{F}|\epsilon - \delta$, there exists $v_1 \in \mathbb{F}$, such that, for at least $k - 20|\mathbb{F}|r$ of the indices $j \in [k]$, 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 $< m|H|$.

By Lemma 2.19, using Claim 2.5.1, with probability $\geq 1 - 5|\mathbb{F}|\epsilon - \delta$, there exists $v_2 \in \mathbb{F}$, such that, for at least $k - 20|\mathbb{F}|r$ of the indices $j \in [k]$, 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 $< m|H|$.

By Lemma 2.19, using Claim 2.5.1, with probability $\geq 1 - 5|\mathbb{F}|\epsilon - \delta$, there exists $v_3 \in \mathbb{F}$, such that, for at least $k - 20|\mathbb{F}|r$ of the indices $j \in [k]$, 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 $\{L_j(t)\}_{j \in [k]}$ and note that this is a set of k random points in D_0. Each point $L_j(t) \in \mathbb{F}^\ell$ can be written as

$$L_j(t) = (L_j^1(t), L_j^2(t), L_j^3(t), L_j(t)_{\ell-2}, L_j(t)_{\ell-1}, L_j(t)_\ell) \in (\mathbb{F}^m)^3 \times \mathbb{F}^3 = \mathbb{F}^\ell$$

where $L_j(t)_{\ell-2}, L_j(t)_{\ell-1}, L_j(t)_\ell$ are the last 3 coordinates of $L_j(t)$. By Claim 2.5.8, using also Claim 2.5.1, for every $t \in \mathbb{F} \setminus \{0\}$, with probability $> 1 - \epsilon - 2\delta$, for at least $k - r$ of the indices $j \in [k]$, we have

$$A_0(L_j(t)) = \hat{\phi}(L_j(t)) \cdot (A_X(L_j^1(t)) - L_j(t)_{\ell-2}) \cdot (A_X(L_j^2(t)) - L_j(t)_{\ell-1}) \cdot (A_X(L_j^3(t)) - L_j(t)_\ell)$$

If in addition the event E occurs, this implies that for every $v_1, v_2, v_3 \in \mathbb{F}$,

$$A_0^0(L_j(t)) = \hat{\phi}(L_j(t)) \cdot (A_X^{i_1 \to v_1}(L_j^1(t)) - L_j(t)_{\ell-2}) \cdot (A_X^{i_2 \to v_2}(L_j^2(t)) - L_j(t)_{\ell-1}) \cdot (A_X^{i_3 \to v_3}(L_j^3(t)) - L_j(t)_\ell)$$

Adding up all this, by the union bound, we obtain that with probability $\geq 1 - \epsilon'$, there exist $v_1, v_2, v_3 \in \mathbb{F}$, such that, for at least $k - r'$ of the indices $j \in [k]$, the following is satisfied (where the probability is over $L_1, \ldots, L_k, A$):

1. $A_0^0 \circ L_j : \mathbb{F} \to \mathbb{F}$ is a univariate polynomial of degree $< 2\ell|H|$.
2. $A_X^{i_1 \to v_1} \circ L_j^1 : \mathbb{F} \to \mathbb{F}$ is a univariate polynomial of degree $< m|H|$.
3. $A_X^{i_2 \to v_2} \circ L_j^2 : \mathbb{F} \to \mathbb{F}$ is a univariate polynomial of degree $< m|H|$.
4. $A_X^{i_3 \to v_3} \circ L_j^3 : \mathbb{F} \to \mathbb{F}$ is a univariate polynomial of degree $< m|H|$.

5. For every $t \in \mathbb{F} \setminus \{0\}$,

$$A_0^0(L_j(t)) =$$

$$\hat{\phi}(L_j(t)) \cdot (A_X^{i_1 \to v_1}(L_j^1(t)) - L_j(t)_{\ell-2}) \cdot (A_X^{i_2 \to v_2}(L_j^2(t)) - L_j(t)_{\ell-1}) \cdot (A_X^{i_3 \to v_3}(L_j^3(t)) - L_j(t)_\ell)$$

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 $L_j(0) = z$, we have

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

□

Lemma 2.22. *Let $r' = 9\ell|\mathbb{F}|r$. Let $\epsilon' = 9\ell|\mathbb{F}|\epsilon + \delta$. Let $i_1, i_2, i_3 \in H^m$. We view i_1, i_2, i_3 as points in D_X. Let $b_1, b_2, b_3 \in \{0, 1\}$ be such that $\phi(i_1, i_2, i_3, b_1, b_2, b_3) = 1$, that is, the clause $(w_{i_1} = b_1) \vee (w_{i_2} = b_2) \vee (w_{i_3} = b_3)$ appears in the 3-CNF formula φ.*

Let $L_1^1, \ldots, L_k^1 : \mathbb{F} \to D_X$ be k random lines, such that for every $L \in \{L_1^1, \ldots, L_k^1\}$, we have $L(0) = i_1$. Let $L_1^2, \ldots, L_k^2 : \mathbb{F} \to D_X$ be k random lines, such that for every $L \in \{L_1^2, \ldots, L_k^2\}$, we have $L(0) = i_2$. Let $L_1^3, \ldots, L_k^3 : \mathbb{F} \to D_X$ be k random lines, such that for every $L \in \{L_1^3, \ldots, L_k^3\}$, we have $L(0) = i_3$.

Let $S = \{L_j^1(t), L_j^2(t), L_j^3(t)\}_{j \in [k], t \in \mathbb{F}} \subset D_X$. Let $A \in_R \mathcal{A}_S$.

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 $A^{i \to v}(i) = v$.

Then, with probability $\geq 1 - \epsilon'$, there exist $v_1, v_2, v_3 \in \mathbb{F}$, such that, for at least $k - r'$ of the indices $j \in [k]$, the following is satisfied
(where the probability is over $L_1^1, \ldots, L_k^1, L_1^2, \ldots, L_k^2, L_1^3, \ldots, L_k^3, A$):

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

Proof. Follows immediately by Lemma 2.21, applied for the point $z = (i_1, i_2, i_3, b_1, b_2, b_3)$, and Claim 2.5.1. □

2.7.6 Property $\mathbb{R}(\epsilon', r')$

Recall that we have a (fanin 2) Boolean circuit $\mathcal{C}_n$ of size $N = O(t(n)s(n))$ that computes $\mathcal{L}$ on inputs of length n. The circuit $\mathcal{C}_n$ is layered, with $O(s(n))$ gates in each layer, such that a child of a gate in layer $i + 1$ is either an input variable (or a negation of an input variable) or a gate in layer i. Recall that we assume that in the circuit $\mathcal{C}_n$ all negations are on input variables, and that the two children of any gate in the circuit are different.

Recall that the gates of the circuit are indexed by the numbers $1, \ldots, N$, in an order that agrees with the layers of the circuit. We assume that $1, \ldots, n$ are the indexes of the n input variables and $n+1, \ldots, 2n$ are the indexes of their negations, and that N is the index of the special output gate.

Recall that $\varphi(w_1, \ldots, w_N)$ is a 3-CNF Boolean formula, such that, $\varphi(w_1, \ldots, w_N) = 1$ if and only if $w_1, \ldots, w_N$ is the computation of the circuit $\mathcal{C}_n$ on the input $x = (x_1, \ldots, x_n)$, and $w_N = 1$. Denote by $x_1, \ldots, x_N$ the computation of the circuit $\mathcal{C}_n$ on the input $x = (x_1, \ldots, x_n)$. Thus, $\varphi(w_1, \ldots, w_N) = 1$ if and only if $(w_1, \ldots, w_N) = (x_1, \ldots, x_N)$, and $x_N = 1$.

Recall that since $N = |H|^m$, we identify $[N]$ and H^m by the lexicographic order on H^m, and view $w_1, \ldots, w_N$ and $x_1, \ldots, x_N$ as indexed by $i \in H^m$ (rather than $i \in [N]$). We hence view $x = (x_1, \ldots, x_N)$ as a function $x : H^m \to \{0,1\}$ (given by $x(i) = x_i$, where we identify $[N]$ and H^m).

Recall that $\phi : (H^m)^3 \times \{0,1\}^3 \to \{0,1\}$ is a function where $\phi(i_1, i_2, i_3, b_1, b_2, b_3) = 1$ if and only if the clause $(w_{i_1} = b_1) \vee (w_{i_2} = b_2) \vee (w_{i_3} = b_3)$ appears in φ, and $\hat{\phi} : \mathbb{F}^\ell \to \mathbb{F}$ is the low-degree extension of ϕ.

We will now give a definition that will be central in the rest of the section. Intuitively, a subset $B \subset H^m \subset D_X$ satisfies property $\mathbb{R}(\epsilon', r')$ if when taking k lines through every point in B, with high probability, for every point $i \in B$, for most of the lines through the point i, the answers correspond to low degree polynomials that "evaluate" the point i to x_i.

To make sure that the property is well defined, we will limit ourselves to sets $B \subset H^m$ such that $k|B||\mathbb{F}| \leq k_{max}$. Since we identify H^m and $[N]$, we view each set B also as a subset of $[N]$. We will think of every set B also as a subset of D_X.

Let $\mathcal{B}$ be the set of all subsets $B \subset [N]$, such that, $k|B||\mathbb{F}| \leq k_{max}/2$.

Definition 2.23. Property $\mathbb{R}(\epsilon', r')$:
Let $\epsilon' \geq 0$ and $r' \geq 0$. Let $B \subset H^m$ be such that $k|B||\mathbb{F}| \leq k_{max}$. We view B as a subset of D_X.

For every $i \in B$, let $L^i_1, \ldots, L^i_k : \mathbb{F} \to D_X$ be k random lines, such that for every $L \in \{L^i_1, \ldots, L^i_k\}$, we have $L(0) = i$.

Let $S = \{L^i_j(t)\}_{i \in B, j \in [k], t \in \mathbb{F}} \subset D_X$. Let $A \in_R \mathcal{A}_S$.

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 $A^{i \to v}(i) = v$.

We say that the set B satisfies property $\mathbb{R}(\epsilon', r')$ (also denoted $B \in \mathbb{R}(\epsilon', r')$) if with probability $\geq 1 - \epsilon'$, for every $i \in B$, for at least $k - r'$ of the lines $L \in \{L^i_1, \ldots, L^i_k\}$, we have that $A^{i \to x_i} \circ L : \mathbb{F} \to \mathbb{F}$ is a univariate polynomial of degree $< m|H|$ (where the probability is over $\{L^i_j\}_{i \in B, j \in [k]}, A$).

We think of the empty set as satisfying $\mathbb{R}(\epsilon', r')$ for any ϵ', r'.

In all that comes below, we fix

$$r' = 9\ell|\mathbb{F}|r$$

Lemma 2.24. *Let $i \in [2n]$. Then, $\{i\} \in \mathbb{R}(\epsilon', r')$, where $\epsilon' = 10\ell|\mathbb{F}|\epsilon$, and $r' = 9\ell|\mathbb{F}|r$.*

Proof. We will give the proof for $i \in [n]$, such that, $x_i = 0$. The proof for $i \in [n]$, such that, $x_i = 1$, and for $i \in \{n+1, \ldots, 2n\}$ is similar.

Recall that for every $i \in [n]$, the formula φ contains a clause that checks that $w_i = x_i$. For example, if $x_i = 0$, we have the clause $(w_i = 0) \vee (w_i = 0) \vee (w_i = 0)$ that ensures that $w_i = 0$.

Let $L_1^1, \ldots, L_k^1, L_1^2, \ldots, L_k^2, L_1^3, \ldots, L_k^3 : \mathbb{F} \to D_X$ be $3k$ random lines, such that for every line $L \in \{L_1^1, \ldots, L_k^1, L_1^2, \ldots, L_k^2, L_1^3, \ldots, L_k^3\}$, we have $L(0) = i$.

Let $S = \{L_j^1(t), L_j^2(t), L_j^3(t)\}_{j\in[k], t\in\mathbb{F}} \subset D_X$. Let $A \in_R \mathcal{A}_S$.

For any $v \in \mathbb{F}$, define $A^{i\to v} : S \to \mathbb{F}$ by $A^{i\to v}(i') = A(i')$ for $i' \neq i$ and $A^{i\to v}(i) = v$.

By Lemma 2.22, with probability $\geq 1 - 9\ell|\mathbb{F}|\epsilon - \delta$, there exist $v_1, v_2, v_3 \in \mathbb{F}$, such that, for at least $k - r'$ of the indices $j \in [k]$, the following is satisfied
(where the probability is over $L_1^1, \ldots, L_k^1, L_1^2, \ldots, L_k^2, L_1^3, \ldots, L_k^3, A$):

1. $A^{i\to v_1} \circ L_j^1 : \mathbb{F} \to \mathbb{F}$ is a univariate polynomial of degree $< m|H|$.
2. $A^{i\to v_2} \circ L_j^2 : \mathbb{F} \to \mathbb{F}$ is a univariate polynomial of degree $< m|H|$.
3. $A^{i\to v_3} \circ L_j^3 : \mathbb{F} \to \mathbb{F}$ is a univariate polynomial of degree $< m|H|$.
4. $v_1 \cdot v_2 \cdot v_3 = 0$.

On the other hand, by Lemma 2.20, with probability $\geq 1 - 10|\mathbb{F}|\epsilon$, there exists $v \in \mathbb{F}$, such that, for at least $3k - 40|\mathbb{F}|r$ of the lines $L \in \{L_1^1, \ldots, L_k^1, L_1^2, \ldots, L_k^2, L_1^3, \ldots, L_k^3\}$, $A^{i\to v} \circ L : \mathbb{F} \to \mathbb{F}$ is a univariate polynomial of degree $< m|H|$.

Thus, by the union bound, with probability $\geq 1 - 9\ell|\mathbb{F}|\epsilon - 10|\mathbb{F}|\epsilon - \delta$, there exist $v_1, v_2, v_3, v \in \mathbb{F}$, such that, $v_1 \cdot v_2 \cdot v_3 = 0$, and

1. For at least $k - r'$ of the indices $j \in [k]$, $A^{i\to v_1} \circ L_j^1 : \mathbb{F} \to \mathbb{F}$ is a univariate polynomial of degree $< m|H|$.
2. For at least $k - r'$ of the indices $j \in [k]$, $A^{i\to v_2} \circ L_j^2 : \mathbb{F} \to \mathbb{F}$ is a univariate polynomial of degree $< m|H|$.
3. For at least $k - r'$ of the indices $j \in [k]$, $A^{i\to v_3} \circ L_j^3 : \mathbb{F} \to \mathbb{F}$ is a univariate polynomial of degree $< m|H|$.
4. For at least $3k - 40|\mathbb{F}|r$ of the lines $L \in \{L_1^1, \ldots, L_k^1, L_1^2, \ldots, L_k^2, L_1^3, \ldots, L_k^3\}$, $A^{i\to v} \circ L : \mathbb{F} \to \mathbb{F}$ is a univariate polynomial of degree $< m|H|$.

Since, $40|\mathbb{F}|r + r' < k$, this implies $v = v_1 = v_2 = v_3$, and hence $v = 0$.

Thus, with probability $\geq 1 - 9\ell|\mathbb{F}|\epsilon - 10|\mathbb{F}|\epsilon - \delta > 1 - \epsilon' + \delta$, for at least $k - r'$ of the lines $L \in \{L_1^1, \ldots, L_k^1\}$, $A^{i\to 0} \circ L : \mathbb{F} \to \mathbb{F}$ is a univariate polynomial of degree $< m|H|$.

The proof of the lemma hence follows by Claim 2.5.1. □

Lemma 2.25. *Let $i_1, i_2, i_3 \in [N]$ be such that the gate indexed by i_1 in the circuit $\mathcal{C}_n$ has children indexed by i_2, i_3. Let $B \in \mathcal{B}$ be such that $i_2, i_3 \in B$. Assume that $B \in \mathbb{R}(\epsilon', r')$, where $r' = 9\ell|\mathbb{F}|r$. Then $B \cup \{i_1\} \in \mathbb{R}(\epsilon'', r')$, where $\epsilon'' = \epsilon' + 9\ell|\mathbb{F}|\epsilon + 3\delta$.*

Proof. Let $B' = B \cup \{i_1\}$. For every $i \in B'$, let $L_1^i, \ldots, L_k^i : \mathbb{F} \to D_X$ be k random lines, such that for every $L \in \{L_1^i, \ldots, L_k^i\}$, we have $L(0) = i$.

Let $S = \{L_j^i(t)\}_{i \in B', j \in [k], t \in \mathbb{F}} \subset D_X$. Let $A \in_R \mathcal{A}_S$.

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 $A^{i \to v}(i) = v$.

Since the gate indexed by i_1 in the circuit $\mathcal{C}_n$ has children indexed by i_2, i_3, the formula φ contains the clause $(w_{i_2} = x_{i_2}) \wedge (w_{i_3} = x_{i_3}) \to (w_{i_1} = x_{i_1})$.

By Lemma 2.22 and Claim 2.5.1, with probability $\geq 1 - 9\ell|\mathbb{F}|\epsilon - 2\delta$, there exist $v_1, v_2, v_3 \in \mathbb{F}$, such that, for at least $k - r'$ of the indices $j \in [k]$, the following is satisfied (where the probability is over $\{L_j^i(t)\}_{i \in B', j \in [k], t \in \mathbb{F}}, A$):

1. $A^{i_1 \to v_1} \circ L_j^{i_1} : \mathbb{F} \to \mathbb{F}$ is a univariate polynomial of degree $< m|H|$.
2. $A^{i_2 \to v_2} \circ L_j^{i_2} : \mathbb{F} \to \mathbb{F}$ is a univariate polynomial of degree $< m|H|$.
3. $A^{i_3 \to v_3} \circ L_j^{i_3} : \mathbb{F} \to \mathbb{F}$ is a univariate polynomial of degree $< m|H|$.
4. $(v_2 = x_{i_2}) \wedge (v_3 = x_{i_3}) \to (v_1 = x_{i_1})$

On the other hand, since the set B satisfies property $\mathbb{R}(\epsilon', r')$, using Claim 2.5.1, with probability $\geq 1 - \epsilon' - \delta$, for every $i \in B$: For at least $k - r'$ of the lines $L \in \{L_1^i, \ldots, L_k^i\}$, we have that $A^{i \to x_i} \circ L : \mathbb{F} \to \mathbb{F}$ is a univariate polynomial of degree $< m|H|$.

Thus, by the union bound, with probability $\geq 1 - \epsilon' - 9\ell|\mathbb{F}|\epsilon - 3\delta$, there exist $v_1, v_2, v_3 \in \mathbb{F}$, such that, $(v_2 = x_{i_2}) \wedge (v_3 = x_{i_3}) \to (v_1 = x_{i_1})$, and

1. For at least $k - r'$ of the indices $j \in [k]$, $A^{i_1 \to v_1} \circ L_j^{i_1} : \mathbb{F} \to \mathbb{F}$ is a univariate polynomial of degree $< m|H|$.
2. For at least $k - r'$ of the indices $j \in [k]$, $A^{i_2 \to v_2} \circ L_j^{i_2} : \mathbb{F} \to \mathbb{F}$ is a univariate polynomial of degree $< m|H|$.
3. For at least $k - r'$ of the indices $j \in [k]$, $A^{i_3 \to v_3} \circ L_j^{i_3} : \mathbb{F} \to \mathbb{F}$ is a univariate polynomial of degree $< m|H|$.
4. For every $i \in B$, for at least $k - r'$ of the lines $L \in \{L_1^i, \ldots, L_k^i\}$, we have that $A^{i \to x_i} \circ L : \mathbb{F} \to \mathbb{F}$ is a univariate polynomial of degree $< m|H|$.

Since, $r' + r' < k$, this implies $v_2 = x_{i_2}$, $v_3 = x_{i_3}$ and hence also $v_1 = x_{i_1}$.

Thus, with probability $\geq 1 - \epsilon' - 9\ell|\mathbb{F}|\epsilon - 3\delta$,

1. For at least $k - r'$ of the indices $j \in [k]$, $A^{i_1 \to x_{i_1}} \circ L_j^{i_1} : \mathbb{F} \to \mathbb{F}$ is a univariate polynomial of degree $< m|H|$.
2. For every $i \in B$, for at least $k - r'$ of the lines $L \in \{L_1^i, \ldots, L_k^i\}$, we have that $A^{i \to x_i} \circ L : \mathbb{F} \to \mathbb{F}$ is a univariate polynomial of degree $< m|H|$.

Thus $B' \in \mathbb{R}(\epsilon'', r')$ □

Lemma 2.26. *Let $B_1, B_2 \in \mathcal{B}$. If $B_1 \in \mathbb{R}(\epsilon_1, r')$ and $B_2 \in \mathbb{R}(\epsilon_2, r')$ then $B_1 \cup B_2 \in \mathbb{R}(\epsilon', r')$, where $\epsilon' = \epsilon_1 + \epsilon_2 + 2\delta$.*

Proof. Follows immediately by the union bound and (two applications of) Claim 2.5.1. □

Lemma 2.27. *Let $B_1, B_2 \in \mathcal{B}$. If $B_1 \subset B_2$ and $B_2 \in \mathbb{R}(\epsilon_2, r')$ then $B_1 \in \mathbb{R}(\epsilon_1, r')$, where $\epsilon_1 = \epsilon_2 + \delta$.*

Proof. Follows immediately by Claim 2.5.1. □

2.7.7 Proof of Lemma 2.5

Lemma 2.5 will be superseded by Lemma 2.31. We include its proof since: (1) it is simpler than the proof of Lemma 2.31, (2) it allows for a more modular proof and (3) what remains to be shown is relatively short.

For the rest of Section 2.7, we assume that $k_{max} \geq 4sk|\mathbb{F}| + 6k\ell|\mathbb{F}|^2$. We assume for a contradiction that for some $x \notin \mathcal{L}$, there exists a δ-no-signaling family of distributions $\{\mathcal{A}_S\}_{S \subset D, |S| \leq k_{max}}$ that fools V' with probability larger than $1 - \epsilon$. That is, the verifier V' accepts with probability $> 1 - \epsilon$, where on queries Q, the answers are given (probabilistically) by $A \in_R \mathcal{A}_Q$.

For every $i \in \{2n, \ldots, N\}$, define $B_i \in \mathcal{B}$ as follows: B_i contains all the indexes $2n < i' \leq i$, such that, in the circuit $\mathcal{C}_n$, the gate indexed by i' is either in the same layer as the gate indexed by i, or in the previous layer. Note that $B_{2n} = \emptyset$ (this was added for the simplicity of the notation) and recall that we think of the empty set as satisfying $\mathbb{R}(\epsilon', r')$ for any ϵ'.

Lemma 2.28. *Assume that $k_{max} \geq 4sk|\mathbb{F}| + 6k\ell|\mathbb{F}|^2$. Let $i \in \{2n+1, \ldots, N\}$. If $B_{i-1} \in \mathbb{R}(\epsilon', r')$, where $r' = 9\ell|\mathbb{F}|r$, then $B_i \in \mathbb{R}(\epsilon'', r')$, where $\epsilon'' = \epsilon' + 30\ell|\mathbb{F}|\epsilon$.*

Proof. Denote by i_1, i_2 the indexes of the two children of the gate indexed by i in the circuit $\mathcal{C}_n$. Note that $\{i_1, i_2\} \subset [2n] \cup B_{i-1}$. Denote $B' = \{i_1, i_2\} \cap [2n]$. Note also that $B_i \subseteq B_{i-1} \cup \{i\}$.

By Lemma 2.24 and Lemma 2.26, $B' \in \mathbb{R}(20\ell|\mathbb{F}|\epsilon + 2\delta, r')$.

Hence, by Lemma 2.26, $B' \cup B_{i-1} \in \mathbb{R}(\epsilon' + 20\ell|\mathbb{F}|\epsilon + 4\delta, r')$.

Hence, by Lemma 2.25, $B' \cup B_{i-1} \cup \{i\} \in \mathbb{R}(\epsilon' + 29\ell|\mathbb{F}|\epsilon + 7\delta, r')$.

Hence, by Lemma 2.27, $B_i \in \mathbb{R}(\epsilon' + 29\ell|\mathbb{F}|\epsilon + 8\delta, r')$. □

Lemma 2.29. *Assume that $k_{max} \geq 4sk|\mathbb{F}| + 6k\ell|\mathbb{F}|^2$. Then, $B_N \in \mathbb{R}(\epsilon', r')$, where $r' = 9\ell|\mathbb{F}|r$ and $\epsilon' = 30N\ell|\mathbb{F}|\epsilon = 0.3$.*

Proof. Follows immediately by an inductive application of Lemma 2.28, and since $\epsilon = \frac{1}{100N\ell|\mathbb{F}|}$. □

Proof of Lemma 2.5

Proof. Let $r' = 9\ell|\mathbb{F}|r$.

Consider the point $N \in [N]$, viewed as a point in $H^m \subset D_X$. Recall that the formula φ contains a clause $(w_N = 1) \vee (w_N = 1) \vee (w_N = 1)$ that checks that $w_N = 1$.

Let $L_1^1, \ldots, L_k^1, L_1^2, \ldots, L_k^2, L_1^3, \ldots, L_k^3 : \mathbb{F} \to D_X$ be $3k$ random lines, such that for every line $L \in \{L_1^1, \ldots, L_k^1, L_1^2, \ldots, L_k^2, L_1^3, \ldots, L_k^3\}$, we have $L(0) = N$.

Let $S = \{L_j^1(t), L_j^2(t), L_j^3(t)\}_{j\in[k],t\in\mathbb{F}} \subset D_X$. Let $A \in_R \mathcal{A}_S$.

For any $v \in \mathbb{F}$, define $A^{N\to v} : S \to \mathbb{F}$ by $A^{N\to v}(i') = A(i')$ for $i' \neq N$ and $A^{N\to v}(N) = v$.

By Lemma 2.22, with probability $\geq 1 - 9\ell|\mathbb{F}|\epsilon - \delta$, there exist $v_1, v_2, v_3 \in \mathbb{F}$, such that, for at least $k - r'$ of the indices $j \in [k]$, the following is satisfied
(where the probability is over $L_1^1, \ldots, L_k^1, L_1^2, \ldots, L_k^2, L_1^3, \ldots, L_k^3, A$):

1. $A^{N\to v_1} \circ L_j^1 : \mathbb{F} \to \mathbb{F}$ is a univariate polynomial of degree $< m|H|$.
2. $A^{N\to v_2} \circ L_j^2 : \mathbb{F} \to \mathbb{F}$ is a univariate polynomial of degree $< m|H|$.
3. $A^{N\to v_3} \circ L_j^3 : \mathbb{F} \to \mathbb{F}$ is a univariate polynomial of degree $< m|H|$.
4. $(v_1 - 1) \cdot (v_2 - 1) \cdot (v_3 - 1) = 0$.

On the other hand, by (three applications of) Lemma 2.29 and Claim 2.5.1:

1. With probability $\geq 1 - 0.3 - 2\delta$, for at least $k - r'$ of the lines $L \in \{L_1^1, \ldots, L_k^1\}$, we have that $A^{N\to x_N} \circ L : \mathbb{F} \to \mathbb{F}$ is a univariate polynomial of degree $< m|H|$.
2. With probability $\geq 1 - 0.3 - 2\delta$, for at least $k - r'$ of the lines $L \in \{L_1^2, \ldots, L_k^2\}$, we have that $A^{N\to x_N} \circ L : \mathbb{F} \to \mathbb{F}$ is a univariate polynomial of degree $< m|H|$.
3. With probability $\geq 1 - 0.3 - 2\delta$, for at least $k - r'$ of the lines $L \in \{L_1^3, \ldots, L_k^3\}$, we have that $A^{N\to x_N} \circ L : \mathbb{F} \to \mathbb{F}$ is a univariate polynomial of degree $< m|H|$.

Thus, by the union bound, with probability $> 0.1 - 9\ell|\mathbb{F}|\epsilon - 7\delta > 0$, there exist $v_1, v_2, v_3 \in \mathbb{F}$, such that, $(v_1 - 1) \cdot (v_2 - 1) \cdot (v_3 - 1) = 0$, and

1. For at least $k - r'$ of the indices $j \in [k]$, $A^{N\to v_1} \circ L_j^1 : \mathbb{F} \to \mathbb{F}$ is a univariate polynomial of degree $< m|H|$.
2. For at least $k - r'$ of the indices $j \in [k]$, $A^{N\to v_2} \circ L_j^2 : \mathbb{F} \to \mathbb{F}$ is a univariate polynomial of degree $< m|H|$.
3. For at least $k - r'$ of the indices $j \in [k]$, $A^{N\to v_3} \circ L_j^3 : \mathbb{F} \to \mathbb{F}$ is a univariate polynomial of degree $< m|H|$.
4. For at least $k - r'$ of the lines $L \in \{L_1^1, \ldots, L_k^1\}$, we have that $A^{N\to x_N} \circ L : \mathbb{F} \to \mathbb{F}$ is a univariate polynomial of degree $< m|H|$.

5. For at least $k - r'$ of the lines $L \in \{L_1^2, \ldots, L_k^2\}$, we have that $A^{N \to x_N} \circ L : \mathbb{F} \to \mathbb{F}$ is a univariate polynomial of degree $< m|H|$.

6. For at least $k - r'$ of the lines $L \in \{L_1^3, \ldots, L_k^3\}$, we have that $A^{N \to x_N} \circ L : \mathbb{F} \to \mathbb{F}$ is a univariate polynomial of degree $< m|H|$.

Since, $r' + r' < k$, this implies $x_N = v_1 = v_2 = v_3$, and hence $x_N = 1$. Thus, the original input x is in the language $\mathcal{L}$. □

2.8 Soundness of V in the Base PCP

Lemma 2.30 will be superseded by Lemma 2.46. We include its proof for completeness.

Recall that $k \leq \mathsf{poly}(n)$, such that $4|\mathbb{F}|^4 \leq k \leq N$, is the security parameter of the PCP, and that $1 \leq r < k$ is the parameter of the relaxed verifier V'. Recall that ℓ and $|\mathbb{F}|$ are bounded by $\mathsf{polylog}(N)$.

Lemma 2.30. *For a security parameter $k \leq \mathsf{poly}(n)$, such that $4|\mathbb{F}|^4 \leq k \leq N$, fix the following parameters: Let $r = \frac{k}{40\ell|\mathbb{F}|}$. Let $\epsilon = 2^{-r/2}$. Let $k_{max} = 4sk|\mathbb{F}| + 12k\ell|\mathbb{F}|^2$, where $s = O(s(n))$ is the maximal number of gates in a layer of the circuit $\mathcal{C}_n$. Let $\delta = \frac{1}{|\mathbb{F}|^{8k\ell|\mathbb{F}|^2}}$. Then, V has soundness ϵ against (k_{max}, δ)-no-signaling strategies.*

Proof. Assume for a contradiction that V doesn't have soundness ϵ against (k_{max}, δ)-no-signaling strategies. By Lemma 2.3, since $\delta < \frac{\epsilon}{8 \cdot |\mathbb{F}|^{6k\ell|\mathbb{F}|^2}}$, we know that V' (with parameter r) doesn't have soundness $1-\epsilon'$ against (k'_{max}, δ')-no-signaling strategies, where $k'_{max} = k_{max} - 6k\ell|\mathbb{F}|^2 = 4sk|\mathbb{F}| + 6k\ell|\mathbb{F}|^2$, and $\delta' = 8\delta|\mathbb{F}|^{6k\ell|\mathbb{F}|^2}/\epsilon < \frac{1}{|\mathbb{F}|^{k\ell|\mathbb{F}|^2}}$, and $\epsilon' = (10\ell|\mathbb{F}|2^{-r} + 2\delta)/\epsilon < \frac{1}{100N\ell|\mathbb{F}|}$.

Hence V' (with parameter r) doesn't have soundness $1 - \epsilon'$ against (k'_{max}, δ')-no-signaling strategies, where $k'_{max} = 4sk|\mathbb{F}| + 6k\ell|\mathbb{F}|^2$, and $\delta' = \frac{1}{|\mathbb{F}|^{k\ell|\mathbb{F}|^2}}$, and $\epsilon' = \frac{1}{100N\ell|\mathbb{F}|}$.

This contradicts Lemma 2.5. □

2.9 The Augmented PCP

In this section we describe the construction of the augmented PCP system, based on the base PCP system described in Section 2.5.

Let $\mathcal{L}$ be a language in $\mathsf{DTISP}(t(n), s(n))$, where $\mathsf{poly}(n) \leq t(n) \leq \mathsf{exp}(n)$ and $\max(n, \log(t(n))) \leq s(n) \leq t(n)$. Let x be an input of length n. Since $\mathcal{L} \in \mathsf{DTISP}(t(n), s(n))$, for any n there is a (fanin 2) Boolean circuit $\mathcal{C}_n$ of size $N = O(t(n)s(n))$ that computes $\mathcal{L}$ on inputs of length n. Moreover, the circuit $\mathcal{C}_n$ is layered, with at most $t = O(t(n))$ layers that consist of $s = O(s(n))$ gates each. For simplicity, we think of the input variables $x_1, \ldots, x_n$ and their negations as being included in each layer of $\mathbb{C}_n$ (since $s \geq n$ this property can be achieved by increasing s by a constant factor)

We augment each circuit $\mathbb{C}_n$ to produce a circuit $\mathbb{C}'_n$ as follows.

Let $\mathbb{G}$ be a finite field of characteristic 2 and size $|\mathbb{G}| = \Theta(\log^2 s)$. Fix an arbitrary set $H_{\mathbb{G}} \subset \mathbb{G}$ of size $|H_{\mathbb{G}}| = \log s$ and a dimension $m_{\mathbb{G}} = \frac{\log s}{\log\log s}$ (such that $|H_{\mathbb{G}}|^{m_{\mathbb{G}}} = s$ and $m_{\mathbb{G}} \cdot |H_{\mathbb{G}}| < \frac{|\mathbb{G}|-1}{2}$). (For simplicity and without loss of generality we assume that $\log s$ and $\frac{\log s}{\log\log s}$ are integers). We construct a circuit $\mathbb{C}_{\mathsf{LDE}} : \{0,1\}^s \to \{0,1\}^{\mathsf{poly}(s)}$ by the following two-step process:

1. Given an input $\alpha \in \{0,1\}^s$, the circuit $\mathbb{C}_{\mathsf{LDE}}$ first computes the LDE $\hat{\alpha}$ of α w.r.t. $\mathbb{G}, H_{\mathbb{G}}, m_{\mathbb{G}}$. Recall that the polynomial $\hat{\alpha} : \mathbb{G}^{m_{\mathbb{G}}} \to \mathbb{G}$ is the (unique) individual degree $|H_{\mathbb{G}}|-1$ polynomial that agrees with α on $H_{\mathbb{G}}^{m_{\mathbb{G}}}$ (when α is interpreted as the truth table of a function $\alpha : H_{\mathbb{G}}^{m_{\mathbb{G}}} \to \{0,1\}$), see Section 2.4.6. Denote the output of this step by α'. We note that α' can be computed by a Boolean circuit of size $\mathsf{poly}(|\mathbb{G}|^{m_{\mathbb{G}}}) = \mathsf{poly}(s)$ and depth $O\big(m_{\mathbb{G}} \cdot \log(|\mathbb{G}|) + \log m_{\mathbb{G}} \cdot \mathsf{polylog}(|\mathbb{G}|)\big) = O(\log(s))$ (see Appendix 2.A).

2. As its second (seemingly redundant) step, the circuit $\mathbb{C}_{\mathsf{LDE}}$ verifies that the restriction of α' to *every* line $L : \mathbb{G} \to \mathbb{G}^{m_{\mathbb{G}}}$ is a degree $m_{\mathbb{G}}|H_{\mathbb{G}}|$ univariate polynomial. That is, for every line L, the circuit $\mathbb{C}_{\mathsf{LDE}}$ checks that the function $\alpha' \circ L$ is a degree $m_{\mathbb{G}}|H_{\mathbb{G}}|$ univariate polynomial. For every such line L there is a corresponding output bit of $\mathbb{C}_{\mathsf{LDE}}$ that equals 1 if $\alpha' \circ L$ has low degree and 0 otherwise. (Indeed, if α' is in fact the LDE of α then every output bit of $\mathbb{C}_{\mathsf{LDE}}$ should have value 1.)

   We note that testing the degree of a univariate function $f : \mathbb{G} \to \mathbb{G}$ can be done by a Boolean circuit of size $\mathsf{poly}(|\mathbb{G}|) = \mathsf{polylog}(s)$ and depth $\mathsf{polylog}(|\mathbb{G}|) = \mathsf{polylog}(\log(s))$.

We denote by d the depth of $\mathbb{C}_{\mathsf{LDE}}$ and note that $d = O(\log s)$. We also note that the circuit $\mathbb{C}_{\mathsf{LDE}}$ has size $\mathsf{poly}(s)$ but if that size is smaller than $2^d \cdot s \cdot \log^5(t)$ then we (artificially) increase the size of $\mathbb{C}_{\mathsf{LDE}}$ to be $2^d \cdot s \cdot \log^5(t)$ while maintaining the depth d (by simply adding dummy gates).[13] We assume that $\mathbb{C}_{\mathsf{LDE}}$ contains no negation gates, but may contain *arbitrary* fan-in 2 Boolean gates. We also note that $\mathbb{C}_{\mathsf{LDE}}$ can be generated by a Turing machine in space $O(\log s)$.

The circuit $\mathbb{C}'_n$ is constructed by adding to $\mathbb{C}_n$ the computation of $\mathbb{C}_{\mathsf{LDE}}$ on *every* layer of $\mathbb{C}_n$. Thus, the circuit $\mathbb{C}'_n$ is composed of t layers, where each layer consists of the corresponding layer of $\mathbb{C}_n$ and the computation of $\mathbb{C}_{\mathsf{LDE}}$ on that layer. That is, the first layer consists of the first layer of $\mathbb{C}_n$ and the computation of $\mathbb{C}_{\mathsf{LDE}}$ on the first layer of $\mathbb{C}_n$ and for each $\mu \in \{2, \ldots, t\}$, the μ-th layer of $\mathbb{C}'_n$ consists of the computation of the μ-th layer of $\mathbb{C}_n$ from the $(\mu-1)$-th layer of $\mathbb{C}'_n$ and the computation of $\mathbb{C}_{\mathsf{LDE}}$ of the μ-th layer of $\mathbb{C}_n$. We denote the size of $\mathbb{C}'_n$ by N'. Recall that the input variables $x_1, \ldots, x_n$ and their negations are included in each layer of $\mathbb{C}_n$. Thus, the layer μ of $\mathbb{C}'_n$ can be computed directly from layer $\mu - 1$ of $\mathbb{C}'_n$. Note that $\mathbb{C}'_n$ has depth $t \cdot d$ and that

$$s \cdot t \cdot 2^d \cdot \log^5(t) \leq N' \leq \mathsf{poly}(N).$$

[13] This step can actually be avoided and we do it solely for convenience.

We call the gate indexed by N' the special output gate and note that its value represents the decision of whether $x \in \mathcal{L}$. We also assume without loss of generality that in the circuit $\mathbb{C}'_n$ all negations are on input variables, and that the two children of any gate in the circuit are different (this property can be achieved by duplicating each gate in the circuit twice, increasing the number of gates in each layer by a factor of 2). Note however that $\mathbb{C}'_n$ contains arbitrary fan-in 2 Boolean gates. Lastly, we note that there exists an $O(\log N')$ space Turing machine that on input $n \in \mathbb{N}$ outputs the circuit $\mathbb{C}'_n$.

For every layer $\mu \in [t]$ we denote by $\beta_\mu \subset [N']$ the set of indices of gates in $\mathbb{C}'_n$ that are associated with the LDE of the μ-th layer of $\mathbb{C}_n$. For $z \in \mathbb{G}^{m_\mathbb{G}}$, we denote by $\beta_\mu[z] \subset \beta_\mu$ the set of indices of the $\log_2 |\mathbb{G}|$ gates associated with the point z in the computation of the LDE of layer μ in $\mathbb{C}'_n$. For a sequence of indices $Z \subset \mathbb{G}^{m_\mathbb{G}}$ we denote by $\beta_\mu[Z] \stackrel{\text{def}}{=} \cup_{z \in Z} \beta_\mu(z)$.

We construct the formulas $\varphi, \varphi_\mathbb{C}, \varphi_x$, as well as the parameters H, $\mathbb{F}$, m and ℓ, exactly as in Section 2.5 but with respect to the circuit $\mathbb{C}'_n$ (of size N') rather than $\mathbb{C}_n$ (of size N). We also construct the corresponding functions $\phi, \phi_{\mathbb{C}'}, \phi_x$ and $\hat{\phi}, \hat{\phi}_{\mathbb{C}'}, \hat{\phi}_x$ as in Section 2.5.

In addition, we construct a formula $\varphi_{extra}(w_1, \ldots, w_{N'})$ as follows. For every $i \in [N']$, the formula φ_{extra} contains a (seemingly redundant) clause that verifies that w_i has a *Boolean* value. Additionally, for every $\mu \in [t]$, we add to φ_{extra} clauses that verify that each one of the output gates of the corresponding $\mathbb{C}_{\mathsf{LDE}}$ circuit of layer μ has value 1. In other words,

$$\varphi_{extra}(w_1, \ldots, w_{N'}) = \bigwedge_{i \in [N']} \big((w_i = 0) \vee (w_i = 0) \vee (w_i = 1)\big) \wedge \bigwedge_{i \text{ is output gate of } \mathbb{C}_{\mathsf{LDE}}} \big((w_i = 1) \vee (w_i = 1) \vee (w_i = 1)\big).$$

Let $\phi_{extra} : (H^m)^3 \times \{0,1\}^3 \to \{0,1\}$ be the function where $\phi_{extra}(i_1, i_2, i_3, b_1, b_2, b_3) = 1$ if and only if the clause $(w_{i_1} = b_1) \vee (w_{i_2} = b_2) \vee (w_{i_3} = b_3)$ appears in φ_{extra}. Extend ϕ_{extra} to be a function $\phi_{extra} : H^{3m+3} \to \{0,1\}$ by setting it to be 0 for inputs outside of $H^{3m} \times \{0,1\}^3$. Let $\hat{\phi}_{extra} : \mathbb{F}^\ell \to \mathbb{F}$ be the low-degree extension of ϕ_{extra}.

Since there is an $O(\log N')$ space deterministic Turing machine that on input n outputs φ_{extra}, by Proposition 2.2, the function $\hat{\phi}_{extra}$ can be computed in $O(\log N')$ space.

Let $\varphi' = \varphi \wedge \varphi_{extra}$ and let $\phi' : (H^m)^3 \times \{0,1\}^3 \to \{0,1\}$ be the function where $\phi'(i_1, i_2, i_3, b_1, b_2, b_3) = 1$ if and only if the clause $(w_{i_1} = b_1) \vee (w_{i_2} = b_2) \vee (w_{i_3} = b_3)$ appears in φ'. Extend ϕ' to be a function $\phi' : H^{3m+3} \to \{0,1\}$ by setting it to be 0 for inputs outside of $H^{3m} \times \{0,1\}^3$. Let $\hat{\phi}' : \mathbb{F}^\ell \to \mathbb{F}$ be the low-degree extension of ϕ'.

Since the sets of clauses of φ and φ_{extra} are disjoint, we have $\hat{\phi}' = \hat{\phi} + \hat{\phi}_{extra} = \hat{\phi}_x + \hat{\phi}_{\mathbb{C}'} + \hat{\phi}_{extra}$.

The PCP proof (i.e., the polynomials $X, P_0, \ldots, P_\ell$) is constructed exactly as in Section 2.5.1 except that we use the circuit $\mathbb{C}'_n$ and the formula φ' (rather than $\mathbb{C}_n$ and φ). The PCP verifier V (resp., the relaxed verifier V') is constructed exactly as in Section 2.5.2 (resp., Section 2.5.3) with respect to the new PCP proof.

2.10 Soundness of V' in the Augmented PCP

In this section we will show that the relaxed verifier V' cannot be fooled to accept $x \notin \mathcal{L}$, with probability close to 1.

Recall that $k \leq \mathsf{poly}(n)$, such that $4|\mathbb{F}|^4 \leq k \leq N'$, is the security parameter of the PCP, and that $1 \leq r < k$ is the parameter of the relaxed verifier V'. Recall that ℓ and $|\mathbb{F}|$ are bounded by $\mathsf{polylog}(N')$.

Recall that t is the depth of the (original) circuit $\mathbb{C}_n$ and that $d = O(\log s)$ is the depth of the circuit $\mathbb{C}_{\mathsf{LDE}}$. We will prove the following lemma.

Lemma 2.31. *Assume that $k_{max} \geq k\mathsf{polylog}(s)\log(t)|\mathbb{F}| + 6k\ell|\mathbb{F}|^2$. Assume that $\delta < \frac{1}{1000N'\ell|\mathbb{F}|}$. Fix $\epsilon = \frac{1}{100N'\ell|\mathbb{F}|}$, and note that $\epsilon > 10\max\left(\delta, \frac{2k}{|\mathbb{F}|^{m-2}}\right)$. Assume $r < \frac{k}{20\ell|\mathbb{F}|}$. Then, V' has soundness $1-\epsilon$ against (k_{max}, δ)-no-signaling strategies.*

The rest of the section is devoted to the proof of Lemma 2.31. From now on, through Section 2.10, fix $k_{max}, \delta, \epsilon, r$ to be as in the statement of Lemma 2.31 and fix

$$r' = 9\ell|\mathbb{F}|r$$

(note that $r' < k/2$). We also fix a parameter

$$\nu = 10(\log(t) + d).$$

We will assume for a contradiction that for some $x \notin \mathcal{L}$, there exists a δ-no-signaling family of distributions $\{\mathcal{A}_S\}_{S \subset D, |S| \leq k_{max}}$ that fools V' with probability larger than $1-\epsilon$. That is, the verifier V' accepts with probability $> 1-\epsilon$, where on queries Q, the answers are given (probabilistically) by $A \in_R \mathcal{A}_Q$.

2.10.1 Reading Multiple Points Together

Let $B \subseteq H^m$ and let $\alpha : B \to \{0,1\}$. We think of B as specifying a subset of the variables of the formula φ' and of α as an assignment to B. We say that α is *consistent* with respect to B if it satisfies all the clauses of φ' in which *only* variables in B appear.

Lemma 2.32. *Let $B \subseteq H^m$ such that $3k|\mathbb{F}||B| < k_{max}$.*

For every $i \in B$, let $L^i_1, \ldots, L^i_k : \mathbb{F} \to D_X$ be k random lines, such that for every $L \in \{L^i_1, \ldots, L^i_k\}$, we have $L(0) = i$. Let $S = \{L^i_j(t)\}_{i \in B, j \in [k], t \in \mathbb{F}} \subset D_X$. Let $A \in_R \mathcal{A}_S$.

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 $A^{i \to v}(i) = v$.

Then, with probability $\geq 1 - 200|B|^3\ell|\mathbb{F}|\epsilon$, there exists $\alpha : B \to \{0,1\}$ that is consistent with respect to B, such that for every $i \in B$, for at least $k - r'$ of the indices $j \in [k]$, it holds that $A^{i \to \alpha(i)} \circ L^i_j : \mathbb{F} \to \mathbb{F}$ is a univariate polynomial of degree $< m|H|$ (where the probability is over $\{L^i_j\}_{i \in B, j \in [k]}, A$).

Proof. For every $i \in B$, let $L^i_1, \ldots, L^i_k, L^i_{k+1}, \ldots, L^i_{2k}, L^i_{2k+1}, \ldots, L^i_{3k} : \mathbb{F} \to D_X$ be $3k$ random lines, such that for every $L \in \left\{L^i_j\right\}_{j\in[3k]}$, we have $L(0) = i$. Let $S = \{L^i_j(t)\}_{i\in B, j\in[3k], t\in\mathbb{F}} \subset D_X$. Let $A \in_R \mathcal{A}_S$.

By Lemma 2.20, using also Claim 2.5.1, for every $i \in B$, with probability $\geq 1 - 10|\mathbb{F}|\epsilon - \delta$, there exists $v_i \in \mathbb{F}$, such that, for at least $3k - r'$ of the indices $j \in [3k]$, $A^{i\to v_i} \circ L^i_j : \mathbb{F} \to \mathbb{F}$ is a univariate polynomial of degree $< m|H|$ (where the probability is over $\{L^i_j\}_{i\in B, j\in[3k]}, A$).[14]

Let $\psi = (w_{i_1} = b_1 \vee w_{i_2} = b_2 \vee w_{i_3} = b_3)$ be a clause in φ' such that $i_1, i_2, i_3 \in B$. By Lemma 2.22 (using also Claim 2.5.1), with probability $\geq 1 - 9\ell|\mathbb{F}|\epsilon - 2\delta$ there exist $v^{(\psi)}_1, v^{(\psi)}_2, v^{(\psi)}_3 \in \mathbb{F}$ such that for at least $k - r'$ of the indices $j \in [k]$ it holds that:

1. $A^{i_1\to v^{(\psi)}_1} \circ L^{i_1}_j : \mathbb{F} \to \mathbb{F}$ is a univariate polynomial of degree $< m|H|$.
2. $A^{i_2\to v^{(\psi)}_2} \circ L^{i_2}_{k+j} : \mathbb{F} \to \mathbb{F}$ is a univariate polynomial of degree $< m|H|$.
3. $A^{i_3\to v^{(\psi)}_3} \circ L^{i_3}_{2k+j} : \mathbb{F} \to \mathbb{F}$ is a univariate polynomial of degree $< m|H|$.
4. $(v^{(\psi)}_1 - b_1) \cdot (v^{(\psi)}_2 - b_2) \cdot (v^{(\psi)}_3 - b_3) = 0$.

By the union bound, and since B contains at most $8|B|^3$ clauses, with probability $\geq 1 - |B|(10|\mathbb{F}|\epsilon + \delta) - 8|B|^3(9\ell|\mathbb{F}|\epsilon + 2\delta) > 1 - 100|B|^3\ell|\mathbb{F}|\epsilon$, for every $i \in B$ there exists $v_i \in \mathbb{F}$ and for every clause $\psi = (w_{i_1} = b_1 \vee w_{i_2} = b_2 \vee w_{i_3} = b_3)$ in φ' that contains only variables from B, there exist $v^{(\psi)}_1, v^{(\psi)}_2, v^{(\psi)}_3 \in F$ such that:

1. For at least $3k - r'$ of the indices $j \in [3k]$, $A^{i\to v_i} \circ L^i_j : \mathbb{F} \to \mathbb{F}$ is a univariate polynomial of degree $< m|H|$.
2. For at least $k - r'$ of the indices $j \in [k]$,
   (a) $A^{i_1\to v^{(\psi)}_1} \circ L^{i_1}_j : \mathbb{F} \to \mathbb{F}$ is a univariate polynomial of degree $< m|H|$.
   (b) $A^{i_2\to v^{(\psi)}_2} \circ L^{i_2}_{k+j} : \mathbb{F} \to \mathbb{F}$ is a univariate polynomial of degree $< m|H|$.
   (c) $A^{i_3\to v^{(\psi)}_3} \circ L^{i_3}_{2k+j} : \mathbb{F} \to \mathbb{F}$ is a univariate polynomial of degree $< m|H|$.
   (d) $(v^{(\psi)}_1 - b_1) \cdot (v^{(\psi)}_2 - b_2) \cdot (v^{(\psi)}_3 - b_3) = 0$.

where the probability is over $\{L^i_j\}_{i\in B, j\in[3k]}, A$. But since $r' + r' < k$, the latter implies that for every clause $\psi = (w_{i_1} = b_1 \vee w_{i_2} = b_2 \vee w_{i_3} = b_3)$ it holds that $v^{(\psi)}_1 = v_{i_1}$, $v^{(\psi)}_2 = v_{i_2}$ and $v^{(\psi)}_3 = v_{i_3}$ and in particular, $(v_{i_1} - b_1) \cdot (v_{i_2} - b_2) \cdot (v_{i_3} - b_3) = 0$. Furthermore, since for every $i \in B$ there is a clause of the form $i_1 = i_2 = i_3 = i$ and $b_1 = b_2 = 0$ and $b_3 = 1$ (indeed, these clauses were added in φ_{extra} to ensure a Boolean value), for every $i \in B$ it holds that $v_i \cdot v_i \cdot (v_i - 1) = 0$ and so $v_i \in \{0, 1\}$.

[14] We note that the statement of Lemma 2.20 refers to a smaller value of r' but of course, in particular, it also holds for larger values of r'.

Thus, with probability $\geq 1 - 100|B|^3\ell|\mathbb{F}|\epsilon$, there exists an assignment $\alpha : B \to \{0,1\}$ that is consistent with respect to B such that for every $i \in B$, for at least $3k - r'$ of the indices $j \in [3k]$ it holds that $A^{i\to\alpha(i)} \circ L^i_j : \mathbb{F} \to \mathbb{F}$ is a univariate polynomial of degree $< m|H|$ (where the probability is over $\{L^i_j\}_{i\in B, j\in[3k]}, A$).

The lemma follows from Claim 2.5.1. □

2.10.2 The Main Lemma

Fix a layer $\mu \in [t]$ of the circuit $\mathbb{C}'_n$. Recall that:

- $\beta_\mu \subset [N']$ refers to the set of indices of gates in $\mathbb{C}'_n$ that are associated with the LDE of the μ-th layer of $\mathbb{C}_n$.
- For every point $z \in \mathbb{G}^{m_\mathbb{G}}$ we denote by $\beta_\mu[z] \subset \beta_\mu$ the set of indices of the $\log_2|\mathbb{G}|$ gates associated with the point z in the computation of the LDE of layer μ in $\mathbb{C}'_n$. For a sequence $Z \subset \mathbb{G}^{m_\mathbb{G}}$ we denote by $\beta_\mu[Z] \stackrel{\text{def}}{=} \cup_{z\in Z}\beta_\mu(z)$.
- The values $x_1, \ldots, x_{N'}$ denote the computation of the circuit $\mathbb{C}'_n$ on the input $(x_1, \ldots, x_n)$.

Lemma 2.33. *Let $\lambda \in \beta_\mu$ be a fixed point and let $Z = (z_1, \ldots, z_\nu)$ be a sequence of ν points, where each point z_i is uniformly distributed in $\mathbb{G}^{m_\mathbb{G}}$.*

Let $B = \beta_\mu[Z] \cup \{\lambda\}$. We view the random variable B as being distributed over subsets of $H^m \subset D_X$.

For every $i \in B$, let $L^i_1, \ldots, L^i_k : \mathbb{F} \to D_X$ be k random lines, such that for every $L \in \{L^i_1, \ldots, L^i_k\}$, we have $L(0) = i$.

Let $S = \{L^i_j(t)\}_{i\in B, j\in[k], t\in\mathbb{F}} \subset D_X$. Let $A \in_R \mathcal{A}_S$.

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 $A^{i\to v}(i) = v$.

Let $\eta > 0$. Suppose that with probability $\geq 1-\eta$, for every $i \in \beta_\mu[Z]$, for at least $k - r'$ of the lines $L \in \{L^i_1, \ldots, L^i_k\}$, we have that $A^{i\to x_i} \circ L : \mathbb{F} \to \mathbb{F}$ is a univariate polynomial of degree $< m|H|$ (where the probability is over $Z, \{L^i_j\}_{i\in B, j\in[k]}, A$).

Then, with probability $\geq 1 - \eta - \mathsf{polylog}(s) \cdot \nu^3\ell|\mathbb{F}|\epsilon - 2^{-\nu}$, for every $i \in \{\lambda\} \cup \beta_\mu[Z]$, for at least $k - r'$ of the lines $L \in \{L^i_1, \ldots, L^i_k\}$, we have that $A^{i\to x_i} \circ L : \mathbb{F} \to \mathbb{F}$ is a univariate polynomial of degree $< m|H|$ (where the probability is over $Z, \{L^i_j\}_{i\in B, j\in[k]}, A$).

Proof. Let $z_0 \in \mathbb{G}^{m_\mathbb{G}}$ be the point such that $\lambda \in \beta_\mu[z_0]$ (i.e., λ belongs to the block associated with the point z_0). Let $Z = (z_1, \ldots, z_\nu)$ be a sequence of ν uniformly distributed points in $\mathbb{G}^{m_\mathbb{G}}$. For every $i \in [\nu]$, let $L_{z_0,z_i} : \mathbb{G} \to \mathbb{G}^{m_\mathbb{G}}$ be the line $L_{z_0,z_i}(t) = (z_i - z_0)\cdot t + z_0$ (i.e., the line that passes through the points z_0 and z_i).

For every line L_{z_0,z_i} let $B_{z_0,z_i} \subset [N']$ be the indices of all gates that are associated with the verification in $\mathbb{C}'_n$ that the LDE of layer μ restricted to the line L_{z_0,z_i} is a degree $m_\mathbb{G}|H_\mathbb{G}|$ univariate polynomial (recall that such gates are a part of the $\mathbb{C}_{\mathsf{LDE}}$ circuit of the μ-th layer of $\mathbb{C}'_n$, see Section 2.9). Let $B' = \cup_{i\in[\nu]} B_{z_o,z_i}$. Note that $|B'| = \nu \cdot \mathsf{polylog}(s)$

(since the verification can be implemented by a Boolean circuit of size $\mathsf{polylog}(s)$, see Section 2.9).

For every assignment $\alpha : B' \to \{0,1\}$, we denote by $\alpha_{\mathbb{G}} : \mathbb{G}^{m_{\mathbb{G}}} \to \mathbb{G}$ the partial function[15] $\alpha_{\mathbb{G}}(\zeta) \stackrel{\text{def}}{=} \alpha(\beta_\mu[\zeta]) \in \{0,1\}^{\log_2 |\mathbb{G}|}$ (where $\alpha_{\mathbb{G}}$ is only defined over $\cup_{i\in[\nu]}\{L_{z_0,z_i}(u) : u \in \mathbb{G}\}$). We say that α is *consistent* (w.r.t. the sequence Z) if for every $i \in [\nu]$ the function $\alpha_{\mathbb{G}} \circ L_{z_0,z_i}$ is a degree $m_{\mathbb{G}}|H_{\mathbb{G}}|$ (univariate) polynomial. We say that the assignment $\alpha : B' \to \{0,1\}$ is *correct* at the point $\zeta \in \cup_{i\in[\nu]}\{L_{z_0,z_i}(u) : u \in \mathbb{G}\}$ if for every $i \in \beta_\mu(\zeta) \subseteq B'$ it holds that $\alpha(i) = x_i$. We say that the assignment α is *correct* at a sequence of points if it is correct at every point in the sequence.

For every $i \in B'$, let $L^i_1, \ldots, L^i_k : \mathbb{F} \to D_X$ be k random lines, such that for every $L \in \{L^i_1, \ldots, L^i_k\}$, we have $L(0) = i$. Let $S = \{L^i_j(t)\}_{i\in B', j\in[k], t\in\mathbb{F}} \subset D_X$. Let $A \in_R \mathcal{A}_S$.

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 $A^{i\to v}(i) = v$.

Since the $\mathbb{C}_{\mathsf{LDE}}$ circuit of layer μ verifies that each line L_{z_0,z_i} has low degree, by applying Lemma 2.32 to the set B' (while noting that $3k|\mathbb{F}||B'| < k_{max}$) we have that, with probability $\geq 1 - 200(\nu \cdot \mathsf{polylog}(s))^3\ell|\mathbb{F}|\epsilon$, (over $\{L^i_j\}_{i\in B', j\in[k]}, A$), there exists a *consistent* assignment $\alpha : B' \to \{0,1\}$ such that for every $i \in B'$, for at least $k - r'$ of the indices $j \in [k]$, it holds that $A^{i\to\alpha(i)} \circ L^i_j : \mathbb{F} \to \mathbb{F}$ is a univariate polynomial of degree $< m|H|$.

On the other hand, by the lemma's hypothesis (using also Claim 2.5.1), with probability $\geq 1 - \eta - \delta$ (over $Z, \{L^i_j\}_{i\in B', j\in[k]}, A$), for every $i \in \beta_\mu[Z]$, for at least $k - r'$ of the lines $L \in \{L^i_1, \ldots, L^i_k\}$, we have that $A^{i\to x_i} \circ L : \mathbb{F} \to \mathbb{F}$ is a univariate polynomial of degree $< m|H|$.

Let E be the event that there exists a *consistent* assignment $\alpha : B' \to \{0,1\}$ *that is correct on* Z such that for every $i \in B'$, for at least $k - r'$ of the indices $j \in [k]$, it holds that $A^{i\to\alpha(i)} \circ L^i_j : \mathbb{F} \to \mathbb{F}$ is a univariate polynomial of degree $< m|H|$.

By the union bound (and using the fact that $r' + r' < k$),

$$\Pr[E] \geq 1 - \eta - \mathsf{polylog}(s) \cdot \nu^3\ell|\mathbb{F}|\epsilon.$$

where the probability is over $Z, \{L^i_j\}_{i\in B', j\in[k]}, A$.

Let E' be the event there exists a consistent assignment $\alpha : B' \to \{0,1\}$ *that is incorrect at the point* z_0 such that for every $i \in B'$, for at least $k - r'$ of the indices $j \in [k]$, it holds that $A^{i\to\alpha(i)} \circ L^i_j : \mathbb{F} \to \mathbb{F}$ is a univariate polynomial of degree $< m|H|$.

Consider the event $E \wedge E'$. If both E and E' occur then, by their definitions:

1. There exists a consistent assignment $\alpha : B' \to \{0,1\}$ that is correct on Z such that for every $i \in B'$, for at least $k - r'$ of the indices $j \in [k]$, it holds that $A^{i\to\alpha(i)} \circ L^i_j : \mathbb{F} \to \mathbb{F}$ is a univariate polynomial of degree $< m|H|$.

[15] We use $\alpha(\beta_\mu[\zeta])$ to denote the element in $\mathbb{G}$ that is obtained by considering the assignment α applied to the gates indexed by $\beta_\mu[\zeta]$ in $\mathbb{C}'_n$ and interpreting the resulting $\log_2 \mathbb{G}$ string as the corresponding element in $\mathbb{G}$.

2. There exists a consistent assignment $\alpha' : B' \to \{0,1\}$ that is incorrect at the point z_0 such that for every $i \in B'$, for at least $k - r'$ of the indices $j \in [k]$, it holds that $A^{i \to \alpha'(i)} \circ L^i_j : \mathbb{F} \to \mathbb{F}$ is a univariate polynomial of degree $< m|H|$.

However, since $r' + r' < k$ the assignment α must agree with α' on every $i \in B'$. Thus, if the event $E \wedge E'$ occurs then there exists a *single* consistent assignment $\alpha : B' \to \{0,1\}$ that is correct on Z *and* incorrect at the point z_0 such that for every $i \in B'$, for at least $k - r'$ of the indices $j \in [k]$, it holds that $A^{i \to \alpha(i)} \circ L^i_j : \mathbb{F} \to \mathbb{F}$ is a univariate polynomial of degree $< m|H|$.

We proceed to compute the probability that the event $E \wedge E'$ occurs. First observe that the sequence Z can be generated by using the following random process. First a sequence $Z' = (z'_1, \ldots, z'_\nu)$ of ν uniformly random points in $\mathbb{G}^{m_\mathbb{G}}$ is selected. For every $i \in [\nu]$, let $L_{z_0,z'_i}(t) = (z'_i - z_0)t + z_0$ be the line that passes through the points z_0 and z'_i. For every $i \in [\nu]$, the point z_i is selected by choosing at random $u_i \in \mathbb{G} \backslash \{0\}$ and setting $z_i = L_{z_0,z'_i}(u_i)$. Note that each one of the sequences Z and Z' is a sequence of ν uniformly distributed points in $\mathbb{G}^{m_\mathbb{G}}$.

Let $\chi : B' \to \{0,1\}$ denote the assignment of correct values to B'. That is, for every $i \in B'$, it holds that $\chi(i) = x_i$. Note that Z' already determines the set B' and that $Z', \{L^i_j\}_{i \in B', j \in [k]}, A$ already determine whether the event E' occurs (regardless of the choice of Z). Furthermore, if $Z', \{L^i_j\}_{i \in B', j \in [k]}, A$ are such that the event E' occurs, then the assignment α (guaranteed by E') is consistent and *incorrect* at the point z_0. Thus, for every $i \in [\nu]$ the two polynomials $\alpha_\mathbb{G} \circ L_{z_0,z'_i}$ and $\chi_\mathbb{G} \circ L_{z_0,z'_i}$ differ (at the point 0) and have degree at most $m_\mathbb{G}|H_\mathbb{G}|$. Hence, the two polynomials can agree on at most $m_\mathbb{G}|H_\mathbb{G}| < \frac{|\mathbb{G}|-1}{2}$ points, or in other words, for every $i \in [\nu]$ the assignment α is correct on less than half of the points on the line L_{z_0,z'_i}. Thus, we have:

$$\begin{aligned}
\Pr_{Z,\{L^i_j\}_{i \in B', j \in [k]}, A}[E \wedge E'] &= \mathop{\mathbf{E}}_{Z',\{L^i_j\}_{i \in B', j \in [k]}, A}\left[\Pr_{u_1,\ldots,u_\nu}[E \wedge E']\right] \\
&= \Pr[E'] \cdot \mathop{\mathbf{E}}_{Z',\{L^i_j\}_{i \in B', j \in [k]}, A}\left[\Pr_{u_1,\ldots,u_\nu}[E \wedge E'] \middle| E'\right] + \\
&\quad \Pr[\neg E'] \cdot \mathop{\mathbf{E}}_{Z',\{L^i_j\}_{i \in B', j \in [k]}, A}\left[\Pr_{u_1,\ldots,u_\nu}[E \wedge E'] \middle| \neg E'\right]
\end{aligned}$$

However, if Z', $\{L^i_j\}_{i \in B', j \in [k]}$ and A are such that $\neg E'$ occurs then $\Pr_{u_1,\ldots,u_\nu}[E \wedge E'] = 0$. On the other hand, by the foregoing discussion, if Z', $\{L^i_j\}_{i \in B', j \in [k]}$, and A are such that E' occurs then $\Pr_{u_1,\ldots,u_\nu}[E \wedge E'] \leq 2^{-\nu}$. Thus:

$$\Pr_{Z,\{L^i_j\}_{i \in B', j \in [k]}, A}[E \wedge E'] \leq \Pr[E'] \cdot 2^{-\nu} \leq 2^{-\nu}$$

and so

$$\Pr[E \wedge \neg E'] = \Pr[E] - \Pr[E \wedge E'] \geq 1 - \eta - \mathsf{polylog}(s) \cdot \nu^3 \ell |\mathbb{F}| \epsilon - 2^{-\nu}.$$

In other words, with probability $1 - \eta - \mathsf{polylog}(s) \cdot \nu^3 \ell |\mathbb{F}| \epsilon - 2^{-\nu}$, there exists a consistent assignment $\alpha : B' \to \{0,1\}$ *that is correct on* Z *and on* z_0 such that for every $i \in B'$, for at least $k - r'$ of the indices $j \in [k]$, it holds that $A^{i \to \alpha(i)} \circ L^i_j : \mathbb{F} \to \mathbb{F}$ is a univariate polynomial of degree $< m|H|$. The lemma follows by Claim 2.5.1.

□

2.10.3 Some Useful Claims

Claim 2.33.1. *Let* $S \subset D, |S| \leq k_{max}$ *be a set generated by some random process. Let* $A \in_R \mathcal{A}_S$. *Let* $g(S, A)$ *be a predicate such that* $\Pr_{A,S}[g(S,A)] \geq 1/2$. *Let* $f(S, A)$ *be a predicate such that* $\Pr_{A,S}[f(S,A) \mid g(S,A)] = p$. *Let* S', Q, *such that* $S' \subseteq Q \subset D, |Q| \leq k_{max}$, *be two sets generated by some random process, such that the distribution of* S' *is identical to the distribution of* S. *Let* $A' \in_R \mathcal{A}_Q$. *Then,*

$$p - 4\delta \leq \Pr_{A',S',Q}\left[f(S', A'_{S'}) \mid g(S', A'_{S'})\right] \leq p + 4\delta$$

.

Proof. Denote:

$$\begin{aligned}
a &\stackrel{\text{def}}{=} \Pr_{S, A \in_R \mathcal{A}_s}[f(S,A) \wedge g(S,A)] \\
b &\stackrel{\text{def}}{=} \Pr_{S, A \in_R \mathcal{A}_s}[g(S,A)] \\
c &\stackrel{\text{def}}{=} \Pr_{Q, S', A' \in_R \mathcal{A}_Q}[f(S', A'_{S'}) \wedge g(S', A'_{S'})] \\
d &\stackrel{\text{def}}{=} \Pr_{Q, S', A' \in_R \mathcal{A}_Q}[g(S', A'_{S'})]
\end{aligned}$$

By Claim 2.5.1, $|a - c| < \delta$ and $|b - d| < \delta$ (and in particular $d \geq 1/2 - \delta > 0.4$ and therefore the conditional probability space in the lemma's conclusion is non-empty). Note that:

$$\begin{aligned}
\frac{a}{b} &= \Pr_{S, A \in_R \mathcal{A}_s}[f(S,A) \mid g(S,A)] \\
\frac{c}{d} &= \Pr_{Q, S', A' \in_R \mathcal{A}_Q}[f(S', A'_{S'}) \mid g(S', A'_{S'})].
\end{aligned}$$

Using elementary manipulations we have that,

$$\left|\frac{a}{b} - \frac{c}{d}\right| = \frac{|ad - bc|}{bd} = \frac{|ad - cd + cd - bc|}{bd} \leq \frac{d|a-c| + c|d-b|}{bd} \leq \frac{|a-c|}{b} + \frac{|d-b|}{b} \cdot c/d \leq 4\delta$$

where the last inequality uses also the hypothesis that $b \geq 1/2$ and the fact that $c \leq d$. □

Claim 2.33.2. *Let* $\gamma \geq 0$ *and let* A *and* B *be events over the same probability space such that* $\Pr[A] \geq 1 - \gamma$ *and* $\Pr[B] \geq \frac{1}{2}$. *Then* $\Pr[A|B] \geq 1 - 2\gamma$.

Proof.

$$\Pr[A|B] = \frac{\Pr[A \wedge B]}{\Pr[B]} \geq \frac{\Pr[A] + \Pr[B] - 1}{\Pr[B]} \geq 1 - \frac{\gamma}{\Pr[B]} \geq 1 - 2\gamma.$$

□

Claim 2.33.3. *Let $\gamma, \eta < 1$ and let A and B be events over the same probability space such that $\Pr[B] \geq 1 - \gamma$ and $\Pr[A|B] \geq 1 - \eta$. Then $\Pr[A] \geq 1 - \gamma - \eta$.*

Proof.

$$\Pr[A] \geq \Pr[A \wedge B] = \Pr[A|B] \cdot \Pr[B] \geq (1 - \gamma)(1 - \eta) \geq 1 - \gamma - \eta.$$

□

Claim 2.33.4 (Union Bound under Conditioning)**.** *Let A, B and C be events over the same probability space, such that C has non-zero probability. Then:*

$$\Pr[A \vee B|C] \leq \Pr[A|C] + \Pr[B|C].$$

Proof.

$$\Pr[A \vee B|C] = \frac{\Pr[(A \vee B) \wedge C]}{\Pr[C]} \leq \frac{\Pr[(A \wedge C)] + \Pr[(B \wedge C)]}{\Pr[C]} = \Pr[A|C] + \Pr[B|C].$$

□

2.10.4 The Property $\mathbb{R}_\mu$ and making Progress under Conditioning

We will now give two definitions that will be central in the rest of the section. The first definition is analogous to the definition of the property $\mathbb{R}$ (Definition 2.23) in Section 2.7.6.

Definition 2.34. Property $\mathbb{R}_\mu(\epsilon', r')$:
Let $\mu \in [t]$, $\epsilon' \geq 0$ and $r' \geq 0$.

Let $B \subseteq [N']$ such that $(|B| + \nu \cdot \log_2 |\mathbb{G}|) \cdot k|\mathbb{F}| < k_{max}$. Let Z be a sequence of ν uniformly distributed points in $\mathbb{G}^{m_\mathbb{G}}$. Let $B' = B \cup \beta_\mu[Z]$. We view both B and B' as being distributed over subsets of $H^m \subseteq D_X$.

For every $i \in B'$, let $L^i_1, \ldots, L^i_k : \mathbb{F} \to D_X$ be k random lines, such that for every $L \in \{L^i_1, \ldots, L^i_k\}$, we have $L(0) = i$.

Let $S = \{L^i_j(t)\}_{i \in B', j \in [k], t \in \mathbb{F}} \subset D_X$. Let $A \in_R \mathcal{A}_S$.

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 $A^{i \to v}(i) = v$.

Denote by E the event that for every point $i \in \beta_\mu[Z]$, for at least $k - r'$ of the lines $L \in \{L^i_1, \ldots, L^i_k\}$, we have that $A^{i \to x_i} \circ L : \mathbb{F} \to \mathbb{F}$ is a univariate polynomial of degree $< m|H|$.

We say that the set B satisfies property $\mathbb{R}_\mu(\epsilon', r')$ (also denoted $B \in \mathbb{R}_\mu(\epsilon', r')$) if, conditioned on the event E, with probability $\geq 1-\epsilon'$, for every point $i \in B$, for at least $k - r'$ of the lines $L \in \{L_1^i, \ldots, L_k^i\}$, we have that $A^{i \to x_i} \circ L : \mathbb{F} \to \mathbb{F}$ is a univariate polynomial of degree $< m|H|$ (where the probability is over $Z, \{L_j^i\}_{i \in B, j \in [k]}, A$).

To ensure that R_μ is well defined, if $\Pr[E] = 0$, then no set B is said to satisfy $\mathbb{R}_\mu(\epsilon', r')$.

Definition 2.35. ***p*-good layers:**
Let $\mu \in [t]$. Let Z be a sequence of ν uniformly distributed points in $\mathbb{G}^{m_\mathbb{G}}$.

For every $i \in \beta_\mu[Z]$, let $L_1^i, \ldots, L_k^i : \mathbb{F} \to D_X$ be k random lines, such that for every $L \in \{L_1^i, \ldots, L_k^i\}$, we have $L(0) = i$.

Let $S = \{L_j^i(t)\}_{i \in \beta_\mu[Z], j \in [k], t \in \mathbb{F}} \subset D_X$. Let $A \in_R \mathcal{A}_S$.

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 $A^{i \to v}(i) = v$.

We say that the layer μ is p-good for $p \in [0,1]$ if, with probability $\geq p$, for every point $i \in \beta_\mu[Z]$, for at least $k - r'$ of the lines $L \in \{L_1^i, \ldots, L_k^i\}$, we have that $A^{i \to x_i} \circ L : \mathbb{F} \to \mathbb{F}$ is a univariate polynomial of degree $< m|H|$ (where the probability is over $Z, \{L_j^i\}_{i \in \beta_\mu[Z], j \in [k]}, A$).

Lemma 2.36. *Let $i_1, i_2, i_3 \in [N']$ be such that the gate indexed by i_1 in the circuit $\mathbb{C}'_n$ has children indexed by i_2, i_3. Let $\mu \in [t]$ be a 0.9-good layer. If $\{i_2\}, \{i_3\} \in \mathbb{R}_\mu(\epsilon', r')$, then $\{i_1\} \in \mathbb{R}_\mu(\epsilon'', r')$ where $\epsilon'' = 2\epsilon' + 34\ell|\mathbb{F}|\epsilon$.*

Proof. Let Z be a sequence of ν uniformly distributed points in $\mathbb{G}^{m_\mathbb{G}}$. Let $B = \{i_1, i_2, i_3\} \cup \beta_\mu[Z]$. We view B as being distributed over subsets of $H^m \subseteq D_X$.

For every $i \in B$, let $L_1^i, \ldots, L_k^i : \mathbb{F} \to D_X$ be k random lines, such that for every $L \in \{L_1^i, \ldots, L_k^i\}$, we have $L(0) = i$.

Let $S = \{L_j^i(t)\}_{i \in B, j \in [k], t \in \mathbb{F}} \subset D_X$. Let $A \in_R \mathcal{A}_S$.

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 $A^{i \to v}(i) = v$.

Denote by E the event that for every point $i \in \beta_\mu[Z]$, for at least $k - r'$ of the lines $L \in \{L_1^i, \ldots, L_k^i\}$, we have that $A^{i \to x_i} \circ L : \mathbb{F} \to \mathbb{F}$ is a univariate polynomial of degree $< m|H|$. Since μ is a 0.9-good layer (using also Claim 2.5.1), the event E occurs with probability $\geq 0.9 - \delta > 1/2$ (where the probability is over $Z, \{L_j^i\}_{i \in B, j \in [k]}, A$).

Since $\{i_2\} \in \mathbb{R}_\mu(\epsilon', r')$, using also Claim 2.33.1, conditioned on the event E occurring, with probability $\geq 1 - \epsilon' - 4\delta$ for at least $k - r'$ of the lines $L \in \{L_1^{i_2}, \ldots, L_k^{i_2}\}$, we have that $A^{i_2 \to x_{i_2}} \circ L : \mathbb{F} \to \mathbb{F}$ is a univariate polynomial of degree $< m|H|$ (where the probability is over $Z, \{L_j^i\}_{i \in B, j \in [k]}, A$).

Similarly, since $i_3 \in \mathbb{R}_\mu(\epsilon', r')$, conditioned on the event E occurring, with probability $\geq 1 - \epsilon' - 4\delta$ for at least $k - r'$ of the lines $L \in \{L_1^{i_3}, \ldots, L_k^{i_3}\}$, we have that $A^{i_3 \to x_{i_3}} \circ L : \mathbb{F} \to \mathbb{F}$ is a univariate polynomial of degree $< m|H|$ (where the probability is over $Z, \{L_j^i\}_{i \in B, j \in [k]}, A$).

Since the gate indexed by i_1 in the circuit $\mathbb{C}'_n$ has children indexed by i_2, i_3, the formula φ contains the clause $(w_{i_2} = x_{i_2}) \wedge (w_{i_3} = x_{i_3}) \to (w_{i_1} = x_{i_1})$. Thus, by Lemma 2.22

(using also Claim 2.5.1), with probability $\geq 1-9\ell|\mathbb{F}|\epsilon-2\delta$, there exist $v_1, v_2, v_3 \in \mathbb{F}$, such that, for at least $k - r'$ of the indices $j \in [k]$,

1. $A^{i_1 \to v_1} \circ L_j^{i_1} : \mathbb{F} \to \mathbb{F}$ is a univariate polynomial of degree $< m|H|$.
2. $A^{i_2 \to v_2} \circ L_j^{i_2} : \mathbb{F} \to \mathbb{F}$ is a univariate polynomial of degree $< m|H|$.
3. $A^{i_3 \to v_3} \circ L_j^{i_3} : \mathbb{F} \to \mathbb{F}$ is a univariate polynomial of degree $< m|H|$.
4. $(v_2 = x_{i_2}) \wedge (v_3 = x_{i_3}) \to (v_1 = x_{i_1})$

Thus, using Claim 2.33.2, conditioned on the event E occurring, with probability $\geq 1 - 18\ell|\mathbb{F}|\epsilon - 4\delta$, there exist $v_1, v_2, v_3 \in \mathbb{F}$, such that, for at least $k - r'$ of the indices $j \in [k]$,

1. $A^{i_1 \to v_1} \circ L_j^{i_1} : \mathbb{F} \to \mathbb{F}$ is a univariate polynomial of degree $< m|H|$.
2. $A^{i_2 \to v_2} \circ L_j^{i_2} : \mathbb{F} \to \mathbb{F}$ is a univariate polynomial of degree $< m|H|$.
3. $A^{i_3 \to v_3} \circ L_j^{i_3} : \mathbb{F} \to \mathbb{F}$ is a univariate polynomial of degree $< m|H|$.
4. $(v_2 = x_{i_2}) \wedge (v_3 = x_{i_3}) \to (v_1 = x_{i_1})$.

Thus, by the union bound under conditioning (Claim 2.33.4), conditioned on the event E occurring, with probability $\geq 1 - 2\epsilon' - 18\ell|\mathbb{F}|\epsilon - 12\delta$, there exist $v_1, v_2, v_3 \in \mathbb{F}$, such that $(v_2 = x_{i_2}) \wedge (v_3 = x_{i_3}) \to (v_1 = x_{i_1})$ and:

- For at least $k - r'$ of the lines $L \in \{L_1^{i_2}, \ldots, L_k^{i_2}\}$, we have that $A^{i_2 \to x_{i_2}} \circ L : \mathbb{F} \to \mathbb{F}$ is a univariate polynomial of degree $< m|H|$.
- For at least $k - r'$ of the lines $L \in \{L_1^{i_3}, \ldots, L_k^{i_3}\}$, we have that $A^{i_3 \to x_{i_3}} \circ L : \mathbb{F} \to \mathbb{F}$ is a univariate polynomial of degree $< m|H|$.
- For at least $k - r'$ of the indices $j \in [k]$,
  1. $A^{i_1 \to v_1} \circ L_j^{i_1} : \mathbb{F} \to \mathbb{F}$ is a univariate polynomial of degree $< m|H|$.
  2. $A^{i_2 \to v_2} \circ L_j^{i_2} : \mathbb{F} \to \mathbb{F}$ is a univariate polynomial of degree $< m|H|$.
  3. $A^{i_3 \to v_3} \circ L_j^{i_3} : \mathbb{F} \to \mathbb{F}$ is a univariate polynomial of degree $< m|H|$.

Since $r' + r' < k$, this implies that $v_2 = x_{i_2}$, $v_3 = x_{i_3}$ and hence $v_1 = x_{i_1}$. Thus, conditioned on the event E occurring, with probability $\geq 1 - 2\epsilon' - 18\ell|\mathbb{F}|\epsilon - 12\delta > 1 - 2\epsilon' - 30\ell|\mathbb{F}|\epsilon$, for at least $k - r'$ of the lines $L \in \{L_1^{i_1}, \ldots, L_k^{i_1}\}$, we have that $A^{i_1 \to x_{i_1}} \circ L : \mathbb{F} \to \mathbb{F}$ is a univariate polynomial of degree $< m|H|$ (where the probability is over $Z, \{L_j^i\}_{i \in B, j \in [k]}, A$).

The lemma follows by an application of Claim 2.33.1. □

If C is a circuit, we say that a subset B of the gates of C is a sub-circuit of C if for every gate $g \in B$ either both of its children are in B or both are not in B. Gates in B whose children are not in B are called input gates of B and gates in B who are not children of any gate in B are called output gates of B. We say that the sub-circuit B has depth Δ if the longest path from an output gate of B to an input gate of B is of length Δ.

Using Lemma 2.36, we are ready to prove the following lemma.

Lemma 2.37. *Let $B \subset [N']$ be a sub-circuit of $\mathbb{C}'_n$ of depth Δ with input gates $B_I \subset B$ and output gates $B_O \subset B$, such that $(|B_O| + \nu \cdot \log_2 |\mathbb{G}|) \cdot k|\mathbb{F}| < k_{max}$. Let $\mu \in [t]$ be a 0.9-good layer of the circuit. If for all $i \in B_I$ it holds that $\{i\} \in \mathbb{R}_\mu(\epsilon', r')$ then $B_O \in \mathbb{R}_\mu(\epsilon'', r')$, where $\epsilon'' = |B_O| \cdot 2^\Delta \cdot (2\epsilon' + 38\ell|\mathbb{F}|\epsilon)$.*

Proof. For every $i \in B_O$, by iterated applications of Lemma 2.36, it holds that $\{i\} \in \mathbb{R}_\mu\left(2^\Delta \cdot (2\epsilon' + 34\ell|\mathbb{F}|\epsilon), r'\right)$. The lemma follows from $|B_O|$ applications of Claim 2.33.1, and the union bound under conditioning (Claim 2.33.4). □

We also prove (simpler) variants of Lemma 2.36 and Lemma 2.37 with respect to the property $\mathbb{R}$ (rather than $\mathbb{R}_\mu$), see Definition 2.23. Recall that, intuitively, a subset $B \subset H^m \subset D_X$ satisfies property $\mathbb{R}(\epsilon', r')$ if when taking k lines through every point in B, with high probability, for every point $i \in B$, for most of the lines through the point i, the answers correspond to low degree polynomials that "evaluate" the point i to x_i.

Lemma 2.38. *Let $i_1, i_2, i_3 \in [N']$ be such that the gate indexed by i_1 in the circuit $\mathbb{C}'_n$ has children indexed by i_2, i_3. If $\{i_2\}, \{i_3\} \in \mathbb{R}(\epsilon', r')$, then $\{i_1\} \in \mathbb{R}(\epsilon'', r')$ where $\epsilon'' = 2\epsilon' + 15\ell|\mathbb{F}|\epsilon$.*

Proof. If $\{i_2\}, \{i_3\} \in \mathbb{R}(\epsilon', r')$, then by Lemma 2.26, it holds that $\{i_2, i_3\} \in \mathbb{R}(2\epsilon' + 2\delta, r')$. Since the gate indexed by i_1 in the circuit $\mathbb{C}'_n$ has children indexed by i_2, i_3, by Lemma 2.25, it holds that $\{i_1, i_2, i_3\} \in \mathbb{R}(2\epsilon' + 9\ell|\mathbb{F}|\epsilon + 5\delta, r')$. The lemma follows from Lemma 2.27. □

Lemma 2.39. *Let $B \subset [N']$ be a sub-circuit of $\mathbb{C}'_n$ of depth Δ with input gates $B_I \subset B$ and output gates $B_O \subset B$, such that $|B_O|k|\mathbb{F}| < k_{max}$. If for all $i \in B_I$ it holds that $\{i\} \in \mathbb{R}(\epsilon', r')$ then $B_O \in \mathbb{R}(\epsilon'', r')$, where $\epsilon'' = |B_O| \cdot 2^\Delta \cdot (2\epsilon' + 16\ell|\mathbb{F}|\epsilon)$.*

Proof. For every $i \in B_O$, by iterated applications of Lemma 2.38, it holds that $\{i\} \in \mathbb{R}\left(2^\Delta \cdot (2\epsilon' + 15\ell|\mathbb{F}|\epsilon), r'\right)$. The lemma follows from $|B_O|$ applications of Claim 2.5.1, and the union bound. □

2.10.5 Proof of Lemma 2.31

In this section we complete the proof of Lemma 2.31. We first show that if layer $\mu - 1$ is a good layer then layer μ is also good. Then we derive that the top layer is good and use that to contradict our assumption that $x \notin \mathcal{L}$.

Lemma 2.40. *Let $\mu \in [t]$ be a 0.9-good layer. Then, for every $\lambda \in \beta_\mu$ it holds that $\lambda \in \mathbb{R}_\mu(\epsilon', r')$, where $\epsilon' = \mathsf{polylog}(s) \cdot \nu^3 \ell |\mathbb{F}| \epsilon + 2^{-\nu+1}$.*

Proof. Let $\lambda \in \beta_\mu$. Let Z be a sequence of ν uniformly distributed points in $\mathbb{G}^{m_\mathbb{G}}$. Let $B = \{\lambda\} \cup \beta_\mu[Z]$.

For every $i \in B$, let $L_1^i, \ldots, L_k^i : \mathbb{F} \to D_X$ be k random lines, such that for every $L \in \{L_1^i, \ldots, L_k^i\}$, we have $L(0) = i$.

Let $S = \{L_j^i(t)\}_{i \in B, j \in [k], t \in \mathbb{F}} \subset D_X$. Let $A \in_R \mathcal{A}_S$.

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 $A^{i \to v}(i) = v$.

Denote by E the event that for every point $i \in \beta_\mu[Z]$, for at least $k - r'$ of the lines $L \in \{L_1^i, \ldots, L_k^i\}$, we have that $A^{i \to x_i} \circ L : \mathbb{F} \to \mathbb{F}$ is a univariate polynomial of degree $< m|H|$ (where the probability is over $Z, \{L_j^i\}_{i \in B, j \in [k]}, A$). Suppose that $\Pr[E] = 1 - \eta$ for some $\eta \in [0, 1]$. Note that by the hypothesis that μ is 0.9-good, using also Claim 2.5.1, $\eta < 0.1 + \delta < 1/2$.

Denote by $E' \subset E$ the event that for every point $i \in \{\lambda\} \cup \beta_\mu[Z]$, for at least $k - r'$ of the lines $L \in \{L_1^i, \ldots, L_k^i\}$, we have that $A^{i \to x_i} \circ L : \mathbb{F} \to \mathbb{F}$ is a univariate polynomial of degree $< m|H|$ (where the probability is over $Z, \{L_j^i\}_{i \in B, j \in [k]}, A$). By Lemma 2.33, $\Pr[E'] \geq 1 - \eta - \mathsf{polylog}(s) \cdot \nu^3 \ell |\mathbb{F}| \epsilon - 2^{-\nu}$.

Thus, the probability that the event E' occurs conditioned on the event E is at least:

$$\Pr[E'|E] = \frac{\Pr[E']}{\Pr[E]} \geq \frac{1 - \eta - \mathsf{polylog}(s) \cdot \nu^3 \ell |\mathbb{F}| \epsilon - 2^{-\nu}}{1 - \eta} \geq 1 - \mathsf{polylog}(s) \cdot \nu^3 \ell |\mathbb{F}| \epsilon - 2^{-\nu+1}$$

(where the last inequality follows from the fact that $\eta \leq 1/2$) and the lemma follows. □

Lemma 2.41. *Let $\mu \in [t-1]$ be a 0.9-good layer. Then, for every set $B \subseteq \beta_{\mu+1}$ of points that belong to layer $\mu + 1$, such that $(|B| + \nu \cdot \log_2 |\mathbb{G}|) \cdot k|\mathbb{F}| < k_{max}$, it holds that $B \in \mathbb{R}_\mu(\epsilon', r')$, where $\epsilon' = |B| \cdot 2^d \cdot \left(\mathsf{polylog}(s) \cdot \nu^3 \ell |\mathbb{F}| \epsilon + 2^{-\nu+3}\right)$.*

Proof. Consider the sub-circuit that computes layer $\mu + 1$ from layer μ. Recall that this sub-circuit has depth $d + 1$ and first computes the $\mu + 1$-th layer of $\mathbb{C}_n$, in depth 1, and then applies a C_{LDE} circuit, of depth d (see Section 2.9). Let $B_I \subseteq \beta_\mu$ be the variables associated with the inputs of this sub-circuit. By Lemma 2.40, for every $\lambda \in B_I$ it holds that $\lambda \in \mathbb{R}_\mu(\mathsf{polylog}(s) \cdot \nu^3 \ell |\mathbb{F}| \epsilon + 2^{-\nu+1}, r')$.

The lemma follows from Lemma 2.37. □

Lemma 2.42. *If a layer $\mu \in [t-1]$ is $(1 - \epsilon')$-good, for some $\epsilon' < 0.1$, then the layer $\mu + 1$ is $(1 - \epsilon'')$-good, where $\epsilon'' = \epsilon' + 2^d \cdot \mathsf{polylog}(s) \cdot \left(\nu^4 \ell |\mathbb{F}| \epsilon + 2^{-\nu/2}\right)$.*

Proof. Let Z and Z' be two sequences of ν uniformly distributed points in $\mathbb{G}^{m_\mathbb{G}}$. Let $B = \beta_\mu[Z] \cup \beta_{\mu+1}[Z']$. We view B as being distributed over subsets of D_X.

For every $i \in B$, let $L_1^i, \ldots, L_k^i : \mathbb{F} \to D_X$ be k random lines, such that for every $L \in \{L_1^i, \ldots, L_k^i\}$, we have $L(0) = i$.

Let $S = \{L_j^i(t)\}_{i \in B, j \in [k], t \in \mathbb{F}} \subset D_X$. Let $A \in_R \mathcal{A}_S$.

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 $A^{i\to v}(i) = v$.

Denote by E the event that for every point $i \in \beta_\mu[Z]$, for at least $k - r'$ of the lines $L \in \{L^i_1, \ldots, L^i_k\}$, we have that $A^{i\to x_i} \circ L : \mathbb{F} \to \mathbb{F}$ is a univariate polynomial of degree $< m|H|$ (where the probability is over $\{Z, Z', L^i_j\}_{i\in B, j\in[k]}, A$). By the hypothesis that μ is $(1-\epsilon')$-good, and using Claim 2.5.1, the event E occurs with probability $\geq 1 - \epsilon' - \delta$.

By Lemma 2.41 (using the fact that $\epsilon' < 0.1$), it holds that $\beta_{\mu+1}[Z'] \in \mathbb{R}_\mu\Big((\log_2 \mathbb{G} \cdot \nu) \cdot 2^d \cdot \big(\mathsf{polylog}(s) \cdot \nu^3 \ell |\mathbb{F}| \epsilon + 2^{-\nu+3}\big), r'\Big)$. In other words, conditioned on the event E, with probability $\geq 1 - 2^d \mathsf{polylog}(s) \cdot \big(\nu^4 \ell |\mathbb{F}| \epsilon + 2^{-\nu/2}\big)$, for every point $i \in \beta_{\mu+1}[Z']$, for at least $k - r'$ of the lines $L \in \{L^i_1, \ldots, L^i_k\}$, we have that $A^{i\to x_i} \circ L : \mathbb{F} \to \mathbb{F}$ is a univariate polynomial of degree $< m|H|$ (where the probability is over $Z, Z', \{L^i_j\}_{i\in B, j\in[k]}, A$).

However, since E occurs with high probability, we can remove the conditioning as follows. Toward this end, we apply Claim 2.33.3 and obtain that with probability $\geq 1 - 2^d \cdot \mathsf{polylog}(s) \cdot \big(\nu^4 \ell |\mathbb{F}| \epsilon + 2^{-\nu/2}\big) - \epsilon' - \delta$, for every point $i \in \beta_{\mu+1}[Z']$, for at least $k - r'$ of the lines $L \in \{L^i_1, \ldots, L^i_k\}$, we have that $A^{i\to x_i} \circ L : \mathbb{F} \to \mathbb{F}$ is a univariate polynomial of degree $< m|H|$ (where the probability is over $Z, Z', \{L^i_j\}_{i\in B, j\in[k]}, A$). The lemma follows by Claim 2.5.1. □

Recall that $1, \ldots, n$ are the indexes of the n input variables and $n+1, \ldots, 2n$ are the indexes of their negations.

Lemma 2.43. *The first layer of $\mathbb{C}'_n$ is $(1-\epsilon')$-good, where $\epsilon' = 2^d \mathsf{polylog}(s) \cdot \nu \ell |\mathbb{F}| \epsilon$.*

Proof. By Lemma 2.24, for every $i \in [2n]$ it holds that $\{i\} \in \mathbb{R}\big(10\ell|\mathbb{F}|\epsilon, r'\big)$. Let $\mathbb{C}^{(1)}_{\mathsf{LDE}}$ be the $\mathbb{C}_{\mathsf{LDE}}$ circuit of the first layer of $\mathbb{C}'_n$. Note that the inputs of $\mathbb{C}^{(1)}_{\mathsf{LDE}}$ are associated with the variables $i \in [2n]$ and that $\mathbb{C}^{(1)}_{\mathsf{LDE}}$ has depth d. Thus, by Lemma 2.39, for every sequence of ν points Z in $\mathbb{G}^{m_\mathbb{G}}$ it holds that $\beta_1[Z] \in \mathbb{R}\Big(\big(\nu \cdot \log_2(|\mathbb{G}|)\big) \cdot 2^d \cdot \mathsf{polylog}(s) \cdot \ell |\mathbb{F}| \epsilon, r'\Big)$ and the lemma follows. □

Lemma 2.44. *The top layer of $\mathbb{C}'_n$ (i.e., the t-th layer) is $(1-\epsilon')$-good, where $\epsilon' \leq t \cdot 2^d \cdot \mathsf{polylog}(s)\big(\nu^4 \ell |\mathbb{F}| \epsilon + 2^{-\nu/2}\big)$.*

Proof. By induction, using Lemma 2.43 and Lemma 2.42. □

Recall that N' is the index of the special output gate.

Lemma 2.45. *$\{N'\} \in \mathbb{R}(\epsilon', r')$, where $\epsilon' = t \cdot 2^d \cdot \mathsf{polylog}(s)\big(\nu^4 \ell |\mathbb{F}| \epsilon + 2^{-\nu/2}\big)$.*

Proof. Let Z be a sequence of ν uniformly distributed points in $\mathbb{G}^{m_\mathbb{G}}$. Let $B = \{N'\} \cup \beta_t[Z]$. Note that the point N' belongs to layer t. We view B as being distributed over subsets of $H^m \subseteq D_X$.

For every $i \in B$, let $L^i_1, \ldots, L^i_k : \mathbb{F} \to D_X$ be k random lines, such that for every $L \in \{L^i_1, \ldots, L^i_k\}$, we have $L(0) = i$.

Let $S = \{L^i_j(t)\}_{i\in B, j\in[k], t\in\mathbb{F}} \subset D_X$. Let $A \in_R \mathcal{A}_S$.

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 $A^{i\to v}(i) = v$.

Denote by E the event that for every point $i \in \beta_t[Z]$, for at least $k - r'$ of the lines $L \in \{L^i_1, \ldots, L^i_k\}$, we have that $A^{i\to x_i} \circ L : \mathbb{F} \to \mathbb{F}$ is a univariate polynomial of degree $< m|H|$. By Lemma 2.44, and using Claim 2.5.1, the event E occurs with probability at least $1 - t \cdot 2^d \cdot \mathsf{polylog}(s)\big(\nu^4\ell|\mathbb{F}|\epsilon + 2^{-\nu/2}\big)$.

Since by our setting of parameters $t \cdot 2^d \cdot \mathsf{polylog}(s)\big(\nu^4\ell|\mathbb{F}|\epsilon + 2^{-\nu/2}\big) < 0.1$, the layer t is 0.9 good and so, by Lemma 2.40, it holds that $N' \in \mathbb{R}_t(\mathsf{polylog}(s) \cdot \nu^3\ell|\mathbb{F}|\epsilon + 2^{-\nu+1}, r')$. In other words, conditioned on the event E, with probability $\geq 1 - \mathsf{polylog}(s) \cdot \nu^3\ell|\mathbb{F}|\epsilon - 2^{-\nu+1}$, for at least $k - r'$ of the lines $L \in \{L^{N'}_1, \ldots, L^{N'}_k\}$, we have that $A^{N'\to x_{N'}} \circ L : \mathbb{F} \to \mathbb{F}$ is a univariate polynomial of degree $< m|H|$ (where the probability is over $Z, \{L^i_j\}_{i\in B, j\in[k]}, A$).

Hence, by Claim 2.33.3, with probability $\geq 1 - t \cdot 2^d \cdot \mathsf{polylog}(s)\big(\nu^4\ell|\mathbb{F}|\epsilon + 2^{-\nu/2}\big)$, for at least $k - r'$ of the lines $L \in \{L^{N'}_1, \ldots, L^{N'}_k\}$, we have that $A^{N'\to x_{N'}} \circ L : \mathbb{F} \to \mathbb{F}$ is a univariate polynomial of degree $< m|H|$ (where the probability is over $Z, \{L^i_j\}_{i\in B, j\in[k]}, A$). The lemma follows by Claim 2.5.1. □

Proof of Lemma 2.31

The following proof is similar to the proof of Lemma 2.5 (in Section 2.7.7) but differs in the actual parameters, and in the use of Lemma 2.45 (rather than Lemma 2.29).

Proof. Consider the point $N' \in [N']$, viewed as a point in $H^m \subset D_X$. Recall that the formula φ' contains a clause $(w_{N'} = 1) \vee (w_{N'} = 1) \vee (w_{N'} = 1)$ that checks that $w_{N'} = 1$.

Let $L^1_1, \ldots, L^1_k, L^2_1, \ldots, L^2_k, L^3_1, \ldots, L^3_k : \mathbb{F} \to D_X$ be $3k$ random lines, such that for every line $L \in \{L^1_1, \ldots, L^1_k, L^2_1, \ldots, L^2_k, L^3_1, \ldots, L^3_k\}$, we have $L(0) = N'$.

Let $S = \{L^1_j(t), L^2_j(t), L^3_j(t)\}_{j\in[k], t\in\mathbb{F}} \subset D_X$. Let $A \in_R \mathcal{A}_S$.

For any $v \in \mathbb{F}$, define $A^{N'\to v} : S \to \mathbb{F}$ by $A^{N'\to v}(i') = A(i')$ for $i' \neq N'$ and $A^{N'\to v}(N') = v$.

By Lemma 2.22, with probability $\geq 1 - 9\ell|\mathbb{F}|\epsilon - \delta$, there exist $v_1, v_2, v_3 \in \mathbb{F}$, such that, for at least $k - r'$ of the indices $j \in [k]$, the following is satisfied
(where the probability is over $L^1_1, \ldots, L^1_k, L^2_1, \ldots, L^2_k, L^3_1, \ldots, L^3_k, A$):

1. $A^{N'\to v_1} \circ L^1_j : \mathbb{F} \to \mathbb{F}$ is a univariate polynomial of degree $< m|H|$.
2. $A^{N'\to v_2} \circ L^2_j : \mathbb{F} \to \mathbb{F}$ is a univariate polynomial of degree $< m|H|$.
3. $A^{N'\to v_3} \circ L^3_j : \mathbb{F} \to \mathbb{F}$ is a univariate polynomial of degree $< m|H|$.
4. $(v_1 - 1) \cdot (v_2 - 1) \cdot (v_3 - 1) = 0$.

On the other hand, by (three applications of) Lemma 2.45 and Claim 2.5.1:

1. With probability $\geq 1 - t \cdot 2^d \cdot \mathsf{polylog}(s)\big(\nu^4\ell|\mathbb{F}|\epsilon + 2^{-\nu/2}\big) - \delta$, for at least $k - r'$ of the lines $L \in \{L^1_1, \ldots, L^1_k\}$, we have that $A^{N'\to x_{N'}} \circ L : \mathbb{F} \to \mathbb{F}$ is a univariate polynomial of degree $< m|H|$.

2. With probability $\geq 1 - t \cdot 2^d \cdot \mathsf{polylog}(s)\big(\nu^4 \ell |\mathbb{F}| \epsilon + 2^{-\nu/2}\big) - \delta$, for at least $k - r'$ of the lines $L \in \{L_1^2, \ldots, L_k^2\}$, we have that $A^{N' \to x_{N'}} \circ L : \mathbb{F} \to \mathbb{F}$ is a univariate polynomial of degree $< m|H|$.

3. With probability $\geq 1 - t \cdot 2^d \cdot \mathsf{polylog}(s)\big(\nu^4 \ell |\mathbb{F}| \epsilon + 2^{-\nu/2}\big) - \delta$, for at least $k - r'$ of the lines $L \in \{L_1^3, \ldots, L_k^3\}$, we have that $A^{N' \to x_{N'}} \circ L : \mathbb{F} \to \mathbb{F}$ is a univariate polynomial of degree $< m|H|$.

Thus, by the union bound, with probability $\geq 1 - t \cdot 2^d \cdot \mathsf{polylog}(s)\big(\nu^4 \ell |\mathbb{F}| \epsilon + 2^{-\nu/2}\big) > 0$, there exist $v_1, v_2, v_3 \in \mathbb{F}$, such that, $(v_1 - 1) \cdot (v_2 - 1) \cdot (v_3 - 1) = 0$, and

1. For at least $k - r'$ of the lines $L \in \{L_1^1, \ldots, L_k^1\}$, we have that $A^{N' \to v_1} \circ L : \mathbb{F} \to \mathbb{F}$ is a univariate polynomial of degree $< m|H|$.

2. For at least $k - r'$ of the lines $L \in \{L_1^2, \ldots, L_k^2\}$, we have that $A^{N' \to v_2} \circ L : \mathbb{F} \to \mathbb{F}$ is a univariate polynomial of degree $< m|H|$.

3. For at least $k - r'$ of the lines $L \in \{L_1^3, \ldots, L_k^3\}$, we have that $A^{N' \to v_3} \circ L : \mathbb{F} \to \mathbb{F}$ is a univariate polynomial of degree $< m|H|$.

4. For at least $k - r'$ of the lines $L \in \{L_1^1, \ldots, L_k^1\}$, we have that $A^{N' \to x_{N'}} \circ L : \mathbb{F} \to \mathbb{F}$ is a univariate polynomial of degree $< m|H|$.

5. For at least $k - r'$ of the lines $L \in \{L_1^2, \ldots, L_k^2\}$, we have that $A^{N' \to x_{N'}} \circ L : \mathbb{F} \to \mathbb{F}$ is a univariate polynomial of degree $< m|H|$.

6. For at least $k - r'$ of the lines $L \in \{L_1^3, \ldots, L_k^3\}$, we have that $A^{N' \to x_{N'}} \circ L : \mathbb{F} \to \mathbb{F}$ is a univariate polynomial of degree $< m|H|$.

Since, $r' + r' < k$, this implies that $x_{N'} = v_1 = v_2 = v_3$, and hence $x_{N'} = 1$. Since by our setting of parameters

$$1 - t \cdot 2^d \cdot \mathsf{polylog}(s)\big(\nu^4 \ell |\mathbb{F}| \epsilon + 2^{-\nu/2}\big) > 0,$$

the original input x is in the language $\mathcal{L}$. □

2.11 Soundness of V in the Augmented PCP

This section is similar to Section 2.8, but with respect to the augmented PCP.

Recall that $k \leq \mathsf{poly}(n)$, such that $4|\mathbb{F}|^4 \leq k \leq N'$, is the security parameter of the PCP, and that $1 \leq r < k$ is the parameter of the relaxed verifier V'. Recall that ℓ and $|\mathbb{F}|$ are bounded by $\mathsf{polylog}(N')$.

Lemma 2.46. *For a security parameter $k \leq \mathsf{poly}(n)$, such that $4|\mathbb{F}|^4 \leq k \leq N'$, fix the following parameters: Let $r = \frac{k}{40\ell|\mathbb{F}|}$. Let $\epsilon = 2^{-r/2}$. Let $k_{max} = k \cdot \mathsf{polylog}(s) \cdot \log(t)|\mathbb{F}| + 12k\ell|\mathbb{F}|^2$. Let $\delta = \frac{1}{|\mathbb{F}|^{8k\ell|\mathbb{F}|^2}}$. Then, V has soundness ϵ against (k_{max}, δ)-no-signaling strategies.*

The proof of Lemma 2.46 is similar to the proof of Lemma 2.30, but based on Lemma 2.31 (rather than Lemma 2.5).

Proof. Assume for a contradiction that V doesn't have soundness ϵ against (k_{max}, δ)-no-signaling strategies. By Lemma 2.3, since $\delta < \frac{\epsilon}{8 \cdot |\mathbb{F}|^{6k\ell|\mathbb{F}|^2}}$, we know that V' (with parameter r) doesn't have soundness $1-\epsilon'$ against (k'_{max}, δ')-no-signaling strategies, where $k'_{max} = k_{max} - 6k\ell|\mathbb{F}|^2 = k \cdot \mathsf{polylog}(s) \cdot \log(t)|\mathbb{F}| + 6k\ell|\mathbb{F}|^2$, and $\delta' = 8\delta|\mathbb{F}|^{6k\ell|\mathbb{F}|^2}/\epsilon < \frac{1}{|\mathbb{F}|^{k\ell|\mathbb{F}|^2}}$, and $\epsilon' = (10\ell|\mathbb{F}|2^{-r} + 2\delta)/\epsilon < \frac{1}{100N'\ell|\mathbb{F}|}$.

Hence V' (with parameter r) doesn't have soundness $1 - \epsilon'$ against (k'_{max}, δ')-no-signaling strategies, where $k'_{max} = k \cdot \mathsf{polylog}(s) \cdot \log(t)|\mathbb{F}| + 6k\ell|\mathbb{F}|^2$, and $\delta' = \frac{1}{|\mathbb{F}|^{k\ell|\mathbb{F}|^2}}$, and $\epsilon' = \frac{1}{100N'\ell|\mathbb{F}|}$.

This contradicts Lemma 2.31. □

2.12 From No-Signaling PCP to No-Signaling MIP

In this section we show how to transform a PCP that has soundness against (k_{max}, δ)-no-signaling strategies into an analogous MIP that uses k_{max} provers and has soundness against δ-no-signaling strategies.

Recall that a PCP (resp., MIP) relative to an oracle $\phi_n : \{0,1\}^{n'} \to \{0,1\}^{n''}$ is a PCP (resp., MIP) in which the verifier has oracle access to the function ϕ_n (see Section 2.4).

Lemma 2.47. *Let $\mathcal{L}$ be a language and suppose that $\mathcal{L}$ has a PCP with soundness ϵ against (k_{max}, δ)-no-signaling strategies relative to an oracle $\{\phi_n : \{0,1\}^{n'} \to \{0,1\}^{n''}\}_n$ (where n is the input length). Let D be the query alphabet, Σ be the answer alphabet, $k \leq k_{max}$ be the number of PCP queries and ℓ be the number of oracle queries. Then, $\mathcal{L}$ has an MIP relative to the same oracle $\{\phi_n\}$ with soundness ϵ against δ-no-signaling strategies. The MIP uses k_{max} provers, query alphabet D, answer alphabet Σ and ℓ oracle queries.*

Furthermore, if the running time of the PCP verifier is T_V then the running time of the MIP verifier is $O\left(T_V + k_{max} \cdot (\log|D| + \log|\Sigma|)\right)$ and if the PCP proof can be generated in time T_P then the running time of each of the MIP provers is $O(T_P)$.

Proof. Let V be the PCP verifier for $\mathcal{L}$ and let G_P be an algorithm that on input $x \in \mathcal{L}$ generates the PCP proof $P = G_P(x)$. We use V and G_P to construct an MIP for $\mathcal{L}$ (relative to the oracle $\{\phi_n\}$) that is sound against δ-no-signaling strategies.

We think of the PCP verifier V as being composed of two algorithms V_1 and V_2. The first algorithm, V_1, on input x of length n and a random string r generates a set $Q \subset D$ of k queries to the PCP and a set $Q_\phi \subset \{0,1\}^{n'}$ of ℓ queries to the oracle. The second algorithm, V_2, given x, the same random string r, the k answers $A \in \Sigma^Q$ (of the PCP) and oracle answers $A_\phi \in (\{0,1\}^{n''})^{Q_\phi}$, decides whether to accept the proof. We also assume (without loss of generality) that the algorithm G_P is deterministic. (Since completeness holds with probability 1, we can de-randomize G_P by fixing its random string arbitrarily.)

We first describe the k_{max} (honest) MIP provers' strategies and then proceed to describe the MIP verifier's strategy. Given an input $x \in \mathcal{L}$, each MIP prover (individually) computes the (deterministic) PCP proof $P = G_P(x)$ and given a query $q \in D$ just answers with $P(q)$.

The MIP verifier, on input x and a random string r, first runs $V_1(x, r)$ to obtain a set of k PCP queries $Q = \{q_1, \ldots, q_k\}$ and ℓ oracle queries Q_ϕ. The set Q is then used to construct a sequence $w \in D^{k_{max}}$ of k_{max} queries as follows. Initially, every entry of w is set to an arbitrary fixed value $z \in D$. Then, the verifier embeds the k queries of Q at random in w (which is of length $k_{max} \geq k$). Formally, for every set $Q \subset D$, every 1-to-1 function $\pi : Q \to [k_{max}]$ and every subset $S \subseteq Q$, let $w_{S,\pi} \in D^{k_{max}}$ be defined as follows. For every $i \in [k_{max}]$, if there exists $q \in S$ such that $i = \pi(q)$ then $(w_{S,\pi})_i = q$ and otherwise $(w_{S,\pi})_i = z$. The verifier chooses at random a 1-to-1 function $\pi : Q \to [k_{max}]$ and sets $w = w_{Q,\pi}$. The verifier then sends w to the k_{max} provers where prover i gets w_i. Simultaneously, the verifier queries the oracle ϕ_n at the points Q_ϕ.

Once the k_{max} provers respond with their answers $\alpha \in \Sigma^{k_{max}}$ (where the answer of the i^{th} prover is α_i) and the oracle responds with A_ϕ, the MIP verifier constructs $A \in \Sigma^Q$ by setting $A_q = \alpha_{\pi(q)}$ for every $q \in Q$. Formally, for every $S \subseteq Q$, let $T_{S,\pi} : \Sigma^{k_{max}} \to \Sigma^S$ be defined as $(T_{S,\pi}(\alpha))_q = \alpha_{\pi(q)}$ for every $q \in S$. The verifier sets $A = T_{Q,\pi}(\alpha)$ and outputs the result of V_2 on input (x, r, A, A_ϕ).

To see that (perfect) completeness holds, observe that the honest MIP provers (that get queries in Q) answer according to the PCP. Likewise, the oracle queries and answers are also exactly as in the PCP and therefore if $x \in \mathcal{L}$ then V_2 accepts and we obtain perfect completeness. We proceed to show that soundness holds against δ-no-signaling strategies.

Suppose that for some $x \notin \mathcal{L}$, there exists a δ-no-signaling family of distributions $\mathcal{A} = \{\mathcal{A}_u\}_{u \in D^{k_{max}}}$ that makes the MIP verifier accept with probability at least ϵ. By the construction of the MIP system this implies that:

$$\Pr_{\alpha \in_R \mathcal{A}_w, r, \pi} [V_2(x, r, A, A_\phi) = 1] \geq \epsilon \tag{2.4}$$

where (Q, Q_ϕ) is the output of $V_1(x, r)$, the function $\pi : Q \to [k_{max}]$ is the random 1-to-1 function, $w = w_{Q,\pi}$, $A = T_{Q,\pi}(\alpha)$, and A_ϕ are the answers of the oracle ϕ_n on the points Q_ϕ.

We use $\mathcal{A}$ to construct a family of distributions $\mathcal{B} = \{\mathcal{B}_Q\}_{Q \subset D, |Q| \leq k_{max}}$ that violates the (k_{max}, δ)-no-signaling soundness of the PCP. For every set Q of size at most k_{max}, the distribution $\mathcal{B}_Q$ is defined by first sampling a random 1-to-1 function $\pi : Q \to [k_{max}]$, setting $w = w_{Q,\pi}$, then sampling α from $\mathcal{A}_w$ and outputting $T_{Q,\pi}(\alpha)$. Note that for every $b \in \Sigma^Q$ it holds that

$$\Pr_{\beta \in_R \mathcal{B}_Q} [\beta = b] = \mathop{\mathbf{E}}_{\pi} \left[\Pr_{\alpha \in_R \mathcal{A}_w} [A = b] \right],$$

where $\pi : Q \to [k_{max}]$ is a random 1-1 function, $w = w_{Q,\pi}$ and $A = T_{Q,\pi}(\alpha)$.

We first show that the family of distributions $\mathcal{B} = \{\mathcal{B}_Q\}_{Q \subset D, |Q| \leq k_{max}}$ fools the PCP verifier into accepting with probability at least ϵ, and then proceed to show that $\mathcal{B}$ is

δ-no-signaling. Indeed, by the definition of $\mathcal{B}$,

$$\Pr_{\beta \in_R \mathcal{B}_Q, r}[V_2(x, r, \beta, A_\phi) = 1] = \Pr_{\alpha \in_R \mathcal{A}_w, r, \pi}[V_2(x, r, A, A_\phi) = 1]$$

where (Q, Q_ϕ) is the output of $V_1(x, r)$, the function $\pi : Q \to [k_{max}]$ is the random 1-to-1 function, $w = w_{Q,\pi}$, $A = T_{Q,\pi}(\alpha)$, and A_ϕ are the answers of the oracle ϕ_n on the points Q_ϕ. Thus, by Eq. (2.4), the PCP verifier accepts $x \notin \mathcal{L}$ with probability at least ϵ.

The next claim shows that $\mathcal{B}$ is δ-no-signaling and therefore we have a contradiction to our assumption that the PCP has ϵ-soundness against (k_{max}, δ)-no-signaling strategies.

Claim 2.47.1. *The family of distributions $\mathcal{B} = \{\mathcal{B}_Q\}_{Q \subset D, |Q| \leq k_{max}}$ is δ-no-signaling.*

Proof. To show that $\mathcal{B}$ is δ-no-signaling we need to show that for every $Q \subset D$ of size at most k_{max} and every $S \subset Q$ it holds that

$$\frac{1}{2} \sum_{b \in \Sigma^S} \left| \Pr_{\beta \in_R \mathcal{B}_S}[\beta = b] - \Pr_{\beta \in_R \mathcal{B}_Q}[\beta_S = b] \right| \leq \delta.$$

For every $b \in \Sigma^S$ it holds that

$$\Pr_{\beta \in_R \mathcal{B}_S}[\beta = b] = \mathop{\mathbf{E}}_{\pi'} \left[\Pr_{\alpha' \in_R \mathcal{A}_{w'}}[T_{S,\pi'}(\alpha') = b] \right] = \mathop{\mathbf{E}}_{\pi} \left[\Pr_{\alpha \in_R \mathcal{A}_w}[T_{S,\pi}(\alpha) = b] \right] = \mathop{\mathbf{E}}_{\pi} \left[\Pr_{\alpha \in_R \mathcal{A}_w}[(T_{Q,\pi}(\alpha))_S = b] \right] \tag{2.5}$$

where $\pi' : S \to [k_{max}]$ and $\pi : Q \to [k_{max}]$ are random 1-to-1 functions, $w' = w_{S,\pi'}$ and $w = w_{S,\pi}$, the second equality follows from the fact that π, restricted to S, is distributed identically to π', and the last equality follows from the fact that $T_{S,\pi}(\alpha) = (T_{Q,\pi}(\alpha))_S$.

On the other hand, using elementary operations and linearity of expectation, for every $b \in \Sigma^S$ it holds that

$$\begin{aligned}
\Pr_{\beta \in_R \mathcal{B}_Q}[\beta_S = b] &= \sum_{b' \in \Sigma^Q \text{ s.t. } b'_S = b} \Pr_{\beta \in_R \mathcal{B}_Q}[\beta = b'] \\
&= \sum_{b' \in \Sigma^Q \text{ s.t. } b'_S = b} \mathop{\mathbf{E}}_{\pi} \left[\Pr_{\alpha \in_R \mathcal{A}_{w''}}[T_{Q,\pi}(\alpha) = b'] \right] \\
&= \mathop{\mathbf{E}}_{\pi} \left[\sum_{b' \in \Sigma^Q \text{ s.t. } b'_S = b} \Pr_{\alpha \in_R \mathcal{A}_{w''}}[T_{Q,\pi}(\alpha) = b'] \right] \\
&= \mathop{\mathbf{E}}_{\pi} \left[\Pr_{\alpha \in_R \mathcal{A}_{w''}}[(T_{Q,\pi}(\alpha))_S = b] \right],
\end{aligned} \tag{2.6}$$

where $\pi : Q \to [k_{max}]$ is a random 1-1 function and $w'' = w_{Q,\pi}$. Using Eq. (2.5) and

Eq. (2.6), we obtain that:

$$\begin{aligned}
\sum_{b\in\Sigma^S}\left|\Pr_{\beta\in_R\mathcal{B}_S}[\beta=b]-\Pr_{\beta\in_R\mathcal{B}_Q}[\beta_S=b]\right| &= \sum_{b\in\Sigma^S}\left|\mathop{\mathbf{E}}_{\pi}\left[\Pr_{\alpha\in_R\mathcal{A}_w}[(T_{Q,\pi}(\alpha))_S=b]-\Pr_{\alpha''\in_R\mathcal{A}_{w''}}[(T_{Q,\pi}(\alpha''))_S=b]\right]\right| \\
&\leq \mathop{\mathbf{E}}_{\pi}\left[\sum_{b\in\Sigma^S}\left|\Pr_{\alpha\in_R A_w}[(T_{Q,\pi}(\alpha))_S=b]-\Pr_{\alpha''\in_R\mathcal{A}_{w''}}[(T_{Q,\pi}(\alpha''))_S=b]\right|\right] \\
&= \mathop{\mathbf{E}}_{\pi}\left[\sum_{b\in\Sigma^S}\left|\Pr_{\alpha\in_R A_w}[\alpha_{\pi(S)}=b]-\Pr_{\alpha''\in_R\mathcal{A}_{w''}}[\alpha''_{\pi(S)}=b]\right|\right] \\
&\leq 2\delta,
\end{aligned}$$

where $\pi : Q \to [k_{max}]$ is a random 1-1 function, $w = w_{S,\pi}$, $w'' = w_{Q,\pi}$ and the last inequality follows from the fact that $w_{\pi(S)} = w''_{\pi(S)}$ and our assumption that $\mathcal{A}$ is δ-no-signaling. Thus, $\mathcal{B}$ is δ-no-signaling. This concludes the proof of Claim 2.47.1 □

This concludes the proof of Lemma 2.47 □

2.13 A No-Signaling MIP for PSPACE with an Inefficient Prover

In this section we construct MIP protocols that have no-signaling soundness for languages that can be computed in bounded space. The protocol's (honest) provers are inefficient and run in time exponential in the space bound. This protocol will prove useful in Section 2.14 where we apply it to logspace computations (so that the provers run in polynomial time). We note that a similar result was obtained both by [KR09] and (independently) by [IKM09].

As a first step we show how to construct MIPs with no-signaling soundness for languages in IP (Lemma 2.48). We later use (a strong version, due to [GKR08], of) the IP = PSPACE [LFKN92, Sha92] theorem to obtain the required result (Lemma 2.49).

Lemma 2.48. *If a language $\mathcal{L}$ has an ℓ-round public-coin interactive proof-system with soundness ϵ (and perfect completeness), then for every $\delta \geq 0$, the language $\mathcal{L}$ has a 1-round ℓ-prover* MIP *with soundness $\epsilon + \delta\ell$ against δ-no-signaling strategies. If Λ is the length of the longest message in the interactive proof then the* MIP *has query and answer alphabet $\{0,1\}^{\ell\cdot\Lambda}$.*

Furthermore, if the running time of the interactive-proof verifier is T_V then the running time of the MIP *verifier is $O(\ell \cdot T_V)$. If the running time of the interactive-proof prover is T_P then the running time of each* MIP *prover is $O(\ell \cdot T_P)$.*

Proof. Let $\delta \geq 0$ and let (P, V) be an ℓ-round public-coin interactive proof for a language $\mathcal{L}$. Let Λ be the length of the longest message in the interactive proof. Let m_i denote the message sent from the verifier to the prover in the i^{th} round and let b_i denote the

prover's response to m_i. Since the protocol is public-coin, we assume that the messages $m_1, \ldots, m_\ell$ are generated by the verifier in the beginning of the protocol and in particular, they do not depend on the prover's answers. We also assume without loss of generality that the *honest* prover's response b_i to the i^{th} message m_i depends only on m_i and x (and not on $m_1, \ldots, m_{i-1}$).[16] We construct a 1-round MIP $(V', P'_1, \ldots, P'_\ell)$ with δ no-signaling soundness for $\mathcal{L}$ as follows.

The verifier V' generates the ℓ messages $m_1, \ldots, m_\ell$ and for every $i \in [\ell]$, it sends m_i to the prover P'_i. The prover P'_i answers the query m_i by b_i which is computed by the next message function of P at round i and with respect to m_i and x. To decide whether to accept, the verifier V' simply runs $V(x, m_1, \ldots, m_\ell, b_1, \ldots, b_\ell)$.

To show that (perfect) completeness holds, observe that for $x \in L$ the probability that V' outputs 1 after interacting with $P'_1, \ldots, P'_\ell$ equals the probability that V outputs 1 after interacting with P. We proceed to prove that no-signaling soundness holds.

Suppose toward a contradiction that there exists a δ-no-signaling cheating strategy $\{\mathcal{A}_q\}_{q \in (\{0,1\}^\Lambda)^\ell}$ that breaks the soundness of V' with probability $\epsilon + \delta\ell$. We use the latter to construct a cheating prover P^* for the interactive proof that breaks soundness with probability at least ϵ (contradicting our assumption on the soundness of V).

The cheating prover P^* is defined as follows. Given V's first message m_1, the cheating prover selects at random $(b_1^{(1)}, \ldots, b_\ell^{(1)}) \in_R \mathcal{A}_{m_1,*,\ldots,*}$, where $*$ denotes an arbitrary fixed string (e.g., the string 0^Λ). It saves only $b_1 \stackrel{\text{def}}{=} b_1^{(1)}$ and sends b_1 to the verifier. The verifier answers with m_2. After receiving m_2, the prover selects $(b_1^{(2)}, \ldots, b_\ell^{(2)}) \in_R \mathcal{A}_{m_1,m_2,*,\ldots,*}$ conditioned on $b_1^{(2)} = b_1$. It saves only $b_2 \stackrel{\text{def}}{=} b_2^{(2)}$ and sends b_2 to the verifier. Generally, after getting the i^{th} message m_i, the prover selects $(b_1^{(i)}, \ldots, b_\ell^{(i)}) \in_R \mathcal{A}_{m_1,\ldots,m_i,*,\ldots,*}$ conditioned on $b_1^{(i)}, \ldots, b_{i-1}^{(i)} = b_1, \ldots, b_{i-1}$ and sends $b_i \stackrel{\text{def}}{=} b_i^{(i)}$ to the prover.

Before proceeding we note that in the above process it might happen that the conditional probability space is empty. In such a case the prover P^* just sends a special symbol $\perp$.

We show that for every ℓ messages $m_1, \ldots, m_\ell$, the distribution of the answers $b_1, \ldots, b_\ell$ described above is $\delta\ell$-close to the distribution $\mathcal{A}_{m_1,\ldots,m_\ell}$. This follows from the following claim by setting $i = \ell$.

Claim 2.48.1. *Fix ℓ messages $m_1, \ldots, m_\ell \in (\{0,1\}^\Lambda)^\ell$. Let B_i denote the distribution of the first i elements in $\mathcal{A}_{m_1,\ldots,m_i,*,\ldots,*}$. Then, for every $0 \leq i \leq \ell$, the distribution $(b_1, \ldots, b_i)$ is δi-close to B_i.*

Proof. We prove the claim by induction. The base case $i = 0$ is trivial and so we proceed to the inductive step. Suppose that the claim holds for some i. For every $\beta_1, \ldots, \beta_i \in \{0,1\}^\Lambda$, consider the random variable $X_{i+1}(\beta_1, \ldots, \beta_i)$ defined by the following random process: select $(z_1, \ldots, z_{i+1})$ according to the distribution B_{i+1} conditioned on $z_1, \ldots, z_i = \beta_1, \ldots, \beta_i$ and output z_{i+1}. As before, if the conditional probability space is empty then output $\perp$.

[16] This can be easily achieved by having the verifier resend its previous messages at every round. Note that this increases the length of each message by a factor of ℓ.

Note that by the definition of P^*, the message b_{i+1} is distributed exactly as $X_{i+1}(b_1, \ldots, b_i)$. Therefore, by the inductive hypothesis, the distributions

- $b_1, \ldots, b_{i+1}$; and
- $B_i, X_{i+1}(B_i)$

are δi-close. Since $\mathcal{A}$ is δ-no-signaling, the distribution B_i is δ-close to the distribution obtained by taking the i first elements of B_{i+1}. Therefore, the distributions

- $B_i, X_{i+1}(B_i)$; and
- B_{i+1}

are δ-close. Thus, $(b_1, \ldots, b_{i+1})$ and B_{i+1} are $\delta(i+1)$-close. This completes the proof of Claim 2.48.1. □

By our assumption, the soundness of V' is violated with probability $\epsilon + \delta\ell$ when the answers that it receives are distributed according to $\mathcal{A}_{m_1,\ldots,m_\ell}$. Therefore, by Claim 2.48.1, the soundness of V is violated with probability at least ϵ when it receives the answers $b_1, \ldots, b_\ell$, in contradiction to our assumption on the soundness of V. □

Using Lemma 2.48, we can prove the following useful lemma.

Lemma 2.49. *If $\mathcal{L}$ can be computed by a Turing machine in space $s \stackrel{\text{def}}{=} s(n) \geq n$ (where n is the input length) then, for every $t \geq 1$, the language $\mathcal{L}$ has a $\mathsf{poly}(s)$-prover MIP with soundness $\mathsf{poly}(s) \cdot 2^{-t}$ against 2^{-t}-no-signaling strategies. The query and answer alphabets are $\{0,1\}^{t \cdot \mathsf{poly}(s)}$.*

Furthermore, the verifier runs in time $t \cdot \mathsf{poly}(s)$ and the (honest) provers run in time $t \cdot \mathsf{poly}(2^s)$.

Proof. Goldwasser *et al.* [GKR08] (see also [Rot09, Corollary 3.4.8]) show that if $\mathcal{L}$ can be computed in space s, then $\mathcal{L}$ has a $\mathsf{poly}(s)$-round public-coin interactive proof with soundness error $\frac{1}{2}$. The verifier's running time is $\mathsf{poly}(s)$ and the prover's running time is $\mathsf{poly}(2^s)$. The length of each message is $\mathsf{poly}(s)$.[17]

To this base protocol we apply a $O(t)$-fold parallel repetition (see, e.g., [Gol08, Exercise 9.1] or [Gol99, Appendix C.1]) which produces a $\mathsf{poly}(s)$-round public-coin interactive proof with soundness error 2^{-2t}. The verifier's running time is $t \cdot \mathsf{poly}(s)$ and the prover's running time is $t \cdot \mathsf{poly}(2^s)$. The length of each message is $t \cdot \mathsf{poly}(s)$. The lemma follows by applying Lemma 2.48 with $\delta = 2^{-t}$. □

[17] Indeed, the advantage in using the [GKR08] protocol is that the running time of the prover is $\mathsf{poly}(2^s)$ rather than $2^{\mathsf{poly}(s)}$ as in the classical [LFKN92, Sha92] protocol.

2.14 Simulating an MIP Oracle

In this section we show that if a language $\mathcal{L}$ has an MIP with soundness against no-signaling strategies relative to an oracle $\{\phi_n\}$ and the function $\{\phi_n\}$ can be computed by a Turing machine that uses only a small amount of space, then the oracle can essentially be simulated and $\mathcal{L}$ has an MIP with soundness against no-signaling strategies *without* an oracle.

Lemma 2.50. *Let $\mathcal{L}$ be a language and suppose that $\mathcal{L}$ has an* MIP *relative to an oracle $\{\phi_n : \{0,1\}^{n'} \to \{0,1\}^{n''}\}_n$ (where n is the input length) with soundness error ϵ against δ-no-signaling strategies. Let k be the number of provers and ℓ be the number of oracle queries used by the* MIP*. Suppose further that the function $\{\phi_n\}$ can be computed by a Turing machine in linear space (i.e., in space $O(n')$). Then, for every $t \geq 1$, the language $\mathcal{L}$ has an* MIP *protocol* without an oracle *that has soundness $\epsilon + \ell \cdot \mathsf{poly}(n^*) \cdot 2^{-t}$, where $n^* = n' + \log(n'')$, against $\min(\delta, 2^{-t})$-no-signaling strategies. The resulting* MIP *uses $k + \ell \cdot \mathsf{poly}(n^*)$ provers.*

Furthermore, if the original MIP *verifier runs in time T_V then the resulting verifier runs in time $T_V + O(\ell \cdot t \cdot \mathsf{poly}(n^*))$. If the original* MIP *provers run in time T_P then the resulting provers run in time $T_P + O(\ell \cdot t \cdot \mathsf{poly}(2^{n^*}))$. If the original* MIP *has query alphabet D and answer alphabet Σ then the resulting* MIP *has query alphabet $D \cup \{0,1\}^{t \cdot \mathsf{poly}(n^*)}$ and answer alphabet $\Sigma \cup \{0,1\}^{t \cdot \mathsf{poly}(n^*)}$.*

The high level approach is to use Lemma 2.49 to transform each oracle query into an additional MIP with no-signaling soundness and to show that composing these MIP protocols maintains the no-signaling soundness. The rest of this section is devoted to the (straightforward and somewhat tedious) proof of Lemma 2.50.

Simulating a single query. To prove Lemma 2.50, we first show that if the oracle can be computed by an MIP protocol with soundness against no-signaling strategies, then the oracle queries can be removed one by one (Lemma 2.51). For simplicity and since it suffices for our purposes, in the following we replace the oracle $\phi_n : \{0,1\}^{n'} \to \{0,1\}^{n''}$ with an equivalent oracle $\phi'_n : \{0,1\}^{n^*} \to \{0,1\}$ that returns Boolean valued answers, where $n^* = n' + \log(n'')$. The oracle ϕ'_n on input $(z,i) \in \{0,1\}^{n'+\log(n'')}$ simply outputs the i-th bit of $\phi_n(z)$.

We note that the requirement in Lemma 2.51 will be that the oracle function $\{\phi'_n\}$ can be *computed*, rather than *decided*, by an MIP protocol (with soundness against no-signaling strategies). This means that both the language $\mathcal{L}_{\phi'} = \{z \in \{0,1\}^{n^*} : \phi'_n(z) = 1\}$ and the complement language $\overline{\mathcal{L}}_{\phi'}$ have MIP protocols with no-signaling soundness. However, it will be convenient for us to assume that there is a *single* protocol for computing $\{\phi'_n\}$ with no-signaling soundness, a notion that will be defined next. Indeed, as will be shown in Claim 2.51.3, the existence of MIP protocols with no-signaling soundness for both $\mathcal{L}_{\phi'}$ and $\overline{\mathcal{L}}_{\phi'}$ implies a single protocol for computing $\{\phi'_n\}$.

Multi-prover protocols for computing a function. In a one-round k-prover interactive protocol for *computing* a function $f : \{0,1\}^* \to \{0,1\}^*$, there are k computationally unbounded provers, $P_1, \ldots, P_k$, that try to convince a (probabilistic) polynomial-time verifier, V, of the value of $f(x)$ where the input $x \in \{0,1\}^*$ is known to all parties.

The interaction is similar to that in a one-round MIP (see Section 2.4.2). Given x and her random string, the verifier generates k queries, $q_1, \ldots, q_k$, one for each prover, and sends them to the k provers. The provers respond with answers $a_1, \ldots, a_k$. Finally, the verifier, based on the answers that she receives (as well as the input x and her random string), either outputs a value (which is supposed to equal $f(x)$) or outputs a special abort symbol $\perp$.

Denote by D the query alphabet and by Σ the answer alphabet. We say that $(V, P_1, \ldots, P_k)$ is a one-round k-prover protocol for computing f, with soundness ϵ against δ-no-signaling strategies, if the following two properties are satisfied:

1. **Completeness:** For every $x \in \{0,1\}^*$, the verifier V outputs $f(x)$ with probability 1, after interacting with $P_1, \ldots, P_k$.

2. **Soundness:** For every $x \in \{0,1\}^*$, and any δ-no-signaling family of distributions $\{\mathcal{A}_q\}_{q \in D^k}$ (where $\mathcal{A}_q$ is distributed over Σ^k, for every $q \in D^k$), with probability $\geq 1 - \epsilon$, the verifier V outputs either $f(x)$ or $\perp$, where on queries $q = (q_1, \ldots, q_k)$ the answers are given probabilistically by $(a_1, \ldots, a_k) \in_R \mathcal{A}_q$.

We are now ready to state and prove Lemma 2.51.

Lemma 2.51. *Let $\mathcal{L}$ be a language and suppose that $\mathcal{L}$ has an MIP relative to an oracle $\{\phi'_n : \{0,1\}^{n^*} \to \{0,1\}\}_n$ (where n is the input length) with soundness ϵ against δ-no-signaling strategies. Let k be the number of provers and $\ell > 0$ be the number of oracle queries used by the MIP. Suppose further that the function $\{\phi'_n\}$ can be computed by a one-round k'-prover protocol with soundness ϵ' against δ'-no-signaling strategies. Then, $\mathcal{L}$ has an MIP relative to the same oracle $\{\phi'_n\}$ with ϵ'' soundness against δ''-no-signaling strategies where $\epsilon'' = \epsilon + \epsilon' + \delta''$ and $\delta'' = \min(\delta, \delta')$. The resulting MIP uses $k + k'$ provers but only $\ell - 1$ oracle queries.*

Furthermore, if the original MIP verifier for $\mathcal{L}$ runs in time T_V, and the verifier of the MIP for $\{\phi'_n\}$ runs in time T'_V, then the resulting verifier runs in time $T_V + O(T'_V + k'n^*)$. If the provers of the MIP for $\mathcal{L}$ run in time T_P and the provers of the MIP for $\{\phi'_n\}$ run in time T'_P then the resulting provers run in time $T_P + O(T'_P + n^*)$. If the original MIP has query alphabet D and answer alphabet Σ and the oracle MIP has query alphabet D' and answer alphabet Σ' then the resulting MIP has query alphabet $D'' = D \cup (\{0,1\}^{n^*} \times D')$ and answer alphabet $\Sigma'' = \Sigma \cup \Sigma'$. [18]

The high level idea is that if the oracle can be computed by a no-signaling MIP protocol then an oracle query can just be simulated by adding sufficiently many provers and running the multi-prover protocol for the oracle function with the additional provers.

[18] Note that we do not assume that $D \cap (\{0,1\}^{n^*} \times D') = \emptyset$ nor that $\Sigma \cap \Sigma' = \emptyset$.

The no-signaling soundness property guarantees that revealing the oracle query to the provers does not harm soundness (too much).

Proof of Lemma 2.51. Let $(V, P_1, \ldots, P_k)$ be the MIP for $\mathcal{L}$ relative to the oracle $\{\phi'_n : \{0,1\}^{n^*} \to \{0,1\}\}$ that has soundness ϵ against δ-no-signaling strategies. Let D be the query alphabet and Σ the answer alphabet. Let $(V', P'_1, \ldots, P'_{k'})$ be the k'-prover interactive-protocol for computing $\{\phi'_n\}$ with soundness ϵ' against δ'-no-signaling strategies. Recall that this means that when interacting with the honest provers, $V'(z)$ outputs $\phi'_n(z)$ (with probability 1) and that no δ'-no-signaling cheating strategy can convince $V'(z)$ to output anything other than $\phi'_n(z)$ or $\perp$, with probability greater than ϵ'. Let D' be the query alphabet and Σ' the answer alphabet of $(V', P'_1, \ldots, P'_{k'})$. Let $\Sigma'' = \Sigma \cup \Sigma'$. It will be convenient for us to extend the answer alphabets of both protocols to Σ''.[19]

Since we deal with 1-round protocols, it will be convenient to think of each one of our verifiers as being composed of two algorithms (that share their randomness) - the query generation step and the verification step. Specifically, we think of V as being composed of two algorithms V_1 and V_2. The first algorithm, V_1, on input x and a random string r, outputs a sequence of k prover-queries $\mathbf{q} \in D^k$ and a sequence of ℓ oracle-queries $\mathbf{q}' \in (\{0,1\}^{n^*})^\ell$. The second algorithm, V_2, on input x, the same random string r, prover answers $\mathbf{a} \in (\Sigma'')^k$ and oracle answers $\mathbf{b} \in \{0,1\}^\ell$ outputs a bit representing whether $x \in \mathcal{L}$. Similarly, we think of V' as being composed of two algorithms V'_1 and V'_2. The first algorithm, V'_1, on input $q^* \in \{0,1\}^{n^*}$ and a random string s, outputs a sequence of k' queries $\mathbf{w} \in (D')^{k'}$. The second algorithm, V'_2, on input q^*, the same random string s, and answers $\mathbf{z} \in (\Sigma'')^{k'}$, outputs the result (which is supposed to be equal to $\phi'_n(q^*)$).

We construct an MIP protocol for $\mathcal{L}$ with only $\ell - 1$ oracle queries (but using $k + k'$ provers) as follows. The verifier V'' is composed of two steps, where V''_1 denotes the query generation step and V''_2 denotes the verification step. The first algorithm, V''_1, on input x and the random string (r, s), first invokes $V_1(x, r)$ to obtain k prover-queries $\mathbf{q} = (q_1, \ldots, q_k) \in D^k$ and ℓ oracle-queries $\mathbf{q}' = (q'_1, \ldots, q'_\ell) \in (\{0,1\}^{n^*})^\ell$. For every $i \in [k]$, the query q_i is sent directly to the i-th prover and the oracle queries $q'_2, \ldots, q'_\ell$ are sent directly to the oracle ϕ'_n. The query $q^* \stackrel{\text{def}}{=} q'_1$ is handled differently (to avoid an additional oracle query). The k' additional prover queries are generated by invoking $V'_1(q^*, s)$, to obtain a sequence of k' queries $\mathbf{w} = (w_1, \ldots, w_{k'}) \in (D')^{k'}$. For every $i \in [k']$ the query (q^*, w_i) is sent to the $(k+i)$-th prover.

We denote the query alphabet by $D'' \stackrel{\text{def}}{=} D \cup (\{0,1\}^{n^*} \times D')$. For every sequence $\omega \in (D')^{k'}$ we denote by $\overline{\omega}(q^*) \stackrel{\text{def}}{=} (q^*, \omega_1), \ldots, (q^*, \omega_{k'}) \in (D'')^{k'}$. Thus, the sequence of queries sent by the verifier is $(\mathbf{q}, \overline{\mathbf{w}}(q^*)) \in (D'')^{k+k'}$.

The honest provers operate as follows. The first k provers operate exactly the same as the provers $P_1, \ldots, P_k$ in the original MIP for $\mathcal{L}$. That is, for every $i \in [k]$, the i-th prover, on input x and query q_i, answers with $a_i = P_i(x, q_i)$. We denote $\mathbf{a} \stackrel{\text{def}}{=} (a_1, \ldots, a_k) \in (\Sigma'')^k$. The last k' provers answer their queries as follows. For every $i \in [k']$, the $(k+i)$-th

[19] This can be done by having the verifiers reject immediately if they see symbols that are not in their original alphabets.

prover, given the query (q^*, w_i), answers its query with $z_i = P'_i(q^*, w_i)$. We denote $\mathbf{z} \stackrel{\text{def}}{=} (z_1, \ldots, z_{k'}) \in (\Sigma'')^{k'}$.

To decide whether to accept, on input x, the random string (r, s), prover answers $(\mathbf{a}, \mathbf{z}) \in (\Sigma'')^{k+k'}$, and oracle answers $b_2, \ldots, b_\ell \in \{0, 1\}$, the algorithm V''_2 first recomputes q^* from x and r and computes $b^* \stackrel{\text{def}}{=} V'_2(q^*, s, \mathbf{z})$. If $b^* = \perp$, then V''_2 rejects. Otherwise, V''_2 outputs the result of $V_2(x, r, \mathbf{a}, \mathbf{b})$, where $\mathbf{b} = (b^*, b_2, \ldots, b_\ell)$. In other words, the verifier computes the result of the original verification process when the answers to $\mathbf{q}$ are $\mathbf{a}$, the answer to the first oracle query is b^* and the answers to the rest of the oracle queries $q'_2, \ldots, q'_\ell$ are (respectively) $b_2, \ldots, b_\ell$.

We first argue that the resulting MIP has perfect completeness and then proceed to prove soundness against δ''-no-signaling strategies. To show that completeness holds, observe that when the verifier V'' interacts with the honest provers on input $x \in \mathcal{L}$, since the protocol $(V', P'_1, \ldots, P'_{k'})$ has perfect completeness, it holds that $b^* = \phi'_n(q^*)$ and therefore V''_2 runs V_2 with the correct oracle answers. The completeness of the protocol follows from the completeness of $(V, P_1, \ldots, P_k)$.

To show that δ''-no-signaling soundness holds, assume for a contradiction that there exists some δ''-no-signaling cheating strategy $\{\mathcal{A}_{(\chi,\omega)}\}_{(\chi,\omega)\in(D'')^{k+k'}}$ that fools V'' into accepting $x \notin \mathcal{L}$ with probability ϵ''. That is,

$$\Pr_{\substack{r,s \\ (\mathbf{a},\mathbf{z})\in_R \mathcal{A}_{(\mathbf{q},\overline{\mathbf{w}}(q^*))}}} \left[V''_2(x, (r, s), (\mathbf{a}, \mathbf{z}), (b_2, \ldots, b_\ell)) = 1\right] \geq \epsilon''$$

where $\mathbf{q}, q^*, \mathbf{w}, b_2, \ldots, b_\ell$ are constructed as above. Using elementary manipulations we have that

$$\begin{aligned}
\epsilon'' &\leq \Pr_{\substack{r,s \\ (\mathbf{a},\mathbf{z})\in_R \mathcal{A}_{(\mathbf{q},\overline{\mathbf{w}}(q^*))}}} \left[V''_2(x, (r, s), (\mathbf{a}, \mathbf{z}), (b_2, \ldots, b_\ell)) = 1\right] \\
&= \Pr_{\substack{r,s \\ (\mathbf{a},\mathbf{z})\in_R \mathcal{A}_{(\mathbf{q},\overline{\mathbf{w}}(q^*))}}} \left[\left(V''_2(x, (r, s), (\mathbf{a}, \mathbf{z}), (b_2, \ldots, b_\ell)) = 1\right) \wedge \left(b^* \neq \phi'_n(q^*)\right)\right] + \\
&\quad \Pr_{\substack{r,s \\ (\mathbf{a},\mathbf{z})\in_R \mathcal{A}_{(\mathbf{q},\overline{\mathbf{w}}(q^*))}}} \left[\left(V''_2(x, (r, s), (\mathbf{a}, \mathbf{z}), (b_2, \ldots, b_\ell)) = 1\right) \wedge \left(b^* = \phi'_n(q^*)\right)\right] \\
&\leq \Pr_{\substack{r,s \\ (\mathbf{a},\mathbf{z})\in_R \mathcal{A}_{(\mathbf{q},\overline{\mathbf{w}}(q^*))}}} \left[b^* \notin \{\phi'_n(q^*), \perp\}\right] + \\
&\quad \Pr_{\substack{r,s \\ (\mathbf{a},\mathbf{z})\in_R \mathcal{A}_{(\mathbf{q},\overline{\mathbf{w}}(q^*))}}} \left[\left(V''_2(x, (r, s), (\mathbf{a}, \mathbf{z}), (b_2, \ldots, b_\ell)) = 1\right) \wedge \left(b^* = \phi'_n(q^*)\right)\right] \qquad (2.7)
\end{aligned}$$

where $\mathbf{q}, q^*, \mathbf{w}, b_2, \ldots, b_\ell$ are as above and $b^* = V'_2(q^*, s, \mathbf{z})$. Lemma 2.51 follows from the following two claims (Claim 2.51.1 and Claim 2.51.2), that analyze the last two terms separately.

Claim 2.51.1.

$$\Pr_{\substack{r,s \\ (\mathbf{a},\mathbf{z})\in_R \mathcal{A}_{(\mathbf{q},\overline{\mathbf{w}}(q^*))}}} \left[b^* \notin \{\phi'_n(q^*), \perp\}\right] < \epsilon'$$

Proof. Suppose otherwise. That is:

$$\Pr_{\substack{r,s \\ (\mathbf{a},\mathbf{z})\in_R \mathcal{A}_{(\mathbf{q},\overline{\mathbf{w}}(q^*))}}} \left[V_2'(q^*,s,\mathbf{z}) \notin \{\phi_n'(q^*),\perp\}\right] \geq \epsilon'.$$

Then, by an averaging argument, there exists a fixed value of r for which:

$$\Pr_{\substack{s \\ (\mathbf{a},\mathbf{z})\in_R \mathcal{A}_{(\mathbf{q},\overline{\mathbf{w}}(q^*))}}} \left[V_2'(q^*,s,\mathbf{z}) \notin \{\phi_n'(q^*),\perp\}\right] \geq \epsilon' \tag{2.8}$$

where $\mathbf{q}, q^*$ are fixed (based on the value of r), and $\mathbf{w} = V_1'(q^*,s)$. For the rest of the proof of Claim 2.51.1, we use $r, \mathbf{q}, q^*$ to refer to the foregoing fixed values.

We construct a δ'-no-signaling strategy $\mathcal{B} = \{\mathcal{B}_\omega\}_{\omega\in(D')^{k'}}$ that on input q^*, fools V' into outputting a value that is neither $\phi_n'(q^*)$ nor $\perp$, with probability $\geq \epsilon'$, contradicting our assumption on the soundness of $(V', P_1', \ldots, P_{k'}')$. For every $\omega \in (D')^{k'}$, the distribution $\mathcal{B}_\omega$ is defined by sampling $(\mathbf{a},\mathbf{z}) \in_R \mathcal{A}_{(\mathbf{q},\overline{\omega}(q^*))}$ and outputting $\mathbf{z}$.

To show that $\mathcal{B}$ violates the δ'-no-signaling soundness of $(V', P_1', \ldots, P_{k'}')$, note that by Eq. (2.8), the probability that V_2', on input q^*, the random string s and given answers $\mathbf{z} \in_R \mathcal{B}_{\mathbf{w}}$, where $\mathbf{w} = V_1'(q^*,s)$, outputs a value that is neither $\phi_n'(q^*)$ nor $\perp$ is at least ϵ'.

We proceed to show that $\mathcal{B}$ is δ'-no-signaling. Let $S \subset [k']$ and $\omega, \omega' \in (D')^{k'}$, such that $\omega_S = \omega'_S$. Suppose that the statistical distance between $\mathbf{z}_S$ and $\mathbf{z}'_S$ is more than δ, where $\mathbf{z} \in_R \mathcal{B}_\omega$ and $\mathbf{z}' \in_R \mathcal{B}_{\omega'}$. Hence,

$$\begin{aligned}
\delta' &< \frac{1}{2} \sum_{\beta\in(\Sigma'')^S} \left| \Pr_{\mathbf{z}\in_R\mathcal{B}_\omega}[\mathbf{z}_S = \beta] - \Pr_{\mathbf{z}'\in_R\mathcal{B}_{\omega'}}[\mathbf{z}'_S = \beta] \right| \\
&= \frac{1}{2} \sum_{\beta\in(\Sigma'')^S} \left| \Pr_{(\mathbf{a},\mathbf{z})\in_R\mathcal{A}_{(\mathbf{q},\overline{\omega}(q^*))}}[\mathbf{z}_S = \beta] - \Pr_{(\mathbf{a}',\mathbf{z}')\in_R\mathcal{A}_{(\mathbf{q},\overline{\omega'}(q^*))}}[\mathbf{z}'_S = \beta] \right|.
\end{aligned} \tag{2.9}$$

Let $S' = \{k+i : i \in S\}$. Then, by Eq. (2.9) the projections of the distributions $\mathcal{A}_{(\mathbf{q},\overline{\omega}(q^*))}$ and $\mathcal{A}_{(\mathbf{q},\overline{\omega'}(q^*))}$ to coordinates in S' are δ'-far. Since $(\mathbf{q},\overline{\omega}(q^*))_{S'} = (\mathbf{q},\overline{\omega'}(q^*))_{S'}$ and $\delta'' \leq \delta'$, this contradicts our assumption that $\mathcal{A}$ is δ''-no-signaling.

This concludes the proof of Claim 2.51.1. □

Claim 2.51.2.

$$\Pr_{\substack{r,s \\ (\mathbf{a},\mathbf{z})\in_R \mathcal{A}_{(\mathbf{q},\overline{\mathbf{w}}(q^*))}}} \left[\left(V_2''(x,(r,s),(\mathbf{a},\mathbf{z}),(b_2,\ldots,b_\ell)) = 1\right) \wedge \left(b^* = \phi_n'(q^*)\right)\right] < \epsilon + \delta''$$

Proof. Suppose otherwise. Thus, by the definition of V_2'',

$$\begin{aligned}
\epsilon + \delta'' &\leq \Pr_{\substack{r,s \\ (\mathbf{a},\mathbf{z})\in_R \mathcal{A}_{(\mathbf{q},\overline{\mathbf{w}}(q^*))}}} \left[\left(V_2''(x,(r,s),(\mathbf{a},\mathbf{z}),(b_2,\ldots,b_\ell)) = 1\right) \wedge \left(b^* = \phi_n'(q^*)\right) \right] \\
&= \Pr_{\substack{r,s \\ (\mathbf{a},\mathbf{z})\in_R \mathcal{A}_{(\mathbf{q},\overline{\mathbf{w}}(q^*))}}} \left[V_2''(x,(r,s),(\mathbf{a},\mathbf{z}),(b_2,\ldots,b_\ell)) = 1 \;\middle|\; b^* = \phi_n'(q^*) \right] \cdot \Pr_{\substack{r,s \\ (\mathbf{a},\mathbf{z})\in_R \mathcal{A}_{(\mathbf{q},\overline{\mathbf{w}}(q^*))}}} \left[b^* = \phi_n'(q^*) \right] \\
&= \Pr_{\substack{r,s, \\ (\mathbf{a},\mathbf{z})\in_R \mathcal{A}_{(\mathbf{q},\overline{\mathbf{w}}(q^*))}}} \left[V_2(x,r,\mathbf{a},\phi_n'(\mathbf{q}')) = 1 \;\middle|\; b^* = \phi_n'(q^*) \right] \cdot \Pr_{\substack{r,s \\ (\mathbf{a},\mathbf{z})\in_R \mathcal{A}_{(\mathbf{q},\overline{\mathbf{w}}(q^*))}}} \left[b^* = \phi_n'(q^*) \right] \\
&= \Pr_{\substack{r,s, \\ (\mathbf{a},\mathbf{z})\in_R \mathcal{A}_{(\mathbf{q},\overline{\mathbf{w}}(q^*))}}} \left[V_2(x,r,\mathbf{a},\phi_n'(\mathbf{q}')) = 1 \right] \qquad (2.10)
\end{aligned}$$

where $\mathbf{q}, \mathbf{q}', q^*, \mathbf{w}, b^*$ are as above, and $\phi_n'(\mathbf{q}') = (\phi_n'(q_1'), \ldots, \phi_n'(q_\ell'))$ (i.e., the correct oracle answers).

We argue that Eq. (2.10) contradicts the δ-no-signaling soundness of V. Toward this end, we construct a cheating strategy $\mathcal{B} = \{\mathcal{B}_\chi\}_{\chi\in D^k}$ that fools V into accepting $x \notin \mathcal{L}$, with probability $\geq \epsilon$. Let $\sigma \in (D'')^{k'}$ be an arbitrary fixed value. For every $\chi \in D^k$, the distribution $\mathcal{B}_\chi$ is defined by sampling $(\mathbf{a},\mathbf{z}) \in_R \mathcal{A}_{(\chi,\sigma)}$ and outputting $\mathbf{a}$.

We first show that $\mathcal{B}$ is δ-no-signaling and proceed to show that it violates the soundness of $(V, P_1, \ldots, P_k)$. Let $S \subset [k]$ and $\chi, \chi' \in D^k$ such that $\chi_S = \chi'_S$. Suppose toward a contradiction that the statistical distance between $\mathbf{a}_S$ and $\mathbf{a}'_S$ is more than δ, where $\mathbf{a} \in_R \mathcal{B}_\chi$ and $\mathbf{a}' \in_R \mathcal{B}_{\chi'}$. Hence,

$$\begin{aligned}
\delta &< \frac{1}{2} \sum_{\beta\in(\Sigma'')^S} \left| \Pr_{\mathbf{a}\in_R\mathcal{B}_\chi}[\mathbf{a}_S = \beta] - \Pr_{\mathbf{a}'\in_R\mathcal{B}_{\chi'}}[\mathbf{a}'_S = \beta] \right| \\
&= \frac{1}{2} \sum_{\beta\in(\Sigma'')^S} \left| \Pr_{(\mathbf{a},\mathbf{z})\in_R\mathcal{A}_{(\chi,\sigma)}}[\mathbf{a}_S = \beta] - \Pr_{(\mathbf{a}',\mathbf{z}')\in_R\mathcal{A}_{(\chi',\sigma)}}[\mathbf{a}'_S = \beta] \right| .
\end{aligned}$$

In particular, the projections of the distributions $\mathcal{A}_{(\chi,\sigma)}$ and $\mathcal{A}_{(\chi',\sigma)}$ to coordinates in S are δ-far. Since $(\chi,\sigma)_S = (\chi',\sigma)_S$ and $\delta'' \leq \delta$, this contradicts our assumption that $\mathcal{A}$ is δ''-no-signaling.

We proceed to show that $\{\mathcal{B}_\chi\}_{\chi\in D^k}$ fools V into accepting $x \notin \mathcal{L}$ with probability $\geq \epsilon$. Assume for a contradiction that

$$\Pr_{\substack{r, \\ \mathbf{a}'\in_R\mathcal{B}_\mathbf{q}}} [V_2(x,r,\mathbf{a}',\phi_n'(\mathbf{q}')) = 1] < \epsilon. \qquad (2.11)$$

where $\mathbf{q}, \mathbf{q}'$ are as above. Combining Eq. (2.10) and Eq. (2.11) we have that:

$$\mathop{\mathbf{E}}_{r,s}\left[\Pr_{(\mathbf{a},\mathbf{z})\in_R\mathcal{A}_{(\mathbf{q},\overline{\mathbf{w}}(q^*))}} \left[V_2(x,r,\mathbf{a},\phi_n'(\mathbf{q}')) = 1\right] - \Pr_{(\mathbf{a}',\mathbf{z}')\in_R\mathcal{A}_{(\mathbf{q},\sigma)}} \left[V_2(x,r,\mathbf{a}',\phi_n'(\mathbf{q}')) = 1\right] \right] > \delta''$$

where $\mathbf{q}, q^*, \mathbf{q}', \phi_n'(\mathbf{q}')$ are as above and $\mathbf{w} = V_1'(q^*, s)$.

By an averaging argument, there exist *fixed* values for r and s such that:

$$\Pr_{(\mathbf{a},\mathbf{z})\in_R \mathcal{A}_{(\mathbf{q},\overline{\mathbf{w}}(q^*))}}\left[V_2(x,r,\mathbf{a},\phi'_n(\mathbf{q}'))=1\right] - \Pr_{(\mathbf{a}',\mathbf{z}')\in_R \mathcal{A}_{(\mathbf{q},\sigma)}}\left[V_2(x,r,\mathbf{a}',\phi'_n(\mathbf{q}'))=1\right] > \delta'' \tag{2.12}$$

where $\mathbf{q}, q^*, \mathbf{q}', \phi'_n(\mathbf{q}')$ and $\mathbf{w}$ are *fixed* based on the fixed values of r and s.

Equation (2.12) gives a statistical test that distinguishes between the projections of the distributions $\mathcal{A}_{(\mathbf{q},\sigma)}$ and $\mathcal{A}_{(\mathbf{q},\overline{\mathbf{w}}(q^*))}$ to coordinates in $[k]$ with gap $> \delta''$, contradicting our assumption that $\mathcal{A}$ is δ''-no-signaling. Hence, $\mathcal{B}$ fools V into accepting $x \notin \mathcal{L}$ with probability $\geq \epsilon$. This concludes the proof of Claim 2.51.2. □

This concludes the proof of Lemma 2.51. □

To prove Lemma 2.50 we also need the following straightforward claim.

Claim 2.51.3. *Let $\mathcal{L}$ be a language and suppose that both $\mathcal{L}$ and $\overline{\mathcal{L}}$ (i.e., the complement language of $\mathcal{L}$) have* MIP *protocols with soundness ϵ against δ-no-signaling strategies. Assume that each of the* MIP *protocols uses k provers. Then, there exists a $2k$-prover interactive protocol for* computing *the function $\mathcal{L}(x) : \{0,1\}^* \to \{0,1\}$, where $\mathcal{L}(x) = 1$ if and only if $x \in \mathcal{L}$, with soundness ϵ against δ-no-signaling strategies. If both of the original* MIP *protocols use query alphabet D and answer alphabet Σ then the resulting $2k$-prover protocol has query alphabet D and answer alphabet $\Sigma \cup \{\perp\}$, where $\perp \notin \Sigma$ is a special symbol.*

Furthermore, if each of the original MIP *verifiers runs in time T_V then the resulting verifier runs in time $O(T_V + k \cdot \log(|\Sigma|))$ and if each of the original* MIP *(honest) provers runs in time T_P then the resulting provers run in time $O(T_P + T_{\mathcal{L}} + \log(|\Sigma|))$, where $T_{\mathcal{L}}$ is the time that it takes for a Turing machine to compute $\mathcal{L}(x)$.*

Proof. Let $(V, P_1, \ldots, P_k)$ be the MIP for $\mathcal{L}$ and let $(V', P'_1, \ldots, P'_k)$ be the MIP for $\overline{\mathcal{L}}$. We assume that V and V' are composed of two algorithms, a query generation algorithm and a verification algorithm. The query generation algorithm V_1 (resp., V'_1) on input x and a random string r (resp., r'), outputs k queries $\mathbf{q} = (q_1, \ldots, q_k) \in D^k$ (resp., $\mathbf{q}' = (q'_1, \ldots, q'_k) \in D^k$). The verification algorithm V_2 (resp., V'_2), on input x, the same random string r (resp., r') and k answers $\mathbf{a} = (a_1, \ldots, a_k) \in \Sigma^k$ (resp., $\mathbf{a}' = (a'_1, \ldots, a'_k) \in \Sigma^k$), outputs a bit representing whether to accept or reject the statement $x \in \mathcal{L}$ (resp., $x \notin \mathcal{L}$).

We construct a $2k$-prover protocol for *computing* $\mathcal{L}$ as follows. The first k provers are the same as $P_1, \ldots, P_k$ except that they first verify that $x \in \mathcal{L}$. If $x \notin \mathcal{L}$, then they answer with the special symbol $\perp$. Similarly, the last k provers are the same as $P'_1, \ldots, P'_k$ except that they first verify that $x \notin \mathcal{L}$. If $x \in \mathcal{L}$, then they send the special symbol $\perp$.

On input x and a random string (r, r') the query generation algorithm V''_1 computes $\mathbf{q} = V_1(x, r)$ and $\mathbf{q}' = V'_1(x, r')$, where $\mathbf{q}, \mathbf{q}' \in D^k$. For every $i \in [k]$, the query q_i is sent to the i-th prover and the query q'_i is sent to the $(k+i)$-th prover. Given the provers' answers $(\mathbf{a}, \mathbf{a}') \in (\Sigma \cup \{\perp\})^{k+k}$, the verification algorithm V''_2 works as follows:

1. If $V_2(x, r, \mathbf{a}) = 1$ and all the entries of $\mathbf{a}'$ are equal to $\perp$, then output 1 and halt.[20]
2. If $V_2'(x, r', \mathbf{a}') = 1$ and all the entries of $\mathbf{a}$ are equal to $\perp$, then output 0 and halt.
3. Output $\perp$ and halt.

To see that completeness holds note that if $x \in \mathcal{L}$ then the last k provers will send $\perp$ and, by the completeness of $(V, P_1, \ldots, P_k)$ the verifier will output 1. If $x \notin \mathcal{L}$ then the first k provers will send $\perp$ and, by the completeness of $(V', P_1', \ldots, P_k')$, the verifier will output 0. We proceed to show that soundness against δ-no-signaling strategies holds.

Fix $x \in \{0,1\}^*$ and assume toward a contradiction that there exists a δ-no-signaling strategy $\{\mathcal{A}_{(u,u')}\}_{(u,u') \in D^{k+k}}$ such that

$$\Pr_{\substack{r,r' \\ (\mathbf{a},\mathbf{a}') \in_R \mathcal{A}_{(\mathbf{q},\mathbf{q}')}}} [V_2''(x, (r, r'), (\mathbf{a}, \mathbf{a}')) \notin \{\mathcal{L}(x), \perp\}] \geq \epsilon,$$

where $\mathbf{q} = V_1(x, r)$ and $\mathbf{q}' = V_1'(x, r')$. For simplicity let us assume that $x \notin \mathcal{L}$. The case that $x \in \mathcal{L}$ is handled analogously (using the soundness of V', rather than the soundness of V). Thus,

$$\Pr_{\substack{r,r' \\ (\mathbf{a},\mathbf{a}') \in_R \mathcal{A}_{(\mathbf{q},\mathbf{q}')}}} [V_2''(x, (r, r'), (\mathbf{a}, \mathbf{a}')) \notin \{0, \perp\}] \geq \epsilon.$$

In particular, by the definition of V'':

$$\Pr_{\substack{r,r' \\ (\mathbf{a},\mathbf{a}') \in_R \mathcal{A}_{(\mathbf{q},\mathbf{q}')}}} [V_2(x, r, \mathbf{a}) = 1] \geq \epsilon. \tag{2.13}$$

where $\mathbf{q} = V_1(x, r)$ and $\mathbf{q}' = V_1'(x, r')$.

By an averaging argument, Eq. (2.13) implies that there exists a fixed value of r' such that

$$\Pr_{\substack{r \\ (\mathbf{a},\mathbf{a}') \in_R \mathcal{A}_{(\mathbf{q},\mathbf{q}')}}} [V_2(x, r, \mathbf{a}) = 1] \geq \epsilon. \tag{2.14}$$

where $\mathbf{q}' = V_1'(x, r')$ is a fixed value and $\mathbf{q} = V_1(x, r)$. For the rest of the proof of Claim 2.51.3 we fix r' and $\mathbf{q}'$ as above.

We use $\mathcal{A}$ to construct a δ-no-signaling strategy $\mathcal{B} = \{\mathcal{B}_u\}_{u \in D^k}$ that fools V into accepting $x \notin \mathcal{L}$ with probability $\geq \epsilon$. For every $u \in D^k$, the distribution $\mathcal{B}_u$ is defined by sampling $(\mathbf{a}, \mathbf{a}') \in_R \mathcal{A}_{(u,\mathbf{q}')}$ and outputting $\mathbf{a}$.

We first show that $\mathcal{B}$ violates the soundness of V_2 and then show that it is δ-no-signaling. Indeed, by the definition of $\mathcal{B}$ and using Eq. (2.14) it holds that

$$\Pr_{\mathbf{a} \in_R \mathcal{B}_\mathbf{q}}[V_2(x, r, \mathbf{a}) = 1] = \Pr_{\substack{r \\ (\mathbf{a},\mathbf{a}') \in_R \mathcal{A}_{(\mathbf{q},\mathbf{q}')}}} [V_2(x, r, \mathbf{a}) = 1] \geq \epsilon.$$

[20] If one of the entries of $\mathbf{a}$ (resp., $\mathbf{a}'$) is $\perp$, then we define $V_2(x, r, \mathbf{a}) = 0$ (resp., $V_2'(x, r', \mathbf{a}') = 0$).

We proceed to show that $\mathcal{B}$ is δ-no-signaling. Let $S \subset [k]$ and let $u_1, u_2 \in D^k$ such that $(u_1)_S = (u_2)_S$, and suppose that the statistical distance between $(\mathbf{a}_1)_S$ and $(\mathbf{a}_2)_S$ is more than δ, where $\mathbf{a}_1 \in_R \mathcal{B}_{u_1}$ and $\mathbf{a}_2 \in_R \mathcal{B}_{u_2}$. Then:

$$\begin{aligned}\delta &< \frac{1}{2}\sum_{\beta\in\Sigma^S}\left|\Pr_{\mathbf{a}_1\in_R\mathcal{B}_{u_1}}[(\mathbf{a}_1)_S=\beta]-\Pr_{\mathbf{a}_2\in_R\mathcal{B}_{u_2}}[(\mathbf{a}_2)_S=\beta]\right| \\ &= \frac{1}{2}\sum_{\beta\in\Sigma^S}\left|\Pr_{(\mathbf{a}_1,\mathbf{a}_1')\in_R\mathcal{A}_{(u_1,\mathbf{q}')}}[(\mathbf{a}_1)_S=\beta]-\Pr_{(\mathbf{a}_2,\mathbf{a}_2')\in_R\mathcal{A}_{(u_2,\mathbf{q}')}}[(\mathbf{a}_2)_S=\beta]\right|.\end{aligned}$$

Thus, the projections of the distributions $\mathcal{A}_{(u_1,\mathbf{q}')}$ and $\mathcal{A}_{(u_2,\mathbf{q}')}$ to coordinates in S are δ-far. Since $(u_1,\mathbf{q}')_S = (u_2,\mathbf{q}')_S$, this contradicts our assumption that $\mathcal{A}$ is δ-no-signaling.

This concludes the proof of Claim 2.51.3. □

Using Lemma 2.49, Lemma 2.51 and Claim 2.51.3 we are ready to prove Lemma 2.50.

Proof of Lemma 2.50. As a first step we replace the oracle $\{\phi_n : \{0,1\}^{n'} \to \{0,1\}^{n''}\}$ with a Boolean valued oracle $\{\phi'_n : \{0,1\}^{n^*} \to \{0,1\}\}$, where $n^* = n' + \log(n'')$, by having the oracle ϕ'_n on input $(z,i) \in \{0,1\}^{n'+\log(n'')}$ simply output the i-th bit of $\phi_n(z)$. Note that this step increases the number of oracle queries from ℓ to $\ell \cdot \log(n'')$.

Fix $t \geq 1$. Since $\{\phi'_n\}$ can be computed in linear (i.e., $O(n^*)$) space, by Lemma 2.49, the language $\mathcal{L}_{\phi'} = \{z \in \{0,1\}^* : \phi'(z) = 1\}$ has a $\mathsf{poly}(n^*)$-prover MIP with soundness error $\mathsf{poly}(n^*)\cdot 2^{-t}$ against 2^{-t}-no-signaling strategies. The verifier runs in time $t\cdot\mathsf{poly}(n^*)$ and the (honest) provers run in time $t\cdot\mathsf{poly}(2^{n^*})$. The query and answer alphabets are $\{0,1\}^{t\cdot\mathsf{poly}(n^*)}$. Similarly, the complement language $\overline{\mathcal{L}}_{\phi'}$ can also be computed in space $O(n^*)$ and so it has an MIP with the same parameters.

Thus, by Claim 2.51.3, there exists a $\mathsf{poly}(n^*)$-prover interactive protocol for computing $\{\phi'_n\}$ with soundness $\mathsf{poly}(n^*)\cdot 2^{-t}$ against 2^{-t}-no-signaling strategies. The verifier of the resulting protocol runs in time $t\cdot\mathsf{poly}(n^*)$ and the provers run in time $t\cdot\mathsf{poly}(2^{n^*})$. The query and answer alphabets are $\{0,1\}^{t\cdot\mathsf{poly}(n^*)}$.

The lemma follows by applying Lemma 2.51 iteratively $\ell\cdot\log(n'')$ times to the original MIP for $\mathcal{L}$ to remove all of the oracle queries. The resulting MIP has soundness $\epsilon + \ell\cdot\mathsf{poly}(n^*)\cdot\left(2^{-t}+\min(\delta,2^{-t})\right)$ against $\min(\delta,2^{-t})$-no-signaling provers. The MIP uses $k+\ell\cdot\mathsf{poly}(n^*)$ provers. The verifier runs in time $T_V + O(\ell\cdot t\cdot\mathsf{poly}(n^*))$ and each prover runs in time $T_P + O(\ell\cdot t\cdot\mathsf{poly}(2^{n^*}))$. The query alphabet is $D\cup\{0,1\}^{t\cdot\mathsf{poly}(n^*)}$ and the answer alphabet is $\Sigma\cup\{0,1\}^{t\cdot\mathsf{poly}(n^*)}$.[21] □

2.15 Proof of Theorem 2.4

Using the tools developed in the previous sections, we are finally ready to prove Theorem 2.4.

[21] Note that the alphabet sizes do not increase on every iteration.

Theorem 2.4. *Suppose that* $\mathcal{L} \in \mathsf{DTIME}(t(n))$*, where* $t = t(n)$ *satisfies* $\mathsf{poly}(n) \leq t \leq \mathsf{exp}(n)$*. Then, for any integer* $(\log t)^c \leq k \leq \mathsf{poly}(n)$*, where* c *is some (sufficiently large) universal constant, there exists an* MIP *for* $\mathcal{L}$ *with* $k \cdot \mathsf{polylog}(t)$ *provers and with soundness error* 2^{-k} *against* $2^{-k \cdot \mathsf{polylog}(t)}$*-no-signaling strategies.*

The verifier runs in time $n \cdot k^2 \cdot \mathsf{polylog}(t)$ *and the provers run in time* $\mathsf{poly}(t, k)$*. Each query and answer is of length* $k \cdot \mathsf{polylog}(t)$*.*

Proof. Let $\mathcal{L} \in \mathsf{DTIME}(t(n))$, where $\mathsf{poly}(n) \leq t(n) \leq \mathsf{exp}(n)$. Then, $\mathcal{L} \in \mathsf{DTISP}(t(n), s(n))$ where $\max(n, \log(t(n))) \leq s(n) \leq t(n)$. Fix $t = t(n)$ and $s = s(n)$.

Let $\mathbb{C}_n$ be a circuit on n inputs of size $N = O(t \cdot s)$ that computes $\mathcal{L}$ and let $\mathbb{C}'_n$ be the augmented circuit of size $N' = \mathsf{poly}(N)$, as described in Section 2.9. Let the parameters ℓ and $\mathbb{F}$ be as defined in Section 2.9.

Let $k' \leq \mathsf{poly}(n)$ be an integer such that $4|\mathbb{F}|^4 \leq k' \leq N'$. Consider the augmented PCP of Section 2.9, with respect to $\mathbb{C}'_n$, and the security parameter k'. Since $4|\mathbb{F}|^4 \leq k' \leq N'$, by Lemma 2.46, the PCP verifier has soundness ϵ' against (k_{max}, δ')-no-signaling strategies where:

$$
\begin{aligned}
|\mathbb{F}| &\leq 8(\log(N')^{10} \\
\ell &= 3\frac{\log(N')}{\log\log(N')} + 3 \\
r &= \frac{k'}{40\ell|\mathbb{F}|} \\
\epsilon' &= 2^{-r/2} \\
k_{max} &= k' \cdot \mathsf{polylog}(s) \cdot \log(t)|\mathbb{F}| + 12k'\ell|\mathbb{F}|^2 \\
\delta' &= \frac{1}{|\mathbb{F}|^{8k'\ell|\mathbb{F}|^2}}.
\end{aligned}
$$

Recall that this PCP is relative to an oracle $\hat{\phi}' : \mathbb{F}^\ell \to \mathbb{F}$ (see Section 2.5 and Section 2.9). As noted in Section 2.5.2.1, the total number of PCP queries as well as the total number of oracle queries is at most $6k'\ell|\mathbb{F}|^2 \leq k'\mathsf{polylog}N'$ and the running time of the verifier is $k'\mathsf{polylog}N'$. As noted in Section 2.5.1.1 (see also Section 2.9), the PCP can be generated in time $\mathsf{poly}(N')$. The query alphabet is of size at most $\mathsf{poly}(N')$ and the answer alphabet is of size $|\mathbb{F}| \leq \mathsf{polylog}N'$.

As a first step, we transform the PCP into an MIP. By applying Lemma 2.47, we obtain an MIP (relative to the same oracle) with soundness ϵ' against δ'-no-signaling strategies. The MIP uses $k_{max} \leq k'\mathsf{polylog}N'$ provers and $k'\mathsf{polylog}N'$ oracle queries. The query and answer alphabets remain unchanged. The running time of the MIP verifier is: $O(k'\mathsf{polylog}N' + k_{max} \log N') \leq k'\mathsf{polylog}N'$. The running time of each MIP prover is $\mathsf{poly}(N')$.

Recall that $\hat{\phi}' = \hat{\phi}_x + \hat{\phi}_{\mathcal{C}'} + \hat{\phi}_{extra}$ where $\hat{\phi}_x$, $\hat{\phi}_{\mathcal{C}'}$, $\hat{\phi}_{extra}$ are the low degree extensions of ϕ_x, $\phi_{\mathcal{C}'}$ and ϕ_{extra} respectively (see Section 2.5 and Section 2.9). As our second step, we replace the use of the oracle $\hat{\phi}'$ in the MIP with the oracle $\hat{\phi}_{\mathcal{C}'} + \hat{\phi}_{extra}$. This is done

by replacing each oracle query $\hat{\phi}'(z)$, by first querying the new oracle $(\hat{\phi}_{\mathcal{C}'} + \hat{\phi}_{extra})(z)$ and adding $\hat{\phi}_x(z)$, which is computed directly by the verifier, to the result. As noted in Section 2.5, the function $\hat{\phi}_x$ can be evaluated in time $n \cdot \mathsf{polylog} N'$. Thus, the resulting verifier runs in time $k'\mathsf{polylog} N' + (k'\mathsf{polylog} N') \cdot (n\mathsf{polylog} N') \leq n \cdot k' \cdot \mathsf{polylog} N'$ and all other parameters of the MIP remain unchanged.

At this point we apply Lemma 2.50 to obtain an MIP *without an oracle*. Note that, as pointed out in Section 2.5 (resp., Section 2.9), the function $\hat{\phi}_{\mathcal{C}'}$ (resp., $\hat{\phi}_{extra}$) can be computed in space that is linear in its input (i.e., $O(\log(|\mathbb{F}|^\ell)) = O(\log N')$ space). Therefore, we can apply Lemma 2.50, with respect to a parameter $t = \log_2(1/\delta')$, to obtain an MIP *without an oracle* that has soundness ϵ against δ-no-signaling strategies where

$$\epsilon = \epsilon' + k'\delta'\mathsf{polylog} N'$$
$$\delta = \delta'$$

The resulting MIP uses $k'\mathsf{polylog} N'$ provers. The resulting MIP verifier runs in time

$$n \cdot k' \cdot \mathsf{polylog} N' + O(k'\mathsf{polylog} N' \log(1/\delta)\mathsf{polylog} N') \leq n \cdot k'^2 \cdot \mathsf{polylog} N'$$

and the resulting provers run in time

$$\mathsf{poly}(N') + O(k'\mathsf{polylog} N' \cdot \log(1/\delta) \cdot \mathsf{poly}(2^{O(\log N')})) \leq \mathsf{poly}(N').$$

The query alphabet is of size $\mathsf{poly}(N') + 2^{\log(1/\delta)\cdot\mathsf{polylog}(N')} \leq 2^{k'\cdot\mathsf{polylog} N'}$ and the answer alphabet is of size $\mathsf{polylog} N' + 2^{\log(1/\delta)\cdot\mathsf{polylog}(N')} \leq 2^{k'\cdot\mathsf{polylog} N'}$.

The theorem follows by setting $k' = k \cdot \mathsf{polylog} N'$ and noting that $N' = \mathsf{poly}(t)$.[22] □

2.16 From No-Signaling MIP's to One Round Arguments

In this section we show how to transform any MIP that has soundness against no-signaling strategies into a 1-round argument system, using a fully-homomorphic encryption scheme (FHE) (or alternatively, a computational private information retrieval (PIR) scheme).

Theorem 2.11. *Suppose that the language $\mathcal{L}$ has an ℓ prover* MIP *that has ϵ soundness against δ-no-signaling strategies. Let D be the query alphabet and Σ be the answer alphabet of the* MIP*. Let $\tau = \tau(n) \geq \max(\ell, \log(|\Sigma|), \log(|D|))$ be a security parameter, where n denotes the input length of the* MIP*. For every $S = S(\tau) \geq \tau$ such that $S \geq \max(n, 2^{\ell\log(|\Sigma|)})$ and $\delta' = \delta'(\tau)$ such that $\delta' \leq \delta/\ell$, if there exists an (S, δ')-secure* FHE*, then the language $\mathcal{L}$ has a 1-round argument system with soundness (S, ϵ).*

If the MIP *verifier runs in time T_V, then the running time of the resulting verifier is $T_V + T_{\mathsf{FHE}}(\tau)$ where T_{FHE} is a polynomial that depends only on the encryption scheme*

[22] Note that we assumed that $k' < N'$. If this is not the case then we can increase N' by adding sufficiently many dummy gates to the circuit $\mathbb{C}'_n$.

(and not on the language $\mathcal{L}$). If the running time of each MIP *prover is T_P, then the running time of the resulting prover is* $\mathsf{poly}\big(T_P, \tau, n\big)$*. The total communication in the resulting argument-system is of length* $\mathsf{poly}(\tau)$*.*

Proof. Let $(V, P_1, \ldots, P_\ell)$ be an ℓ prover MIP for $\mathcal{L}$ with soundness ϵ against δ-no-signaling strategies. Let D be the query alphabet and Σ be the answer alphabet. Since $(V, P_1, \ldots, P_\ell)$ is a 1-round protocol, it will be convenient for us to think of V as being composed of two algorithms that use the same randomness, V_1 and V_2. The first algorithm, V_1, on input x and the random string r outputs a sequence of ℓ queries $\mathbf{q} \in D^\ell$. The second algorithm, V_2, on input x, the same random string r and answers $\mathbf{a} \in \Sigma^\ell$ outputs a bit that represents whether it believes that $x \in \mathcal{L}$. We assume without loss of generality that the provers algorithms $P_1, \ldots, P_\ell$ are deterministic.

Let $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval})$ be an FHE and let $\tau = \tau(n)$ be a security parameter. We use the MIP and FHE to construct an argument system (V', P') as follows. The verifier, given as input x, first invokes V_1 on input x and a random string r to obtain a sequence of ℓ queries $\mathbf{q} = (q_1, \ldots, q_\ell) \in D^\ell$. Then, for every $i \in [\ell]$, the verifier invokes $\mathsf{Gen}(1^\tau)$, where $\tau = \tau(|x|)$, to obtain a key-pair $(\mathsf{pk}_i, \mathsf{sk}_i)$. The verifier then runs $\mathsf{Enc}_{\mathsf{pk}_i}(q_i)$ to obtain a ciphertext $\hat{q}_i$, for every $i \in [\ell]$. We denote $\mathsf{pk} = (\mathsf{pk}_1, \ldots, \mathsf{pk}_\ell)$, and $\hat{\mathbf{q}} \stackrel{\text{def}}{=} (\hat{q}_1, \ldots, \hat{q}_\ell)$. The verifier sends $(\mathsf{pk}, \hat{\mathbf{q}})$ to P'.

The prover P' is given as input x and a message $(\mathsf{pk}, \hat{\mathbf{q}})$ from the verifier. For every $i \in [\ell]$, let $\mathbb{C}_{x,i} : D \to \Sigma$ be a Boolean circuit that on input q computes the function $P_i(x, q)$. For every $i \in [\ell]$, the prover P' computes $\hat{a}_i = \mathsf{Eval}(\mathsf{pk}_i, C_{x,i}, \hat{q}_i)$. The prover sends $\hat{\mathbf{a}} \stackrel{\text{def}}{=} (\hat{a}_1, \ldots, \hat{a}_\ell)$ to the verifier.

The verifier V', given the message $\hat{\mathbf{a}}$ from the prover, computes $a_i = \mathsf{Dec}_{\mathsf{sk}_i}(\hat{a}_i)$, for every $i \in [\ell]$. The verifier outputs the result of $V_2(x, (a_1, \ldots, a_\ell), r)$, where r is the same random string used by V_1 to generate the queries.

We proceed to show that (V', P') is an argument system with soundness (S, ϵ) (see definition in Section 2.4.8).

Completeness. Let $x \in \mathcal{L}$. By the construction and the correctness of the FHE protocol, for every $i \in [\ell]$ it holds that $a_i = P_i(x, q_i)$, with overwhelming probability. When V_2 is invoked with the answers of the honest provers $P_1, \ldots, P_\ell$, by the (perfect) completeness of the MIP, the verifier V outputs 1. Hence, V' accepts with overwhelming probability.

Soundness. Let $\{P^*_n\}_{n \in \mathbb{N}}$ be a family of circuits of size at most $\mathsf{poly}(S(n))$ such that there exist infinitely many $x \notin \mathcal{L}$ such that

$$\Pr[(P^*_{|x|}, V')(x) = 1] > \epsilon, \tag{2.15}$$

where $(P^*_{|x|}, V')(x)$ denotes the output of V' after interacting with the prover $P^*_{|x|}$ on common input x (and the probability is over the random coins of V'). We show a contradiction by constructing a δ-no-signaling (cheating) strategy that fools the underlying MIP verifier V into accepting some $x \notin \mathcal{L}$ with probability greater than ϵ.

For every $x \notin \mathcal{L}$, consider an MIP prover strategy $\mathcal{A}^{(x)} \stackrel{\text{def}}{=} \{\mathcal{A}^{(x)}_{\mathbf{q}}\}_{\mathbf{q}\in D^\ell}$, where for every $\mathbf{q} = (q_1, \ldots, q_\ell) \in D^\ell$, the distribution $\mathcal{A}^{(x)}_{\mathbf{q}}$ is sampled as follows. First, for every $i \in [\ell]$ invoke $\mathsf{Gen}(1^\tau)$, where $\tau = \tau(|x|)$, to obtain $(\mathsf{pk}_i, \mathsf{sk}_i)$ and compute $\hat{q}_i \in_R \mathsf{Enc}_{\mathsf{pk}_i}(q_i)$. Then, compute $\hat{\mathbf{a}} = (\hat{a}_1, \ldots, \hat{a}_\ell) \in_R P^*_{|x|}(x, (\mathsf{pk}, \hat{\mathbf{q}}))$, where $\mathsf{pk} = (\mathsf{pk}_1, \ldots, \mathsf{pk}_\ell)$ and $\hat{\mathbf{q}} = (\hat{q}_1, \ldots, \hat{q}_\ell)$, and for every $i \in [\ell]$, set $a_i = \mathsf{Dec}_{\mathsf{sk}_i}(\hat{a}_i)$. Finally, output $\mathbf{a} \stackrel{\text{def}}{=} (a_1, \ldots, a_\ell)$.

By the definition of $\mathcal{A}^{(x)}$ and V' and using Eq. (2.15), for infinitely many $x \notin \mathcal{L}$, it holds that

$$\Pr_{\substack{r \\ \mathbf{a}\in_R \mathcal{A}^{(x)}_{\mathbf{q}}}}[V_2(x, \mathbf{a}, r) = 1] = \Pr\left[(P^*_{|x|}, V')(x) = 1\right] > \epsilon$$

where $\mathbf{q} = V_1(x, r)$. It remains to be shown that for all sufficiently large $x \notin \mathcal{L}$, the strategy $\mathcal{A}^{(x)}$ is δ-no-signaling.

We need to prove that for all sufficiently large x, every $S \subseteq [\ell]$, and every two sequences of queries $\mathbf{q} = (q_1, \ldots, q_\ell) \in D^\ell$ and $\mathbf{q}' = (q'_1, \ldots, q'_\ell) \in D^\ell$ such that $\mathbf{q}_S = \mathbf{q}'_S$ (i.e., $q_i = q'_i$ for all $i \in S$), the following two distributions are δ-close:

- $\mathbf{a}_S$ where $\mathbf{a} \in_R \mathcal{A}^{(x)}_{\mathbf{q}}$; and
- $\mathbf{a}'_S$ where $\mathbf{a}' \in_R \mathcal{A}^{(x)}_{\mathbf{q}'}$.

Toward this end, assume for a contradiction that for infinitely many x this is not the case. That is, for infinitely many x, there exist corresponding S, $\mathbf{q}$, $\mathbf{q}'$ and a distinguisher $\mathcal{D}_x$ such that

$$\left| \Pr_{\mathbf{a}\in_R \mathcal{A}^{(x)}_{\mathbf{q}}}[\mathcal{D}_x(\mathbf{a}_S) = 1] - \Pr_{\mathbf{a}'\in_R \mathcal{A}^{(x)}_{\mathbf{q}'}}[\mathcal{D}_x(\mathbf{a}'_S) = 1] \right| > \delta. \tag{2.16}$$

Since $\mathcal{D}_x$ takes as input a string of length at most $\ell \cdot \log(|\Sigma|)$, it can be implemented by a circuit of size at most $\mathsf{poly}(2^{\ell \cdot \log(|\Sigma|)})$. We use $\{\mathcal{D}_x\}_x$ to construct a family of circuits $\{\mathbb{C}_\tau\}_\tau$ that breaks the security of the underlying FHE scheme.

For every x as above and for $\tau = \tau(|x|)$, let $\mathbb{C}_\tau$ be a circuit that takes as input a set of public-keys $\{\mathsf{pk}_i\}_{i\in[\ell]\setminus S}$ (with respect to security parameter τ) and a set of ciphertexts $\{c_i\}_{i\in[\ell]\setminus S}$. We show that the circuit $\mathbb{C}_\tau$ distinguishes between the case that (1) each c_i was sampled from $\mathsf{Enc}_{\mathsf{pk}_i}(q_i)$; and the case that (2) each c_i was sampled from $\mathsf{Enc}_{\mathsf{pk}_i}(q'_i)$. The circuit $\mathbb{C}_\tau$ works as follows:

1. For every $i \in S$, sample $(pk_i, sk_i) \in_R \mathsf{Gen}(1^\tau)$ and $c_i \in_R \mathsf{Enc}_{pk_i}(q_i)$. Set $\mathsf{pk} = (pk_1, \ldots, pk_\ell)$ and $\mathbf{c} = (c_1, \ldots, c_\ell)$ (note that for $i \notin S$, the values pk_i and c_i are given as input to the circuit).
2. Compute $\hat{\mathbf{a}} \stackrel{\text{def}}{=} (\hat{a}_1, \ldots, \hat{a}_\ell) = P^*_{|x|}(x, \mathsf{pk}, \mathbf{c})$, where $P^*_{|x|}(x, \mathsf{pk}, \mathbf{c})$ denotes the output of $P^*_{|x|}$ given as input x and a message $(\mathsf{pk}, \mathbf{c})$. (Note that x is fixed inside the description of $\mathbb{C}_\tau$.)
3. For every $i \in S$, set $a_i = \mathsf{Dec}_{sk_i}(\hat{a}_i)$.
4. Output $\mathcal{D}_x(\mathbf{a}_S)$ where $\mathbf{a}_S = (a_i)_{i\in S}$.

Note that $\mathbb{C}_\tau$ has size $\mathsf{poly}\left(2^{\ell \cdot \log(|\Sigma|)}, \tau, S(\tau), |x|\right) \leq \mathsf{poly}(S(\tau))$.

By Eq. (2.16), the circuit $\mathbb{C}_\tau$ distinguishes between the two cases with probability δ for infinitely many values of τ. By a standard hybrid argument we obtain a circuit that breaks the semantic security of the encryption scheme with distinguishing gap at least $\delta/\ell \geq \delta'(\tau)$ in contradiction to our assumption. Thus, we obtain that for all sufficiently large $x \notin \mathcal{L}$, the prover strategy $\mathcal{A}^{(x)}$ is δ-no-signaling and the lemma follows. □

2.17 Delegation for P

Using all the results above, we are ready to prove Theorem 2.8.

Theorem 2.8. *Suppose that $\mathcal{L} \in \mathsf{DTIME}(t(n))$, where $t = t(n)$ satisfies $\mathsf{poly}(n) \leq t \leq \mathsf{exp}(n)$. Let $\tau = \tau(n)$ be a security parameter such that $\log(t) \leq \tau \leq \mathsf{poly}(t)$. Let $S = S(\tau) \geq \tau$ such that $2^{(\log(t))^c} \leq S \leq 2^{\mathsf{poly}(n)}$ and $S \leq 2^{\max(n,\tau)}$, where c is some sufficiently large universal constant. If there exists an $\left(S, 2^{-\sqrt{\log S}}\right)$-secure FHE, then $\mathcal{L}$ has a 1-round argument system with soundness $\left(S, 2^{-\frac{\sqrt{\log S}}{\mathsf{polylog}(t)}}\right)$. The verifier runs in time $n \cdot \log(S) \cdot \mathsf{polylog}(t) + \mathsf{poly}(\tau)$ and the prover runs in time $\mathsf{poly}(t)$. The total communication is of length $\mathsf{poly}(\tau)$.*

Proof. Suppose that $\mathcal{L} \in \mathsf{DTIME}(t(n))$, where $t = t(n)$ satisfies $\mathsf{poly}(n) \leq t \leq \mathsf{exp}(n)$. Let $\tau = \tau(n)$ be a security parameter such that $\log(t) \leq \tau \leq \mathsf{poly}(t)$. Let $S = S(\tau)$ such that $2^{(\log(t))^{c''}} \leq S \leq 2^{\mathsf{poly}(n)}$ and $S \leq 2^{\max(n,\tau)}$, where $c'' = 2(c + c')$, the constant c is as in Theorem 2.4 and c' is some sufficiently large universal constant. Let $\delta \stackrel{\text{def}}{=} 2^{-\sqrt{\log S}}$ and $k \stackrel{\text{def}}{=} \frac{\sqrt{\log S}}{(\log(t))^{c'}}$. Note that by the restriction on S, and our setting of k and δ, it holds that:

1. $(\log(t))^c \leq k \leq \mathsf{poly}(n)$.
2. $S \geq \max\left(n, 2^{k^2 \mathsf{polylog}(t)}\right)$.
3. $\delta \leq 2^{-k\mathsf{polylog}(t)}$.

(where the latter two conditions are obtained by setting c' to be a sufficiently large constant).

By applying Theorem 2.4 (with respect to the parameter k) to the language $\mathcal{L}$, we obtain an MIP for $\mathcal{L}$ with $k \cdot \mathsf{polylog}(t)$ provers and with soundness error 2^{-k} against $2^{-k \cdot \mathsf{polylog}(t)}$-no-signaling strategies. The verifier of the MIP runs in time $n \cdot k^2 \cdot \mathsf{polylog}(t) \leq n \cdot \log(S) \cdot \mathsf{polylog}(t)$ and the provers run in time $\mathsf{poly}(t, k)$. Each query and answer is of length $k \cdot \mathsf{polylog}(t)$.

Assume that there exists an (S, δ)-secure FHE. By Theorem 2.11 (and our setting of k, S and δ), we obtain that $\mathcal{L}$ has a 1-round argument system with soundness $(S, 2^{-k})$. The running time of the verifier is $n \cdot \log(S) \cdot \mathsf{polylog}(t) + \mathsf{poly}(\tau)$ and the running time of the prover is $\mathsf{poly}(t)$. The total communication is of length $\mathsf{poly}(\tau)$.

□

Replacing FHE with PIR. As noted in the introduction, Theorem 2.11 and Theorem 2.8 can be based on the assumption that a (sufficiently hard) PIR scheme exists rather than a full-blown FHE. Indeed, instead of encrypting the MIP queries, the verifier can send them encapsulated inside PIR queries. The prover, instead of homomorphically evaluating the MIP prover algorithm on encrypted queries, can pre-compute the answers to every possible query and answer according to a corresponding PIR database. However, one must be careful since in the straightforward implementation, the running time of the prover is exponential in the communication complexity of the underlying MIP. This is a real concern in our protocol since the MIP has poly-logarithmic communication complexity (which translates to a quasi-polynomial running time of the prover). We resolve this issue by noting that the next message function of the prover depends only on a logarithmic number of bits and therefore the PIR database can be constructed in polynomial-time.

Appendix for Chapter 2

2.A Computing LDE over Characteristic 2 Fields

Recall that $\mathbb{G}$ is a finite field of characteristic 2, $H_{\mathbb{G}} \subseteq \mathbb{G}$ is an arbitrary subset of $\mathbb{G}$ and $m_{\mathbb{G}}$ is the dimension.

Proposition 2.52. *There exists a Turing Machine that runs in time* $\mathsf{poly}(|\mathbb{G}|^{m_{\mathbb{G}}})$ *and space* $O\big(m_{\mathbb{G}} \cdot \log(|\mathbb{G}|) + \mathsf{polylog}(|\mathbb{G}|)\big)$ *and outputs a Boolean circuit of depth* $O\big(m_{\mathbb{G}} \cdot \log(|\mathbb{G}|) + \log m_{\mathbb{G}} \cdot \mathsf{polylog}(|\mathbb{G}|)\big)$ *and size* $\mathsf{poly}(|\mathbb{G}|^{m_{\mathbb{G}}})$ *that on input a truth table of a function* $\alpha : H_{\mathbb{G}}^{m_{\mathbb{G}}} \to \{0,1\}$ *outputs the truth table of the* LDE $\hat{\alpha} : \mathbb{G}^{m_{\mathbb{G}}} \to \mathbb{G}$ *of* α.

Proof. By the proof of Proposition 2.1,

$$\hat{\alpha}(z) = \sum_{x \in H_{\mathbb{G}}^{m_{\mathbb{G}}}} \hat{\beta}_x(z) \cdot \alpha(x) \tag{2.17}$$

where each $\hat{\beta}_x$ can be computed by an arithmetic circuit (over $\mathbb{G}$) of depth $O(\log(m_{\mathbb{G}}) + \log(|H_{\mathbb{G}}|))$ and size $\mathsf{poly}(|H_{\mathbb{G}}|, m_{\mathbb{G}})$ and each arithmetic circuit can be generated (by a Turing Machine) in time $\mathsf{poly}(|H_{\mathbb{G}}|, m_{\mathbb{G}}, \log|\mathbb{G}|)$ and in space $O(\log(|\mathbb{G}|) + \log(m_{\mathbb{G}}))$.

Since the field operations can be implemented by Boolean circuits of depth $\mathsf{polylog}(|\mathbb{G}|)$ and size $\mathsf{polylog}(|\mathbb{G}|)$, we can replace each arithmetic circuit by a Boolean circuit of depth $\mathsf{polylog}(|\mathbb{G}|) \cdot \log(m_{\mathbb{G}})$ and size $\mathsf{poly}(|H_{\mathbb{G}}|, m_{\mathbb{G}}, \log(|\mathbb{G}|))$. Each Boolean circuit can be generated in time $\mathsf{poly}(|H_{\mathbb{G}}|, m_{\mathbb{G}}, \log(|\mathbb{G}|))$ and in space $O(\mathsf{polylog}(|\mathbb{G}|) + \log(m_{\mathbb{G}}))$.

The sum of the terms in Eq. 2.17 can be computed by an arithmetic circuit of depth $O(\log(|H_{\mathbb{G}}|^{m_{\mathbb{G}}}))$ and size $O(|H_{\mathbb{G}}|^{m_{\mathbb{G}}})$. Moreover, since *addition* over $\mathbb{G}$ can be computed by a *constant* depth (fan-in 2) Boolean circuit (because $\mathbb{G}$ has characteristic 2), the sum can be computed by a Boolean circuit of depth $O(\log(|H_{\mathbb{G}}|^{m_{\mathbb{G}}}))$ and size $\mathsf{polylog}(|\mathbb{G}|) \cdot O(|H_{\mathbb{G}}|^{m_{\mathbb{G}}})$. The latter Boolean circuit can be generated in time $\mathsf{polylog}(|\mathbb{G}|) \cdot O(|H_{\mathbb{G}}|^{m_{\mathbb{G}}})$ and space $O(\log(|H_{\mathbb{G}}|^{m_{\mathbb{G}}}))$. □

Chapter 3

Non-interactive Proofs of Proximity

3.1 Introduction

Understanding the power and limitations of sublinear algorithms is a central question in the theory of computation. The study of *property testing*, initiated by Rubinfeld and Sudan [RS96] and Goldreich, Goldwasser and Ron [GGR98], aims to address this question by considering highly-efficient randomized algorithms that solve approximate decision problems, while only inspecting a small fraction of the input. Such algorithms, commonly referred to as *property testers*, are given oracle access to some object, and are required to determine whether the object has some predetermined property, or is far (say, in Hamming distance) from every object that has the property. Remarkably, it turns out that many natural properties can be tested by making relatively few queries to the object.

Once a model of computation has been established, a fundamental question that arises is to understand the power of *proof-systems* in this model. Recall that a proof-system consists of a powerful prover that wishes to convince a weak verifier, which does not trust the prover, of the validity of some statement. Since verifying is usually easier than computing, using the power of proofs, it is often possible to overcome limitations of the basic model of computation. In this paper we study proof-systems in the context of property testing, with the hope that by augmenting testers with proofs we can indeed overcome inherent limitations of property testers.

Thus, we are interested in proof-systems in which the verifier reads only a small fraction of the input. Of course we cannot hope for such a verifier to reject *every* false statement. Instead, as is the case in property testing, we relax the soundness condition and only require that it be impossible to convince the verifier to accept statements that are *far* from true statements. Such proof-systems were first introduced by Ergün, Kumar and Rubinfeld [EKR04] and were recently further studied by Rothblum, Vadhan and Wigderson [RVW13] who were motivated by applications to *delegation of computation* in sublinear time. Rothblum *et al.* [RVW13] showed that by allowing a property tester to interact with an untrusted prover (who can read the *entire* input), sublinear time verification is indeed possible for a wide class of properties. As in the property testing framework, the tester is only assured of the proximity of the input to the property and

hence such protocols are called *interactive proofs of proximity* (IPPs).

3.1.1 The Notion of MAP

In this work, we also consider proofs of proximity, but restrict the verification process to be *non-interactive*. In other words, we augment the property testing framework by allowing the tester full and free access to an (alleged) proof. Such a proof-aided tester for a property Π, is given oracle access to an input x and free access to a proof string w, and should distinguish between the case that $x \in \Pi$ and the case that x is far from Π while using a sublinear number of queries. We require that for inputs $x \in \Pi$, there exist a proof that the tester accepts with high probability, and for inputs x that are *far* from Π no proof will make the tester accept, except with some small probability of error.

This type of proof-system can be viewed as the property testing analogue of an NP proof-system (whereas IPP is the property testing analogue of IP). However, in contrast to polynomial-time algorithms, sublinear time algorithms inherently rely on *randomization*.[1] Since an NP proof-system in which the verifier is randomized is known as a *Merlin-Arthur* (MA) proof-system, we call these sublinear non-interactive proof-systems *Merlin-Arthur proofs of proximity* or simply MAPs.

Following the property testing literature, we consider the number of queries that the tester makes as the main computational resource. We ask whether non-interactive proofs can reduce the number of queries that property testers make, and if so by how much. (We note that [RVW13] showed that it is possible to significantly reduce the query complexity of property testers using interactive proofs, but their proof systems rely fundamentally on two-way interaction.)

Given the (widely believed) power of proofs in the context of *polynomial-time* computation, one would hope that proofs can help decrease the number of queries that is needed to test various properties. This is indeed the case. In fact, for every property Π, consider a proof-system for the statement $x \in \Pi$, wherein the proof w is simply equal to x. In order to verify the statement, the tester need only verify that indeed $w \in \Pi$ and that w is close to x (i.e., that the relative Hamming distance between w and x is a small constant). The former check can be carried out without any queries to x, whereas for the latter a constant number of queries suffice. Thus, using a proof of length linear in the input size, *any* property can be tested using a constant number of queries (furthermore, the tester has one-sided error). In contrast, there exist properties for which *linear* lower bounds on the query complexity of standard property testers are known (cf. [GGR98]).

The foregoing discussion leads us to view the proof length, in addition to the number of queries, as a central computational resource, which we should try to minimize. Thus, we measure the complexity of an MAP by the total amount of information available to the tester, namely, the sum of the MAPs query complexity (i.e., the number of queries that the tester makes) and proof complexity (i.e., the length of the proof). In this work

[1] It is not difficult to see that the sublinear time *deterministic* computation or even verification is limited to trivial properties (cf. [GS10b]).

we study the complexity of MAPs in comparison to property testers and to the recently introduced IPPs.

A Concrete Motivation. We note that the non-interactive nature of such proof-systems may have significant importance to applications such as *delegation of computation*. Specifically, consider a scenario wherein a computationally weak client has reliable query access to a massive dataset x. The client wishes to compute a function f on x, but its limited power, along with the massive size of the dataset, prevents it from doing so. In this case, the client can use a powerful server (e.g., a cloud computing provider) to compute $f(x)$ for it. However, the client may be distrustful of the server's answer (as it might cheat or make a mistake). Thus, an MAP for f can be used to verify the correctness of the computation delegated to the server: Given access to x, the server can send the value $y = f(x)$, together with a proof of proximity that ascertains that x is close to a dataset x' for which $f(x') = y$. The latter can be verified using an MAP verifier that makes only a small number of queries to x.

We emphasize that the advantage in using *non-interactive* proofs of proximity (rather than interactive ones) is not only in removing the need for two-way communication, but also: (1) the proof can be "annotated" to the dataset by the server in a cheap off-line phase; and (2) the proof can be re-used for multiple clients.

The Computational Complexity of Generating and Verifying the Proof. As noted above, we view the number of queries and proof length as the main computational resources. It is natural to also consider the computational complexity of generating and verifying the proof. However, in this work our main focus is on the query and proof complexities. Still, we note that unless stated otherwise, our protocols can be implemented efficiently; that is, the proof can be generated in *polynomial-time* and verified in *sublinear-time*.

Comparison with PCPs of Proximity. PCPs of proximity (PCPPs), first studied by Ben-Sasson *et al.* [BSGH+06] and by Dinur and Reingold [DR06] (where they are called assignment testers) are also non-interactive proof-systems in which the verifier has oracle access to an object, and needs to decide whether the object is close to having a predetermined property. However, PCPPs differ from MAPs in that the verifier is only given *query* (i.e., oracle) access to the proof, whereas in MAPs, the verifier has free (*explicit*) access to the proof. Indeed, in contrast to MAPs, the proof string in PCPPs is typically of super-linear length (but only a small fraction of it is actually read at random). Thus, PCPPs may be thought of as the PCP analogue of property testing, whereas MAPs are the NP analogue of property testing.

In fact, considering a variety of non-interactive proof-systems that differ in whether the main input and the proof are given explicitly or implicitly (i.e., via query access or free access), leads to the taxonomy depicted in Table 3.1. Interestingly, the three other variants, corresponding to NP, PCP and PCPP, have all been well studied. Thus, we view the notion of MAPs as completing this taxonomy of non-interactive proof-systems.

	Access to Proof		
Access to Main Input	**No Proof**	**Free Access**	**Oracle Access**
Free Access	$\mathcal{P}$	NP or MA	PCP
Oracle Access	Property Testers	MAP (**this work**)	PCPP

Table 3.1: Taxonomy of non-interactive proof-systems.

3.1.2 The Power of MAP

The first question that one might ask about the model of MAPs is whether proofs give a significant savings in the query complexity of property testers (indeed, such savings are the main reason to introduce a proof-system in the first place). Given the above discussion on the importance of bounding the proof length, we seek savings in the query complexity while using only a relatively short proof. Our first result shows that indeed there exists a property for which a dramatic saving is possible:

Informal Theorem 3.1 (see Theorem 3.7)**.** *There exists a (natural) property that has an* MAP *that uses a logarithmic-length proof and only a* constant *number of queries, but requires* $n^{0.999}$ *queries for every property tester.*
Here and throughout this work, n denotes the length of the object being tested.

Having established an exponential separation between property testers and MAPs, we continue our study of MAPs by asking how many queries can be saved by slightly increasing the length of the proof. The following result shows a property for which a smooth *multiplicative* trade-off, which is (almost) tight, between the number of queries and length of the proof holds:

Informal Theorem 3.2 (see Theorem 3.13)**.** *There exists a (natural) property* Π *such that, for every* $p \geq 1$*, there is an* MAP *for* Π *that uses a proof of length* p *and makes* $\frac{n^{0.999}}{p}$ *queries. Furthermore, for every* p*, the trade-off is (almost) tight.*

Recall that for property testers huge gaps may exist between the query complexity of testers that have *one-sided error* and the query complexity of testers that have two-sided error (where a one-sided tester is one that accepts every object that has the property with probability 1). Notable examples for properties for which such gaps are known are *Cycle-Freeness* in the bounded degree graph model (see [CGR+12]) and *ρ-Clique* in the dense graph model (see [GGR98]). In contrast, we observe that *such gaps can not exist in the case of* MAP*s.*

Informal Theorem 3.3 (see Theorem 3.20)**.** *Any two-sided error* MAP *can be converted to have* one-sided error *with only a poly-logarithmic overhead to the query and proof complexities.*

Since every property tester can be viewed as an MAP that uses an empty proof, as an immediate corollary, we obtain a transformation from every two-sided error *property tester* into a one sided MAP that uses a proof of only poly-logarithmic length (with only a poly-logarithmic increase in the query complexity). Moreover, since (as noted above) there are well-known properties for which *one-sided error* property testing is exponentially harder than *two-sided error* property testing, Informal Theorem 3.3 implies an exponential separation between MAPs (with poly-logarithmically long proofs) and *one-sided error* property testing. We note that Informal Theorem 3.1 shows such a separation for the more general case of two-sided error.

We note that all of the explicit properties that were discussed thus far are properties "with distance"; that is, properties for which every two objects that have the property are far apart. In other words, the set of objects forms an error-correcting code. This distance, along with a form of local *self-correction*, is a crucial ingredient of the foregoing MAPs. In contrast, all of the properties described next are properties "without distance". Hence, the power of MAPs is not limited to properties with distance.

MAPs for parameterized concatenation problems. We identify a family of natural properties, for which it is possible to construct efficient MAPs, by using a generic scheme. Specifically, for every problem that can be expressed as a parameterized concatenation problem, we show how to construct an efficient MAP that allows a trade-off between the query and proof complexity. Loosely speaking, a property Π is a parameterized concatenation problem if $\Pi = \Pi_{\alpha_1} \times \cdots \times \Pi_{\alpha_k}$, for some integer k, where each property Π_{α_i} is a property parameterized by α_i.

Using this generic scheme, we obtain MAPs for a couple of natural problems, including: (1) approximating the Hamming weight of a string, and (2) graph orientation problems. (For more details, see Section 3.6).

MAPs for graph properties. To see that MAPs are also useful for testing graph properties, we consider the problem of testing bipartiteness in the *bounded-degree* graph model. We construct an MAP protocol for verifying bipartiteness of *rapidly-mixing graphs*, with proof complexity p and query complexity q, *for every* p and q such that $p \cdot q \geq N$ (where N is the number of vertices in the graph). In particular, we obtain an MAP verifier that uses a proof of length $N^{2/3}$ and makes only $N^{1/3}$ queries. This stands in contrast to the $\Omega(\sqrt{N})$ lower bound on the query complexity of property testers (which do not use a proof), shown by Goldreich and Ron [GR02], which also holds for *rapidly-mixing graphs*. We remark that in [RVW13] a (multi-round) IPP was given for the same problem (see Section 3.7).

We note that in the *dense* graph model, testing bipartiteness (or more generally k-colorability) can be easily done using only $O(1/\varepsilon)$ queries (where ε represents the desired proximity to the object) when given a proof that is simply the k-coloring of the graph (which can be represented by $N \log_2 k$ bits where N is the number of vertices and k is the

number of colors).[2] In contrast, for standard property testers such query complexity is impossible (see [BT04]). We note that a similar protocol (described as a PCPP) for testing bipartiteness in the dense graph model was suggested in [EKR04] and in [BSGH+06].

MAPs for sparse properties. If a property is relatively sparse, in the sense that it contains only t objects, then a proof of length $\log_2 t$ (which fully describes the object) can be used, and only $O(1/\varepsilon)$ queries suffice to verify the proof's consistency with the object. Using this observation we note that testing k-juntas and k-linearity can be verified using only $O(1/\varepsilon)$ queries and a proof of length $O(k \log n)$, whereas a lower bound of $\Omega(k)$ queries is well-known for standard property testers (cf. [Bla10]).

3.1.3 The Limitations of MAP

In the previous section, we described results that exhibit the power of MAPs. But what are the limitations of MAPs? As discussed above, a proof of linear length suffices to reduce the query complexity to $O(1/\varepsilon)$. Moreover, Informal Theorem 3.1 shows that even a logarithmically long proof can be extremely useful for a specific property. Thus, it is natural to ask whether a *sublinear* proof can reduce the query complexity for *every* property. The following result shows that for *almost all* properties, even a proof of length $n/100$ cannot improve the query complexity by more than a constant factor.

Informal Theorem 3.4 (see Theorem 3.22)**.** *For almost all properties, every* MAP *verifier that uses a proof of length* $n/100$ *must make* $\Omega(n)$ *queries.*[3]

Although Theorem 3.22 holds for most properties, finding an *explicit* property for which a similar statement holds remains an interesting open question. We note that Informal Theorem 3.4 improves upon a result of Fischer *et al.* [FGL14] (see discussion in Section 3.1.5).

Since Informal Theorem 3.4 shows that even a relatively long proof cannot help in general for *every* property, one might ask whether there are specific properties for which short proofs do suffice. As was shown in Informal Theorem 3.1, this is indeed the case and a logarithmically long proof allows for an exponential improvement in the query complexity for a specific property. But can an even shorter, say constant-size proof, help? Unfortunately, the answer is negative since an MAP with query complexity q and proof complexity p can be emulated by a property tester that enumerates all possible proofs and makes a total of $\tilde{O}(2^p \cdot q)$ queries. Still, are there any further limits to how proofs can help a tester?

We first note that the ability to query the object in a way that depends on the proof is essential to the power of MAP. In contrast, consider *proof-oblivious queries* MAPs, which are MAPs in which the verifer's queries are independent of the provided proof. Such

[2] Note that the size of the tested object is N^2, and so $N \log_2 k$ is sublinear in the input size. In order to verify this proof, the verifier chooses $O(1/\varepsilon)$ edges at random and accepts if all are properly colored.

[3] In fact, we show a general additive tradeoff between proof and query complexities, that is, every MAP verifier that uses a proof of length p must make $\tilde{\Omega}(n - p)$ queries.

MAPs can be viewed as a two step process in which the verifier first (adaptively) queries the object and only then it receives the proof and decides whether to accept or reject based on both the answers and the proof. We say that such MAPs have proof oblivious queries. The following result shows that MAPs with *proof-oblivious queries* can provide at most a *quadratic* improvement over standard property testers.

Informal Theorem 3.5 (see Theorem 3.19)**.** *If a property* Π *has an* MAP *that makes* q proof oblivious *queries and uses a proof of length* p*, then* Π *has a property tester that makes* $O(q \cdot p)$ *queries.*

By Informal Theorem 3.1, the restriction to *proof oblivious queries* is a necessary precondition for Informal Theorem 3.5 (and indeed, the MAP verifier of Informal Theorem 3.1 must make proof-dependent queries).

Having inspected the relationship between MAPs and property testing, we proceed to consider the relationship between MAPs and IPPs. Recall that MAPs are actually a special case of IPPs in which the interaction is limited to a single message sent from the prover to the verifier. When comparing MAPs and IPPs it is natural to compare both the query complexity and the total amount of communication with the prover (which in the case of MAPs is simply the length of the proof).

The following theorem shows that IPPs are stronger than MAPs not only syntactically but also in essence. We show that even 3-message IPPs may have exponentially better query complexity than MAPs (while using the same amount of communication). Moreover, we show that IPPs with *poly-logarithmically* many messages of poly-logarithmic length can also have exponentially better communication complexity.

Informal Theorem 3.6 (see Theorem 3.16 and Theorem 3.17)**.** *There exists a property* Π *such that on the one hand, any* MAP *for* Π *with proof of length* $n^{0.499+o(1)}$ *has query complexity* $n^{0.499+o(1)}$*, and on the other hand,* Π *has:*

1. *A* 3*-message* IPP *that makes* $\mathsf{polylog}(n)$ *queries while using a total of* $n^{0.499+o(1)}$ *communication.*
2. *An* IPP *with only* $\mathsf{polylog}(n)$ *query and communication complexities but using a* poly-logarithmic *number of messages.*

3.1.4 Techniques

Several of our results (in particular Informal Theorems 3.2 and 3.6) are based on a specific algebraic property, which we call *Sub-Tensor Sum* and denote by TensorSum (c.f. [LFKN92]). Let $\mathbb{F}$ be a finite field and let $H \subset \mathbb{F}$ be an arbitrary subset. We consider m-variate polynomials over $\mathbb{F}$ that have individual degree d. The TensorSum property contains all such polynomials whose sum on H^m equals 0.[4] That is, TensorSum contains all polynomials $P : \mathbb{F}^m \to \mathbb{F}$ of individual degree d such that

$$\sum_{x \in H^m} P(x) = 0.$$

[4]The choice of the constant 0 is arbitrary.

Selecting $|\mathbb{F}|, m, d$ and $|H|$ suitably (as poly-logarithmic functions in the input size $n = |\mathbb{F}|^m$), we obtain the following roughly stated upper and lower bounds for TensorSum (for the formal statements, see the technical sections):

1. **PT**: The query complexity of testing TensorSum (without a proof) is $\Theta(n^{0.999\pm o(1)})$ queries.

2. **MAP**: The $\mathcal{MAP}$ complexity of TensorSum is $\Theta\left(n^{0.499\pm o(1)}\right)$. Moreover, for every $p \geq 1$, the MAP query complexity of TensorSum with respect to proofs of length p is $\Theta\left(\frac{n^{0.999\pm o(1)}}{p}\right)$.

3. **IPP[3]**: TensorSum has a 3-*message* IPP with query complexity $\mathsf{polylog}(n)$ and communication complexity $O\left(n^{0.499+o(1)}\right)$.

4. **IPP**: TensorSum has an IPP with query and communication complexities $\mathsf{polylog}(n)$. However, in contrast to Item 3, this IPP uses *poly-logarithmically* many messages.

To get a taste of our proofs, consider the (relatively) simple case wherein we restrict the TensorSum property to dimension $m = 2$ and a field $\mathbb{F}$ of size $\sqrt{n}$ (i.e., bivariate polynomials over a field of size $\sqrt{n}$). Naturally, we call this variant the *Sub-Matrix Sum* property and denote it by MatrixSum. Note that MatrixSum contains all polynomials $P : \mathbb{F}^2 \to \mathbb{F}$ of individual degree $d = |\mathbb{F}|/10$ such that

$$\sum_{x,y \in H} P(x, y) = 0.$$

As an MAP proof to the claim that the polynomial P is in MatrixSum, consider the univariate polynomial $Q(x) \stackrel{\text{def}}{=} \sum_{y \in H} P(x, y)$. To verify that P is indeed in MatrixSum the verifier acts as follows:

1. If $\sum_{x \in H} Q(x) \neq 0$, then reject.

2. Verify that P is (close to) a low degree polynomial and reject if not. This can be done with $O(d)$ queries via the classical low degree test (see Theorem 3.30).

3. Verify that Q is consistent with P. Since both are low degree polynomials, it suffices for the verifier to check that $Q(r) = \sum_{y \in H} P(r, h)$ for a random $r \in \mathbb{F}$.

   Actually, a technical difficulty arises from the fact that P can only be verified to be *close* to a low degree polynomial. The naive solution of reading every point via self-correction is too expensive in the case of MatrixSum. While it is possible to overcome this difficulty using a slightly more sophisticated technique (to appear in a forthcoming revision), the naive solution suffices for our actual setting of parameters (for TensorSum) and so we ignore this difficulty here.

By setting $|H| = O(|\mathbb{F}|)$ we obtain an MAP with proof and query complexity $O(\sqrt{n})$ (since $n = |\mathbb{F}|^2$). Using more sophisticated techniques in the same spirit, we obtain both MAP and IPP upper bounds for the TensorSum problem.[5]

Parameterized Concatenation Problems. Our techniques for showing MAPs for properties that do not have distance (and a structure that allows for self-correction) differ from the above. One class of problems that we consider is that of *parameterized concatenation problems*. Such properties consists of strings that are a concatenation of substrings, where each substring satisfies a particular parameterized property. The actual parameterization is not known a priori to the tester, and so an MAP proof that simply provides this parameterization turns out to be quite useful. Given this parameterization, the MAP verifier can simply test each substring individually (or a random subset of these substrings). Actually, in order to solve the problem more efficiently, the different substrings are tested with respect to different values of the proximity parameter by using a technique known as *precision sampling* (see survey [Gol14, Appendix A]).

Verifying Bipartiteness of Well-Mixing Graphs. Our MAP protocol for proving bipartiteness of a given *well-mixing* graph $G = (V, E)$ of size $N = |V|$ proceeds as follows. The proof consists of a subset $W \subseteq V$ of vertices that are allegedly on the same side of the graph. The verifier selects a random vertex $s \in V$ and takes roughly $N/|W|$ random walks of length $\Theta(\log n)$, starting at s. The verifier rejects if two of the walks pass through vertices of the set W, where the lengths of the paths from s to these vertices of W have opposite parities. Indeed, such walks cannot occur in bipartite graphs, assuming that all vertices in S are on the same side.

We show that if the graph is rapidly mixing and far from bipartite, then, for a $O(1/\log(N))$ fraction of vertices $s \in W$, the probability that a random walk starting in s will end in W with odd (respectively, even) parity is roughly $|W|/N$. Since the verifier takes $N/|W|$ random walks starting in s, with constant probability, it will detect a violation and reject. The analysis of our protocol is inspired by [GR02]. Interestingly, in contrast to the analysis of the rapidly-mixing case in [GR02], our analysis crucially relies on the random selection of the starting vertex.

Lower Bounds via MA Communication Complexity. As for our property testing *lower bounds*, we base these on the recently introduced technique of Blais, Brody and Metulef [BBM11]. The [BBM11] methodology enables one to obtain property testing lower bounds from *communication complexity* lower bounds. To obtain MAP lower bounds, we extend the [BBM11] framework. We show that lower bounds on the MA *communication complexity* of a communication complexity problem related to a property Π can be used to derive lower bounds on the MAP *complexity* of Π.

[5] We use TensorSum rather than MatrixSum because we do not know how to obtain an IPP nor a *full* trade-off between proof and query complexities for MatrixSum.

MA communication complexity, introduced by Babai, Frankl and Simon [BFS86], extends standard communication complexity by adding a third player, Merlin, who sees both the input x of Alice and y of Bob and attempts to convince them that $f(x, y) = 1$ where f is the function that they are trying to compute. We require that if $f(x, y)$ indeed equals 1, then there exist a proof for which Alice and Bob output the correct value (with high probability), but if $f(x, y) = 0$, then no proof will cause them to output a wrong value (except with some small error probability).

In order to show lower bounds for MAP we are thus left with the task of showing lower bounds for related MA communication complexity problems. Fortunately, Klauck [Kla03] showed a strong lower bound for the set-disjointness problem, which we use in our reductions. Additionally, we extend a recent result of Gur and Raz [GR13b] who give an MA communication complexity lower bound on the classical problem of *Gap Hamming Distance*.

We note that nearly all of the lower bounds shown in [BBM11] are proved via reductions from the communication complexity problems of *set-disjointness* and *gap Hamming distance*. Since these communication complexity problems have known MA communication complexity lower bounds (cf. [Kla03, GR13b]), these reductions, together with our extension of the [BBM11] framework to MAPs, gives MAP *lower bounds for the problems studied in [BBM11]* (e.g., testing juntas, Fourier degree, sparse polynomials, monotonicity, etc.).

Lower Bounds via the Probabilistic Method. Lastly, to prove Informal Theorem 3.4, which shows a property that requires $\Omega(n)$ queries even from an MAP that has access to a proof of length $n/100$, we use a technique that is inspired by [GGR98], and also uses ideas from [RVW13]. In more detail, we note that MAPs can be represented by a relatively small class of functions. Since this class of functions is small, using the probabilistic method, we argue that a "random property" (chosen from an adequate distribution) fools every MAP verifier in the sense that the verifier cannot distinguish between a random input that has the property and a totally random input (which will be far from the property).

3.1.5 Related Works

The notion of interactive proofs of proximity was first considered by Ergün, Kumar and Rubinfeld [EKR04] (where it was called approximate interactive proofs). More recently, Rothblum, Vadhan and Wigderson [RVW13] initiated a systematic study of the power of this notion. Their main result is that all languages in NC have interactive proofs of proximity with query and communication complexities roughly $\sqrt{n}$, and $\mathsf{polylog}(n)$ communication rounds. On the negative side, [RVW13] show that there exists a language in NC^1 for which the sum of queries and communication in any constant-round interactive proof of proximity must be polynomially related to n.

The study of interactive proofs-systems (in the polynomial-time setting), of which the class MA is a special case, was initiated in the seminal works of Goldwasser, Micali and

Rackoff [GMR89] and Babai [Bab85]. In the last decade, MA proof-systems were introduced for various computational models. There is a rich body of work in the literature addressing MA communication complexity protocols (e.g., [Kla03, GS10a, Kla11, She12]). Aaronson and Wigderson [AW09] used MA communication complexity lower bounds to show that, for many fundamental questions in complexity theory, any solution will require "non-algebraizing" techniques. In addition, in a recent line of research, the data stream model was extended to support several interactive and non-interactive proof systems. The model of streaming algorithms with non-interactive proofs was first introduced in [CCM09] and extended in [CMT13, GR13b, CCGT13]. Moreover, Cormode *et al.* [CMT12] have made a significant step toward a practical implementation of the generic interactive proof-system of Goldwasser *et al.* [GKR08] for delegation of data stream computation.

Relation to Partial Testing [FGL14]. Independently of this work, Fischer, Goldhirsh and Lachish [FGL14] introduced the notion of *partial testing*, which is closely related to MAPs. A property Π is a said to be Π'-partially testable, for $\Pi' \subseteq \Pi$, if inputs in Π' can be distinguished from inputs that are far from Π by a tester that makes only few queries. As pointed out by [FGL14], an $\mathsf{MAP}(p,q)$ for a property Π is equivalent to the existence of sub-properties $\Pi_1, \ldots, \Pi_{2^p} \subseteq \Pi$ such that $\cup_{i \in [2^p]} \Pi_i = \Pi$ and for every $i \in [2^p]$, the property Π is Π_i-partially testable using q queries.

In our terminology, the main result of [FGL14] is that there exists a (natural) property Π such that every $\mathsf{MAP}(p,q)$ for Π must satisfy that $p \cdot q = \Omega(n)$. In contrast, Informal Theorem 3.2 shows a different property Π' for which $p \cdot q = \Omega(n^{0.999})$. However, we also show an (almost) matching *upper* bound for our property Π' (see Informal Theorem 3.2). We also note that Informal Theorem 3.4 (see Theorem 3.22), which was discovered following the publication of [FGL14], shows a property for which every $\mathsf{MAP}(p,q)$ must satisfy $p + q = \Omega(n)$; that is, if $p = n/100$, then $q = \Omega(n)$. We note that the latter result also resolves (a natural interpretation of) a question asked by [FGL14, Open Question 1.4].[6]

Applications of our Work and Follow-Up Works. Our work has also found applications in unrelated studies. For example, in the study of *sample-based testers*, Goldreich and Ron [GR15a] used the separation between the power of MAPs and property testers (see Theorem 3.7) in order to show that proximity-oblivious testers do not necessarily imply *fair* proximity-oblivious testers (where fair proximity-oblivious testers are such in which every query is almost uniformly distributed). Another example is an application for *testing dynamic environments*. Specifically, the separation between the power of standard MAPs and MAPs with *proof-oblivious queries* (see Lemma 3.6 and Theorem 3.19) was used to show that time-conforming testers can be exponentially weaker than their non-time-conforming counterparts (see [GR14] for details). In addition, following the publication of this work, Goldreich, Gur, and Komargodski [GGK14] improved on Informal

[6] Loosely speaking, in the terminology of [FGL14], Theorem 3.22 implies that for every r there exists a property Π that can be tested with r queries, but every partition of Π into k properties $\Pi_1, \ldots, \Pi_k$, such that Π is P_i-partially testable with $O(1)$ queries, must satisfy that $k = 2^{\Omega(r)}$.

Theorem 3.1 by tightening the separation between MAPs and testers (see Section 3.3.1 for more details).

Non-Deterministic Testing of Graphs Last, we note that Alon *et al.* [AFNS06] discussed the notion of *non-deterministic property testing of graphs*, which was formally stated recently by Lovász and Vesztergombi [LV12], and further studied by Gishboliner and Shapira *et al.* [GS13]. This model is a form of PCP of proximity in which both the proof and verification procedure are restricted to be of a particular form.

3.1.6 Organization

This paper's organization differs from the order in which our results were reviewed in the introduction, so that technically related results are grouped together. In Section 3.2 we formally define MAPs and property testers (which are essentially MAPs with an empty string). In Section 3.3 we formally state and prove all of our separation results, whereas in Section 3.4 we prove our general transformation results. In Section 3.5 we show a property that is hard for MAPs even given a (relatively) long proof. In Section 3.6 we consider MAPs for concatenation problems and in Section 3.7 we show our MAP for verifying bipartiteness of rapidly-mixing graphs in the bounded degree model. Important background material is provided in Section 3.A.

3.2 Definitions

In this section we formally define Merlin-Arthur proofs of proximity. We start by introducing some relevant notations and standard definitions.

A property may be defined as a set of strings. However, since we mostly consider properties that consist of (non-Boolean) functions, it will be useful for us to use the following (also commonly used) equivalent definition.

For every $n \in \mathbb{N}$, let D_n and R_n be sets. For simplicity we use the convention that $D_n = [n]$ (and R_n will usually be of size much smaller than n). Let $\mathcal{F}_n$ be the set of all functions from D_n to R_n. A property is an ensemble $\Pi = \cup_{n\in\mathbb{N}} \Pi_n$, where $\Pi_n \subseteq \mathcal{F}_n$. In the (rare) case that we test properties of strings (rather than functions), we view the n-bit string x as a function $I_x : [n] \to \{0,1\}$ where $I_x(i) = x_i$ for all $i \in [n]$. For the rest of this work, it will sometimes be convenient for us to refer to Π as a problem (rather than a property), where we actually refer to the testing problems that are associated with Π (and are defined in the following subsections).

Let $x, y \in \Sigma^n$ be two strings of length $n \in \mathbb{N}$ over a (finite) alphabet Σ. We define the (absolute) distance of x and y as $\Delta(x,y) \stackrel{\text{def}}{=} |\{x_i \neq y_i : \ i \in [n]\}|$. If $\Delta(x,y) \leq \varepsilon \cdot n$, then we say that x is ε-close to y, and otherwise we say that x is ε-far from y. We define the distance of x from a set $S \subseteq \Sigma^n$ as $\Delta(x,S) \stackrel{\text{def}}{=} \min_{y\in S} \Delta(x,y)$. If $\Delta(x,S) \leq \varepsilon \cdot n$, then we say that x is ε-close to S and otherwise we say that x is ε-far from S. We extend these definitions from strings to functions, while identifying a function with its truth table.

Notation. For a finite set S, we denote by $x \in_R S$ a random variable x that is uniformly distributed in S. We denote by $A^f(x)$ the output of algorithm A given an explicit input x and implicit (i.e., oracle) access to the function f. Last, given a binary string s, we denote its Hamming weight by $\mathsf{wt}(x)$.

Integrality Issues. Throughout this work, for simplicity of notation, we use the convention that all (relevant) integer parameters that are stated as real numbers are implicitly rounded to the nearest integer.

3.2.1 Merlin-Arthur Proofs of Proximity

We are now ready to define Merlin-Arthur proofs of proximity.

Definition 3.1. *A* Merlin-Arthur proof of proximity (MAP) *for a property $\Pi = \cup_{n\in\mathbb{N}}\Pi_n$ consists of a probabilistic algorithm V, called the verifier, that is given as explicit inputs an integer $n \in \mathbb{N}$, a proximity parameter $\varepsilon > 0$, and a proof string $w \in \{0,1\}^*$; in addition, it is given oracle access to a function $f \in \mathcal{F}_n$. The verifier satisfies the following two conditions:*

1. Completeness: *For every $n \in \mathbb{N}$ and $f \in \Pi_n$, there exists a string w (referred to as a* proof *or* witness*) such that for every proximity parameter $\varepsilon > 0$:*

$$\Pr\left[V^f(n,\varepsilon,w) = 1\right] \geq 2/3.$$

   where the probability is over the random coin tosses of the verifier V.

2. Soundness: *For every $n \in \mathbb{N}$, function $f \in F_n$, string w, and proximity parameter $\varepsilon > 0$, if f is ε-far from Π_n, then:*

$$\Pr\left[V^f(n,\varepsilon,w) = 1\right] \leq 1/3.$$

   where the probability is over the random coin tosses of the verifier V.

If the completeness condition holds with probability 1, then we say that the MAP *has a* one-sided error *and otherwise we say that it has* two-sided error.

We note that MAPs can be viewed as a restricted form of the *interactive* proofs of proximity, studied by [RVW13] (see Section 3.2.2 for the definition of IPP).

An MAP is said to have query complexity $q : \mathbb{N} \times \mathbb{R}^+ \to \mathbb{N}$ if for every $n \in \mathbb{N}$, $\varepsilon > 0$, $f \in \mathcal{F}_n$ and any $w \in \{0,1\}^*$, the verifier makes at most $q(n,\varepsilon)$ queries to f. The MAP is said to have proof complexity $p : \mathbb{N} \to \mathbb{N}$ if for every $n \in \mathbb{N}$ and $f \in \Pi_n$ there exists $w \in \{0,1\}^{p(n)}$ for which the completeness condition holds.[7] If the MAP has query complexity q and proof complexity p, we say that it has complexity $t(n,\varepsilon) \stackrel{\text{def}}{=} q(n,\varepsilon)+p(n)$.

[7] Without loss of generality, using adequate padding, we assume that there is a fixed proof length $p(n)$ for objects of size n. The latter can be complemented by restricting the soundness condition to hold only for strings of length $p(n)$ (rather than strings of arbitrary length), since the verifier can immediately reject proofs that have length that is not $p(n)$.

For every pair of functions $q : \mathbb{N} \times \mathbb{R}^+ \to \mathbb{N}$ and $p : \mathbb{N} \to \mathbb{N}$, we denote by $\mathsf{MAP}_2(p, q)$ (resp., $\mathsf{MAP}_1(p, q)$) the complexity class of all properties that have an MAP with proof complexity $O(p)$, query complexity $O(q)$ and two-sided error (resp., one-sided error). We also use MAP as a shorthand for the class MAP_2.

Note that we defined MAPs such that the proofs do not depend on the proximity parameter ε. Since our focus is on demonstrating the power of MAPs (and our lower bounds refer to fixed valued of the proximity parameter), this makes our results stronger. Nevertheless, see Section 3.2.1 for a discussion of the alternate notion, in which the proof *may* depend on the proximity parameter.

Proof oblivious queries. An aspect of MAP proof-systems, which turns out to be very important, is whether the queries that the verifier makes depend on the proof. An MAP in which the queries *do not depend on the proof* may be thought of as the following two step process:

1. The verifier is given oracle access to the object being tested. The verifier's queries may be adaptively generated (based on answers to previous queries).

2. After getting answers to all of its queries, the verifier is given explicit and free access to the proof string (which is chosen obliviously of the verifier's queries). Based on the queries, answers and the proof, the verifier decides whether to accept or reject.

The foregoing discussion gives rise to the following definition.

Definition 3.2. *An* MAP *verifier for a property* $\Pi \subseteq \{F_n\}_n$ *is said to make* proof oblivious queries *if for every* $n \in \mathbb{N}$*, function* $f \in F_n$*, proximity parameter* $\varepsilon > 0$*, random string* r *and two proof string* $w, w' \in \{0,1\}^*$*, the* MAP *verifier, given oracle access to* f*, the random string* r *and explicit access to* n, ε*, and given either the proof string* w *or* w'*, makes the same sequence of queries.*

MA proximity-oblivious testing. We also present an MA version of *proximity-oblivious testing* (defined in [GR11]). Loosely speaking, a proximity-oblivious tester (POT) is a testing algorithm that satisfies the following conditions: (1) it is oblivious of the proximity parameter ε (i.e., it does not get ε as part of its input) and (2) it rejects statements that are ε-far from true statements with probability that is some increasing function of ε. A standard property tester can be obtained by repeating the POT sufficiently many times.

We give a definition of *one-sided error* MA proximity-oblivious testers, and note that a *two-sided error* variant of MA proximity-oblivious testers can be defined similarly to [GS12].

Definition 3.3. *Let* $\rho : (0,1] \to (0,1]$ *be some increasing function. A (one-sided error)* MA proximity-oblivious tester *for a property* $\Pi = \cup_{i \in \mathbb{N}} \Pi_n$ *with* detection probability ρ *consists of a probabilistic verifier* V *that is given as explicit inputs an integer* $n \in \mathbb{N}$ *and a proof string* $w \in \{0,1\}^*$*, and is given oracle access to a function* $f \in \mathcal{F}_n$*. The verifier satisfies the following two conditions:*

1. Completeness: *For every $n \in \mathbb{N}$ and $f \in \Pi_n$, there exists a proof w such that:*
$$\Pr\left[V^f(n,w) = 1\right] = 1.$$
2. Soundness: *For every $n \in \mathbb{N}$, function $f \in F_n$, and proof w, if f is ε-far from Π_n, then:*
$$\Pr\left[V^f(n,w) = 0\right] \geq \rho(\varepsilon).$$

(In both conditions the probability is over the random coin tosses of the verifier V.)

We remark that a few of the MAPs presented in this work are based on corresponding MA proximity-oblivious testers. The most notable example is the MAP in Theorem 3.9.

MAPs with Proximity-Dependent Proofs We defined the notion of MAPs such that the proof of proximity is *oblivious* of the proximity parameter ε. However, it is also natural to consider a relaxation of MAPs wherein the proof of proximity *may* depend on the proximity parameter. In fact, one can consider two levels of relaxation: (1) the content of the proof *but not its length* may depend on the proximity parameter, and (2) both the contents and the length of the proof may depend on the proximity parameter. We note that the first possibility is almost equivalent to the standard definition of MAP, since it always suffices to refer to only a logarithmic number of values of ε (i.e., $\varepsilon = 2^i$ for all $i \in [\log n]$), and concatenate the proofs for these values, thus obtaining a standard MAP with only a logarithmic overhead to the proof complexity.

Property Testing The standard definition of property testing may be derived from Definition 3.1 by restricting both the completeness and soundness conditions to hold when the proof length is fixed to 0. Hence, MAPs are a strict syntactic generalization of property testers. We will always refer to a tester that uses a proof as an "MAP verifier" and reserve "tester" solely for (standard) property testers that *do not use a proof*.

For a property Π and a proximity parameter $\varepsilon > 0$, we denote by $\mathsf{PT}_\varepsilon(\Pi)$ the minimum, over all testers T for Π, of the query complexity of T with respect to proximity ε. For every function $q : \mathbb{N} \times \mathbb{R}^+ \to \mathbb{N}$, we denote by $\mathsf{PT}_2(q)$ (resp., $\mathsf{PT}_1(q)$) the class $\mathsf{MAP}_2(0,q)$ (resp., $\mathsf{MAP}_1(0,q)$). We also use PT as a shorthand for the class PT_2.

For a detailed introduction to property testing, see the surveys [Ron08, Ron09] and the collection [Gol10a].

3.2.2 Interactive Proofs of Proximity

In this section we define *interactive proofs of proximity*, following Rothblum *et al.* [RVW13].[8] For two interactive algorithms A and B, we denote by $(A^f, B^f)(x)$ the output of (say) A when interacting with B when both algorithms are given x as an explicit input and implicit (i.e., oracle) access to the function f.

[8] Our definition of IPP slightly differs from that of [RVW13] in that they consider the absolute distance of objects from the property rather relative distance. (Needless to say, we take this into account when discussing their results.)

Definition 3.4. *An* interactive proof of proximity *system (*IPP*) for a property* Π *is an interactive protocol with two parties: a (computationally unbounded) prover* $\mathcal{P}$ *and a verifier* $\mathcal{V}$*, which is a probabilistic algorithm. The parties send messages to each other, and at the end of the communication, the following two conditions are satisfied:*

1. Completeness: *For every* $\varepsilon > 0$*,* $n \in \mathbb{N}$*, and* $f \in \Pi_n$ *it holds that,*

$$\Pr\left[(\mathcal{V}^f, \mathcal{P}^f)(n, \varepsilon) = 1\right] \geq 2/3.$$

   where the probability is over the coin tosses of $\mathcal{V}$*.*

2. Soundness: *For every* $\varepsilon > 0$*,* $n \in \mathbb{N}$*,* $f \in \mathcal{F}_n$ *that is* ε*-far from* Π_n *and for every computationally unbounded (cheating) prover* $\mathcal{P}^*$ *it holds that*

$$\Pr\left[(\mathcal{V}^f, \mathcal{P}^*)(n, \varepsilon) = 1\right] \leq 1/3.$$

   where the probability is over the coin tosses of $\mathcal{V}$*.*

If the completeness condition holds with probability 1, then we say that the IPP *has a* one-sided error *and otherwise the* IPP *is said to have a* two-sided error*.*

An IPP is said to have query complexity $q : \mathbb{N} \times \mathbb{R}^+ \to \mathbb{N}$ if for every $n \in \mathbb{N}$, $\varepsilon > 0$, $f \in \mathcal{F}_n$ and any prover strategy $\mathcal{P}^*$, the verifier makes at most $q(n, \varepsilon)$ queries to f when interacting with $\mathcal{P}^*$. The IPP is said to have communication complexity $c : \mathbb{N} \times \mathbb{R}^+ \to \mathbb{N}$ if for every $n \in \mathbb{N}$, $\varepsilon > 0$ and $f \in \Pi_n$ the communication between $\mathcal{V}$ and $\mathcal{P}$ consists of at most $c(n, \varepsilon)$ bits. If the IPP has query complexity q and communication complexity c, we say that it has IPP complexity $q + c$.

For every pair of functions $c, q : \mathbb{N} \times \mathbb{R}^+ \to \mathbb{N}$, we denote by $\mathsf{IPP}_2(c, q)$ (resp., $\mathsf{IPP}_1(c, q)$) the complexity class of all properties that have an IPP with communication complexity $O(c)$, query complexity $O(q)$ and two-sided error (resp., one-sided error). We also use IPP as a shorthand for the class IPP_2.

An important parameter of an IPP is the number of messages m sent between the two parties. We denote by $\mathsf{IPP}[m](c, q)$ the set of properties that have m-message IPP protocols in which the verifier uses at most $O(c)$ bits of communication, and makes at most $O(q)$ oracles queries.

3.2.3 Useful Conventions

The proximity parameter. We view the proximity parameter as a function $\varepsilon = \varepsilon(n)$. For simplicity we assume that $\varepsilon(n)$ is a non-increasing function.

Our definition of MAPs requires that soundness hold with respect to *every* value of $\varepsilon > 0$. However, throughout this work we sometimes find it convenient to restrict the proximity to $\varepsilon \in (0, \varepsilon_0)$ for some constant $\varepsilon_0 \in (0, 1)$. We note that latter type of MAPs can be extended to the more general form by simply running the base tester with respect to proximity $\varepsilon' = \min(\varepsilon, \varepsilon_0)$ (incurring only a constant overhead).

Implicit input length and proximity parameter. Throughout this work, for simplicity of notation, we use the convention that the input length n and proximity parameter ε are given *implicitly* to all testers and verifiers (e.g., when we write T^f we actually mean $T^f(n, \varepsilon)$).

3.3 Separation Results

In this section we explore the power of MAP verifiers in comparison to other types of testers, such as property testers and IPP verifiers and present properties that exhibit a separation between these different types of testers.

In Section 3.3.1 we show an exponential gap between the complexity of PT and MAP. In Section 3.3.2 we show a problem that has an MAP with an (almost) tight multiplicative tradeoff between the proof length and number of queries. In Section 3.3.3 we consider 3-*message* IPP verifiers and show that they may have exponentially smaller *query* complexity than MAP verifiers (when using a proof of similar length). Finally, in Section 3.3.4 we also show an exponential gap between the total complexity (i.e., query plus proof/communication complexities) of MAP and general IPP (which uses a poly-logarithmic number of messages).

3.3.1 Exponential Separation between PT and MAP

In this section we show an *exponential* separation between the power of property testing and MAP. Roughly speaking, we show a property that requires roughly $n^{0.999}$ queries for every property tester but has an MAP that, while using a proof of only *logarithmic* length, requires only a *constant* number of queries. We prove the following incomparable variants of this result.

Theorem 3.7. *For every constant $\alpha > 0$, there exists a property Π_α that has an* MAP *that uses a proof of length $O(\log n)$ and makes $\mathsf{poly}(1/\varepsilon)$ queries for every $\varepsilon > 1/\mathsf{polylog}(n)$, but for which every* property tester *must make $\Omega(n^{1-\alpha})$ queries. Furthermore, the* MAP *has one-sided error.*

A limitation of the foregoing theorem is that the proximity parameter is required to be larger than $1/\mathsf{polylog}(n)$. We also consider two incomparable variants of Theorem 3.7 that let us handle general values of ε. In Theorem 3.8 we do so but at the cost of increasing the MAP query complexity to depend poly-logarithmically on n.

Theorem 3.8. *For every constant $\alpha > 0$, there exists a property Π_α that has an* MAP *that uses a proof of length $O(\log n)$ and makes $\mathsf{poly}(\log n, 1/\varepsilon)$ queries, but for which every* property tester *must make $\Omega(n^{1-\alpha})$ queries. Furthermore, the* MAP *has one-sided error.*

The above separation results refer to the general (i.e., two-sided error) classes PT_2 and MAP_2. As noted in the introduction, a more restricted separation between the *one-sided error* classes (i.e., between PT_1 and MAP_1) can be obtained by using Theorem 3.20. We

remark that the preliminary technical report [GR13c] also contained a proof of the following (incomparable) variant, which can handle all values of the proximity parameter while using $\mathsf{poly}(1/\varepsilon)$ query complexity, at the cost of having a smaller (yet still exponential) gap between the power of property testers and MAPs.

Theorem 3.9 ([GR13c])**.** *There exists a universal constant $c \in (0,1)$ and a property Π that has an* MAP *that uses a proof of length $O(\log n)$ and makes $\mathsf{poly}(1/\varepsilon)$ queries (without limitation on ε), but for which every* property tester *must make n^c queries. Furthermore, the* MAP *has one-sided error.*[9]

A different proof of Theorem 3.9 is sketched in [FGL14] who, using a result of Alon *et al.* [AKNS00], showed a property that requires $\Omega(\sqrt{n})$ queries (without a proof) but can be tested using only $O(1/\varepsilon)$ queries and a proof of length $O(\log n)$.

Follow-Up Work. Following the publication of this work, Goldreich, Gur, and Komargodski [GGK14] improved the separation between MAPs and testers, achieving the best of Theorems 3.7 to 3.9 simultaneously; that is, they obtain a separation for all values of the proximity parameter, with constant query complexity for the MAPs, and nearly-linear query complexity for testers.

Theorem 3.10 ([GGK14])**.** *For every constant $\alpha > 0$, there a property Π_α that has an* MAP *that uses a proof of length $O(\log n)$ and makes $\mathsf{poly}(1/\varepsilon)$ queries (without limitation on ε), but for which every* property tester *must make $n^{1-\alpha}$ queries. Furthermore, the* MAP *has one-sided error.*

In the next subsections we will show two lemmas (Lemmas 3.5 and 3.6) that allow us to reduce the problem of separating the power of MAPs and testers to the problem of designing error-correcting codes that are both locally testable and locally decodable. Theorems 3.7 to 3.10 are then obtained by instantiating Lemmas 3.5 and 3.6 with such codes. Since the codes of [GGK14] improve upon the codes that are used to obtain Theorems 3.7 to 3.9, we omit the more involved proof of Theorem 3.9, which consists of a construction of a code with the desired properties (see technical report [GR13c] for details and proof). We provide the proofs of Theorems 3.7 and 3.8, which are instantiations of Lemmas 3.5 and 3.6 for known codes.

3.3.1.1 Our Approach

The proof of Theorem 3.7 is heavily based on error correcting codes. Recall that a code is an injective function $C : \Sigma^k \to \Sigma^n$ over an alphabet Σ. The relative distance of the code is the minimal relative distance between every two (distinct) codewords, and the stretch of the code is n when viewed as a function of k. Further necessary background is provided in Section 3.A.3.

[9] We remark that the proof of Theorem 3.9 can be adapted to yield an MA *proximity-oblivious tester* (see Definition 3.3) for Π.

As discussed in the introduction, the complexities of property testers and MAP verifiers with *proof oblivious queries* are polynomially related (see Theorem 3.19). Thus, in order to show an *exponential* separation between PT and MAP, one has to use an MAP for which the queries inherently depend on the proof. That is, the property Π should satisfy the following:

1. Π can be efficiently verified by an MAP in which the queries are "strongly affected" by the proof;
2. Π is hard for property testers (and hence for MAPs with proof oblivious queries).

Thus, intuitively, we seek a property that is based on a "hidden structure" that can be tested locally if one knows where to look but cannot be tested locally otherwise.

As a first (naive) candidate, consider the property containing the set of all non-zero strings. A short proof for this property could direct us to the exact location of a non-zero bit, which can then be verified by a single query. However, the aforementioned property is (almost) trivial — as all strings are close to a string with a non-zero bit. Hence, we seek a robust version of this property.

This naturally leads us to consider an encoded version of the foregoing naive property. Fix an error-correcting code C and consider the property that contains all codewords that encode non-zero strings. Assuming that the code is both locally testable and locally decodable (i.e., both an LTC and an LDC, see Section 3.A.3), it is easy to test this property using an MAP that simply specifies a non-zero coordinate of the encoded message. However, this property may also be easy to test without a proof since all one needs to do is test that the string is not the (single) encoding of the zero message but is (close to) a codeword.

To overcome this difficulty, we consider a "twist" of the foregoing property in which we consider two codewords that must be non-zero on the same coordinate. That is, for every code C, we define the encoded intersecting messages property, denoted by EIM_C as:

$$\mathsf{EIM}_C \stackrel{\text{def}}{=} \left\{ \big(C(x), C(y)\big) \;:\; x, y \in \Sigma^k,\ k \in \mathbb{N} \text{ and } \exists i \in [k] \text{ s.t. } x_i \neq 0 \text{ and } y_i \neq 0 \right\},$$

where we assume that $0 \in \Sigma$. We note that we could have slightly modified our definition by requiring that $x_i = y_i = 1$ (where the choice of 1 is arbitrary) rather than $x_i, y_i \neq 0$. Another notable variant is obtained by requiring that $\Sigma = \{0, 1\}$; then the property EIM_C contains all pairs of codewords whose corresponding encoded messages (viewed as sets) intersect (i.e., are not disjoint).

For the lower bound, we only require that C have constant relative distance and the quality of the lower bound is directly related to the stretch of the code. For the upper bound, in addition to the constant relative distance, we need C to be both an LTC and an LDC with small query complexities. Indeed, the query complexity of the MAP that we construct is proportional to the number of queries required by the LTC and LDC procedures.

It is well-known that (a suitable instantiation of) the Reed-Muller code is both an LTC and LDC with $\mathsf{polylog}(n)$ query complexities, and almost linear stretch. By instantiating

EIM with this code, we can obtain Theorem 3.8; namely, a property that has an MAP with a proof of length $O(\log n)$ and $\mathsf{polylog}(n)$ query complexity, but requires an almost linear number of queries by any (standard) property tester.

In order to obtain a result with *constant* MAP query complexity (as in Theorem 3.7), we need a code that is both an LTC and an LDC, with constant query complexities. While LTCs with constant query complexity (and almost linear stretch) are known, constructing LDCs with constant query complexity (and polynomial stretch) is a major open problem in the theory of computation. However, we observe that for our construction it actually suffices that C be a *relaxed*-LDC. Relaxed-LDCs, introduced by Ben-Sasson *et al.* [BSGH+06], are a weaker form of LDCs in which the decoder is allowed to output a special abort symbol $\perp$ in case it is unable to decode a corrupt codeword. However, the decoder is not allowed to abort when given as input a correct codeword. We refer the reader to Definition 3.27 for the formal definition.

Ben-Sasson *et al.* [BSGH+06] used PCPPs to construct an $O(1)$-relaxed-LDC with almost linear stretch. Furthermore, [BSGH+06] argue that their relaxed-LDC is also a $\mathsf{poly}(1/\varepsilon)$-LTC. However, the LTC property only holds for proximity parameter $\varepsilon > 1/\mathsf{polylog}(n)$. Thus, using the [BSGH+06] code, we (only) obtain Theorem 3.7. In addition, by combining ideas and results of [BSGH+06] and [GS06] we construct an $O(1)$-relaxed-LDC that is also a $\mathsf{poly}(1/\varepsilon)$-LTC *for general values of* $\varepsilon > 0$, albeit with polynomial (rather than almost linear) stretch. Using the latter result, which may be of independent interest, we obtain Theorem 3.9.

Organization. In Section 3.3.1.2 we show that for every code $C : \Sigma^k \to \Sigma^n$ that is a t_1-relaxed-LDC and a t_2-LTC, it holds that $\mathsf{EIM}_C \in \mathsf{MAP}\big(\log k, t_1(n/2) + t_2(n/2, \varepsilon/2)\big)$. In Section 3.3.1.3 we show an $\Omega(k/\log|\Sigma|)$ lower bound on the query complexity of testing EIM_C (without a proof of proximity). In Section 3.3.1.4 we state the result of [BSGH+06] and derive Theorem 3.7, and in Section 3.3.1.5 we prove Theorem 3.8 using an appropriate instantiation of the Reed-Muller code.

3.3.1.2 An MAP Upper Bound for EIM

Lemma 3.5. *Let $C : \Sigma^k \to \Sigma^n$ be a code with constant relative distance that is a t_1-relaxed-LDC and also a t_2-LTC. Then, $\mathsf{EIM}_C \in \mathsf{MAP}_1\big(\log k, t_1(n/2) + t_2(n/2, \varepsilon/2)\big)$.*

Proof. We prove Lemma 3.5 by showing an MAP proof-system for proving proximity to EIM_C. The proof of proximity for the statement $(C(x), C(y)) \in \mathsf{EIM}_C$ is simply a coordinate $i \in [k]$ such that the messages x and y are non-zero i (i.e., $x_i, y_i \neq 0$). Given the proof i and oracle access to a pair of strings (α, β), it suffices for the verifier to check that both α and β are close to codewords (using the LTC property) and if so to reconstruct the i^{th} symbol of the underlying messages (using the relaxed-LDC property). (Lastly, it verifies that both symbols are non zero.)

The full protocol is described in Fig. 3.1, where $\delta_0 \in (0, 1)$ denotes the relative distance of C, and $\delta \in (0, \delta_0/2)$ denotes the decoding radius of C (i.e., strings that are δ-close to codewords are correctly decoded by the relaxed-LDC procedure).

MAP for EIM_C (where $C : \Sigma^k \to \Sigma^n$ is a t_1-relaxed-LDC and a t_2-LTC)

Input: a proximity parameter $\varepsilon \in (0, 2\delta)$ (where δ is the decoding radius) and oracle access to a pair $(\alpha, \beta) \in \Sigma^{n+n}$.

The Proof:

- Let $x, y \in \Sigma^k$ be the unique messages encoded in α and β, respectively; that is, $C(x) = \alpha$ and $C(y) = \beta$. Denote the i^{th} symbol of x by x_i, and the i^{th} symbol of y by y_i.
- The proof consists of a coordinate $i \in [k]$ such that $x_i \neq 0$ and $y_i \neq 0$ (which exists, for $(\alpha, \beta) \in \mathsf{EIM}_C$).

The Verifier:

1. Run the local *testing* algorithm of C on α and on β with respect to proximity parameter $\varepsilon/2$ and reject if either test rejects.
2. Run the (relaxed) local *decoding* algorithm of C to obtain the i^{th} message symbol of α, denoted σ, and the i^{th} message symbol of β, denoted τ.
3. Accept if both $\sigma \neq 0$ and $\tau \neq 0$, and reject otherwise.

Figure 3.1: MAP for EIM_C

Since the code is a t_1-relaxed-LDC and a t_2-LTC, the query complexity of the MAP is $2t_1(n/2) + 2t_2(n/2, \varepsilon/2)$, and the proof complexity is $\log_2 k$. We proceed to show that both completeness and soundness hold.

Completeness. If $(\alpha, \beta) \in \mathsf{EIM}_c$, then there exist $x, y \in \Sigma^k$ such that $\alpha = C(x)$ and $\beta = C(y)$, and therefore the local testing algorithm succeeds. Since the proof consists of a coordinate i for which $x_i, y_i \neq 0$, and the local decoding algorithm always succeeds, the MAP verifier always accepts.

Soundness. Suppose that (α, β) is ε-far from EIM_C and let $i \in [k]$ be some alleged proof to the false statement $(\alpha, \beta) \in \mathsf{EIM}_C$. There are two possible scenarios to consider:

1. either α or β are $\varepsilon/2$-far from C; or
2. both α and β are $\varepsilon/2$-close to C.

In the first case, with probability at least $1/2$, the local testing algorithm will fail and therefore the MAP verifier rejects with probability at least $1/2$. We proceed to the second case.

Suppose that both α and β are $\varepsilon/2$-close to the code. Then, there exist unique $x, y \in \Sigma^k$ s.t. α is $\varepsilon/2$-close to $C(x)$ and β is $\varepsilon/2$-close to $C(y)$, where uniqueness holds

since $\varepsilon/2 < \delta < \delta_0/2$. However, since (α, β) is ε-far from having the property EIM_C, this implies that either $x_i = 0$ or $y_i = 0$ (where i is the alleged proof). Thus, when running the relaxed local decoding algorithm (since $\varepsilon/2 < \delta$), with probability at least $2/3$, the decoder will output either 0 or $\perp$ on one of the two codewords (with respect to coordinate i), in which case the verifier rejects. We conclude that in both scenarios the verifier rejects with probability at least $1/2$. □

3.3.1.3 A PT Lower Bound for EIM

Next, we show a that the query complexity of property testing the EIM property must be linear in k.

Lemma 3.6. *Let $C : \Sigma^k \to \Sigma^n$ be an error-correcting code with relative distance at least $\delta_0 \in (0, 1)$. Then, for any $\varepsilon \in (0, \delta_0/2)$ it holds that:*

$$\mathsf{PT}_\varepsilon\big(\mathsf{EIM}_C\big) = \Omega(k/\log|\Sigma|)$$

The proof of Lemma 3.6 uses the framework of [BBM11] for showing property testing lower bounds via communication complexity lower bounds. The necessary background on communication complexity is provided in Section 3.A.1 (for a comprehensive introduction to communication complexity, see [KN97]).

The basic approach of [BBM11] is to reduce a hard communication complexity problem to the property testing problem for which we want to show a lower bound. We follow [BBM11] by showing a reduction from the well-known communication complexity problem of *set-disjointness*. The aforementioned framework allows us to obtain a lower bound on the query complexity of testing the *encoded intersecting messages* property.

For sake of self containment, we state the relevant definitions and lemmas that we need from [BBM11].

Definition 3.7 (Combining operators)**.** *A* combining operator *is an operator ψ that takes as input two functions $f, g : D \to R$ (where D and R are some finite sets) and returns a function $h_{f,g}$. We denote by $|\psi| \stackrel{\text{def}}{=} \log_2|R|$. The combining operator is called* simple *if $h_{f,g}(x)$ can be computed from $x, f(x)$ and $g(x)$ (i.e., without requiring access to f and g).*

Let Π be a property, and let ψ be a combining operator. For every integer $n \in \mathbb{N}$ and proximity parameter $\varepsilon > 0$, we denote by $\mathcal{C}^{\Pi}_{\psi,\varepsilon}$ the communication complexity problem wherein Alice gets a function f, and Bob gets a function g,[10] and their goal is to decide whether $\psi(f, g) \in \Pi$ or $\psi(f, g)$ is ε-far from Π.[11] Next, we state the main lemma from [BBM11].

[10] More formally, the parties get as input strings that represent the truth table of the functions.

[11] Due to the symmetrical definition of the communication complexity model, it is unimportant which of these cases (i.e., $\psi \in \Pi$ or ψ that is ε-far from Π) is viewed as a YES-instance of Π. In contrast, see Footnote 13.

Lemma 3.8. *For any* simple *combining operator* ψ*, any property* Π *and any proximity parameter* $\varepsilon > 0$*, we have that:*

$$\mathsf{PT}_{\varepsilon}(\Pi) \geq \frac{\mathsf{CC}(\mathcal{C}^{\Pi}_{\psi,\varepsilon})}{2|\psi|}$$

where $\mathsf{PT}_{\varepsilon}(\Pi)$ *refers to the query complexity of the property* Π *with respect to proximity* ε *and* $\mathsf{CC}(\mathcal{C})$ *refers to the communication complexity of* $\mathcal{C}$ *(see Section 3.A.1).*

Recall that the set-disjointness problem is the communication complexity problem wherein Alice gets an n-bit string x, Bob gets an n-bit string y, and their goal is to decide whether there exists $i \in [n]$ such that $x_i = y_i = 1$. Equivalently, Alice and Bob's inputs can be viewed as indicator vectors of sets $A, B \subseteq [n]$. In this case, the goal of the players is to decide if the sets corresponding to their inputs intersect or not. Following many works in the literature we consider the promise problem (sometimes also called *unique* disjointness) in which the intersection is of size at most 1. That is, the two parties need to distinguish between the case that their intersection is empty, and the case that it is of size exactly 1. We denote the latter problem by DISJ_n.

It is well-known (see Section 3.A.1) that the randomized communication complexity of the *set-disjointness* problem is linear in the size of the inputs, even under the promise that A and B intersect in at most one element.

Theorem 3.11 ([KS92])**.** *For every* $n \in \mathbb{N}$*,*

$$\mathsf{CC}(\mathsf{DISJ}_n) = \Omega(n).$$

Using the aforementioned results, we are ready to prove Lemma 3.6.

Proof of Lemma 3.6. Let $C : \Sigma^k \to \Sigma^n$ be an error-correcting code with relative distance $\delta_0 \in (0,1)$ where we assume without loss of generality that $\{0,1\} \subseteq \Sigma$. Denote by Pair the operator that takes two strings $x, y \in \Sigma^k$ and returns a function $z : [k] \to \Sigma$ that outputs (x_i, y_i) on input $i \in [k]$. Consider $\mathcal{C}^{\mathsf{EIM}_C}_{\mathsf{Pair},\varepsilon}$, the communication complexity problem wherein Alice gets a string $x \in \Sigma^k$, Bob gets a string $y \in \Sigma^k$, and their goal is to decide whether $(x,y) \in \mathsf{EIM}_C$ or (x,y) is ε-far from EIM_C. Using the results of [BBM11] (see Lemma 3.8) we have,

$$\mathsf{PT}_{\varepsilon}(\mathsf{EIM}_C) \geq \frac{1}{2\log|\Sigma|}\mathsf{CC}\left(\mathcal{C}^{\mathsf{EIM}_C}_{\mathsf{Pair},\varepsilon}\right). \tag{3.1}$$

Since by Theorem 3.11 we have $\mathsf{CC}(\mathsf{DISJ}_k) = \Omega(k)$, then it suffices to show that

$$\mathsf{CC}\left(\mathcal{C}^{\mathsf{EIM}_C}_{\mathsf{Pair},\varepsilon}\right) \geq \mathsf{CC}(\mathsf{DISJ}_k). \tag{3.2}$$

Toward this end, we show a reduction from the communication complexity problem DISJ_k to the communication complexity problem $\mathcal{C}^{\mathsf{EIM}_C}_{\mathsf{Pair},\varepsilon}$. We note that, under the natural association of EIM_C with YES-instances and "far from EIM_C" with NO-instances, our

reduction maps YES (resp., NO) instances of DISJ_k to NO (resp., YES) instances of EIM_C. Let π be a protocol for $\mathcal{C}^{\mathsf{EIM}_C}_{\mathsf{Pair},\varepsilon}$ with communication complexity c. Consider the following protocol for DISJ_k.

Let $x, y \in \{0,1\}^k$ be the inputs of Alice and Bob (respectively) for DISJ_k. Alice computes $\alpha = C(x)$. Bob computes $\beta = C(y)$. The players then run π on (α, β) and return the *negation* of its output.

Indeed, if $(x, y) \in \mathsf{DISJ}_k$ (i.e., their intersection is empty), then for every $i \in [k]$, either $x_i = 0$ or $y_i = 0$. Since the relative distance of C is at least δ_0, it holds that (α, β) is $(\delta_0/2)$-far from EIM_c. On the other hand, if $(x, y) \notin \mathsf{DISJ}_k$ (i.e., their intersection is of size 1), then there exists $i \in [k]$ such that $x_i = y_i = 1$. Hence, $\big(\alpha, \beta\big) \in \mathsf{EIM}_c$. Moreover, note that the total number of bits that were communicated is exactly c.

Using Eq. (3.1) and Eq. (3.2), together with Theorem 3.11, we conclude that for every $\varepsilon > 0$,

$$\mathsf{PT}_\varepsilon(\mathsf{EIM}_c) \geq \frac{1}{2\log|\Sigma|}\mathsf{CC}\left(\mathcal{C}^{\mathsf{EIM}_C}_{\mathsf{Pair},\varepsilon}\right) \geq \frac{1}{2\log|\Sigma|}\mathsf{CC}(\mathsf{DISJ}_k) = \Omega(k).$$

□

3.3.1.4 Proof of Theorem 3.7

In order to obtain an $O(1)$-relaxed-LDC that is also a $\mathsf{poly}(1/\varepsilon)$-LTC, we shall use the following construction of Ben-Sasson *et al.* [BSGH+06].

Theorem 3.12 ([BSGH+06, Remark 4.6])**.** *For every $\alpha > 0$, there exists a binary code that is an $O(1)$-relaxed-LDC and a t-LTC with constant relative distance and stretch $n = k^{1+\alpha}$, where for $\varepsilon > 1/\mathsf{polylog}(n)$ it holds that $t(n, \varepsilon) = \mathsf{poly}\left(\frac{1}{\alpha\varepsilon}\right)$.*

Theorem 3.7 follows by combining Theorem 3.12 with Lemma 3.5 and Lemma 3.6.

3.3.1.5 Proof of Theorem 3.8

In this section we show that a well-known variant of the Reed-Muller error-correcting code is an $\mathsf{polylog}(n)$-LDC (and in particular a $\mathsf{polylog}(n)$-relaxed-LDC) and a $\mathsf{poly}(\log n, 1/\varepsilon)$-LTC. Combining the latter with Lemma 3.5 and Lemma 3.6, we prove Theorem 3.8.

Lemma 3.9. *For every constant $\alpha > 0$, there exists a $\mathsf{polylog}(n)$-LDC that is also a $\mathsf{poly}(\log n, 1/\varepsilon)$-LTC with stretch $n = k^{1+\alpha}$ and relative distance $1 - o(1)$.*

Proof. We construct a code $C : \Sigma^k \to \Sigma^n$ as follows. Fix a finite field $\mathbb{F}$ and an integer m such that $|\mathbb{F}|^m = n$. The alphabet of the code is $\Sigma = \mathbb{F}$. Consider an arbitrary subset $H \subset \mathbb{F}$ of size $k^{1/m}$. We view a message $x \in \mathbb{F}^k$ as a function $x : H^m \to \mathbb{F}$ by identifying H^m and $[k]$ in some canonical way. The encoding $C(x)$ is the low degree extension $\hat{x}$ of x with respect to the field $\mathbb{F}$. Namely, the (unique) m-variate polynomial of individual degree $|H| - 1$ that agrees with x on H^m.

The code stretches $k = |H|^m$ symbols to $n = |\mathbb{F}|^m$ symbols, and by the Schwartz-Zippel Lemma it has relative distance at least $1 - \frac{m|H|}{|\mathbb{F}|}$. Furthermore, the code can be

locally tested using $O(m|H| \cdot \mathsf{poly}(1/\varepsilon))$ queries (see Theorem 3.31), and locally decoded using $O(m|H|)$ queries (see Theorem 3.29). Thus, to obtain our result we need to set our parameters as to maximize the ratio $|H|/|\mathbb{F}|$, while minimizing $m \cdot |H|$ and keeping $|\mathbb{F}| > m \cdot |H|$.

For every constant $\alpha > 0$ and every integer $n \in \mathbb{N}$, we let $\mathbb{F}$ be a finite field of size $(\log n)^{1/\alpha}$, let $m = \alpha \cdot \frac{\log n}{\log\log(n)}$ and let H be some fixed (arbitrary) subset of $\mathbb{F}$ of size $|\mathbb{F}|^{1-\alpha}$. Hence, $\frac{m \cdot |H|}{|\mathbb{F}|} = \alpha \cdot \frac{\log n}{\log\log n} \cdot |\mathbb{F}|^{-\alpha} = o(1)$. The code has relative distance $1 - \frac{(|H|-1) \cdot m}{|\mathbb{F}|} = 1 - o(1)$, stretch $n = |\mathbb{F}|^m = |H|^{m/(1-\alpha)} = k^{1/(1-\alpha)}$. In addition, it can be locally tested using $\mathsf{poly}(\log n, 1/\varepsilon)$ queries, and locally decoded using $\mathsf{polylog}(n)$ queries. □

A natural property. We remark that when the *encoded intersecting messages* property is instantiated with the foregoing variant of the Reed-Muller code, we obtain a *natural* property that consists of pairs (P, Q) of low-degree polynomials, whose product $P \cdot Q$ is non-zero on a given subset of its domain. That is, the property is

$$\Pi_{\mathbb{F},d,m,H} = \left\{ (P,Q) : P, Q : \mathbb{F}^m \to \mathbb{F} \text{ have individual degree } d \text{ and } \sum_{x \in H^m} (P \cdot Q)(x) \neq 0 \right\}.$$

3.3.2 Trade-off between Query and Proof Complexity

In this section we show a property that has a multiplicative trade-off between proof and query complexities for MAP testing. We show a property that can be tested with a nearly smooth tradeoff between the proof and query complexities.

Theorem 3.13. *For every constant $\alpha > 0$, there exists a property Π_α such that for every sublinear function $p : \mathbb{N} \to \mathbb{N}$, the query complexity of Π for MAP verifiers, which use proofs of length p, is upper bounded by $\frac{n^{1-\alpha+o(1)}}{p} \cdot \mathsf{poly}(1/\varepsilon)$ and lower bounded by $\tilde{\Omega}\left(\frac{n^{1-\alpha}}{p}\right)$.*

Our proof is heavily based on multivariate polynomials, and we refer the reader to Section 3.A.4 for the necessary background (e.g., the Schwartz-Zippel lemma and low degree testing). In fact, the proof of Theorem 3.13 is based on a specific algebraic property that we call *Sub-Tensor Sum*. We note that this property will also be used in Section 3.3.3 and Section 3.3.4.

We proceed to describe the sub-tensor sum problem. Let $\mathbb{F}$ be a finite field, let $m, d \in \mathbb{N}$ such that $d \cdot m < |\mathbb{F}|/10$ and let $H \subset \mathbb{F}$. Consider the following property.

Definition 3.10. *The* Sub-Tensor Sum *property, denoted $\mathsf{TensorSum}_{\mathbb{F},m,d,H}$, is parameterized by a field $\mathbb{F}$, a dimension $m \in \mathbb{N}$, a degree $d \in \mathbb{N}$ and a subset $H \subset \mathbb{F}$, and contains all polynomials $P : \mathbb{F}^m \to \mathbb{F}$ of individual degree d, such that*

$$\sum_{x \in H^m} P(x) = 0$$

where the arithmetic is over $\mathbb{F}$.

To obtain a tight trade-off, we shall be using some $d = \Theta(|H|)$. To simplify the notation, when the parameters are clear from the context, we shorthand $\mathsf{TensorSum}$ for $\mathsf{TensorSum}_{\mathbb{F},m,d,H}$. Next, we proceed to show the (almost) tight multiplicative trade-off for $\mathsf{TensorSum}$. In Section 3.3.2.1 we prove the upper bound and in Section 3.3.2.2 we prove the lower bound. Finally, in Section 3.3.2.3 we set the parameters for proving Theorem 3.13.

3.3.2.1 MAP Upper Bound for TensorSum

We start by proving the following upper bound.

Lemma 3.11. *If $dm < |\mathbb{F}|/10$, then, for every $\ell \in \{0,\ldots,m\}$, the $\mathsf{TensorSum}_{\mathbb{F},m,d,H}$ property has an MAP with proof complexity $(d+1)^\ell \cdot \log(|F|)$ and query complexity $|H|^{m-\ell} \cdot (dm^2 \log|H|) \cdot \mathsf{poly}(1/\varepsilon)$. Furthermore, the MAP has a one-sided error.*

We note that the additional parameter ℓ essentially controls the proof length (and will be set as roughly the logarithm of the desired proof length). Moreover, d will be set such that $d = \Theta(|H|)$ and therefore $d^\ell \cdot |H|^{m-\ell} \approx |H|^m$ and so we can set ℓ to obtain the desired trade-off between proof and query complexities.

Proof of Lemma 3.11. We prove the lemma by showing an MAP protocol for the statement $P \in \mathsf{TensorSum}$. The main idea is to partition H^m into $|H|^\ell$ sub-tensors of the form $(x_1,\ldots,x_\ell,*,*,\ldots,*)$ for every $x_1,\ldots,x_\ell \in H$, and use a *low degree* ℓ-variate polynomial Q such that $Q(x_1,\ldots,x_\ell)$ equals the sum of the $(x_1,\ldots,x_\ell)^{\text{th}}$ tensor over $H^{m-\ell}$. Specifically, we refer to the polynomial:

$$Q(x_1,\ldots,x_\ell) = \sum_{x_{\ell+1},\ldots,x_m \in H} P(x_1,\ldots,x_m).$$

Thus, the MAP proof for the statement $P \in \mathsf{TensorSum}$, consists of the polynomial Q. The verifier checks that (1) P is (close to) a low degree polynomial, (2) the sum of Q on H^ℓ is 0, and (3) that Q is consistent with P. The last step uses the fact that both Q and P are low degree polynomials and so it suffices to verify consistency of a random point in Q by reading the entire corresponding sub-tensor (i.e., $|H|^{m-\ell}$ points) from P. Actually, since P can only be verified to be *close to* a low degree polynomial, the $|H|^{m-\ell}$ points are read via self-correction. The detailed protocol is presented in Fig. 3.2 (where all arithmetic is over $\mathbb{F}$).

Note that the proof of proximity consists of $|Q| = O((d+1)^\ell \log|\mathbb{F}|)$ bits and that the total number of queries to the oracle is dominated by the $|H|^{m-\ell}$ invocations of the self-correction algorithm (which requires $(m\log(|H|) \cdot dm \cdot \mathsf{poly}(1/\varepsilon)$ queries for each invocation to obtain the desired soundness level). We proceed to show that completeness and soundness hold.

MAP for TensorSum with parameter $\ell \leq m$

Parameters: $\mathbb{F}$ (field), m (dimension), d (individual degree) and $H \subset \mathbb{F}$.

Input: a proximity parameter $\varepsilon \in (0, 1/3)$, and oracle access to a function $P : \mathbb{F}^m \to \mathbb{F}$.

The Proof:

- The proof consists of a multivariate polynomial $\tilde{Q} : \mathbb{F}^\ell \to \mathbb{F}$ of individual degree d (specified by its $(d+1)^\ell$ coefficients), which allegedly equals

$$Q(x_1, \ldots, x_\ell) \stackrel{\text{def}}{=} \sum_{x_{\ell+1}, \ldots, x_m \in H} P(x_1, \ldots, x_m).$$

The Verifier:

1. If $\sum_{x_1, \ldots, x_\ell \in H} \tilde{Q}(x_1, \ldots, x_\ell) \neq 0$, then reject.
2. Run the low individual d-degree test (see Theorem 3.31) on P with respect to the proximity parameter ε. If the test fails, then reject.
3. Select uniformly at random $r_1, \ldots, r_\ell \in_R \mathbb{F}$.
4. For every $x_{\ell+1}, \ldots, x_m \in H$, read the value of $P(r_1, \ldots, r_\ell, x_{\ell+1}, \ldots, x_m)$ using self correction (see Theorem 3.29) repeated $O(m \log(|H|))$ times (to reduce the error probability to $\frac{1}{10|H|^m}$ for each point). Denote the value read by $z_{r_1, \ldots, r_\ell, x_{\ell+1}, \ldots, x_m}$.
5. Accept if $\tilde{Q}(r_1, \ldots, r_\ell) = \sum_{x_{\ell+1}, \ldots, x_m \in H} z_{r_1, \ldots, r_\ell, x_{\ell+1}, \ldots, x_m}$ and otherwise reject.

Figure 3.2: MAP for TensorSum

Completeness. If $P \in$ TensorSum, then $\sum_{x_1, \ldots, x_\ell \in H} Q(x_1, \ldots, x_\ell) = 0$ and P has individual degree d (and so the individual degree test passes). Moreover, in this case $\tilde{Q} = Q$ and

$$Q(r_1, \ldots, r_\ell) = \sum_{x_{\ell+1}, \ldots, x_m \in H} P(r_1, \ldots, r_\ell, x_{\ell+1}, \ldots, x_m).$$

By the zero-error feature of the self-correction procedure, with probability 1,

$$z_{r_1, \ldots, r_\ell, x_{\ell+1}, \ldots, x_m} = P(r_1, \ldots, r_\ell, x_{\ell+1}, \ldots, x_m),$$

and therefore $\sum_{x_{\ell+1}, \ldots, x_m \in H} z_{r_1, \ldots, r_\ell, x_{\ell+1}, \ldots, x_m} = \tilde{Q}(r_1, \ldots, r_\ell)$. Hence, in this case, the verifier accepts with probability 1.

Soundness. Let $\varepsilon > 0$ and let $P : \mathbb{F}^m \to \mathbb{F}$ be a polynomial that is ε-far from TensorSum. Let $\tilde{Q}$ be an alleged proof (to the false statement $P \in$ TensorSum).

Consider first the case that P is ε-far from having individual degree d. In this case, by the individual degree test (Theorem 3.31), the verifier rejects with probability at least $1/2$. Thus, we focus on the case that P is ε-close to a polynomial P' of individual degree d. We may also assume that $\sum_{x_1,\ldots,x_\ell \in H} \tilde{Q}(x_1,\ldots,x_\ell) = 0$ (since otherwise the verifier rejects with probability 1). Define

$$Q'(x_1,\ldots,x_\ell) \stackrel{\text{def}}{=} \sum_{x_{\ell+1},\ldots,x_m \in H} P'(x_1,\ldots,x_m).$$

Clearly $\sum_{x_1,\ldots,x_\ell} Q'(x_1,\ldots,x_\ell) \neq 0$ (since otherwise P is ε-close to $P' \in \mathsf{TensorSum}$). Thus, the individual degree d polynomials Q' and $\tilde{Q}$ differ, and so, by the Schwartz-Zippel Lemma they can agree on at most a $\frac{d\ell}{\mathbb{F}}$ fraction of their domain $\mathbb{F}^\ell$.

To complete the argument note that the self-correction algorithm guarantees that every $z_{r_1,\ldots,r_\ell,x_{\ell+1},\ldots,x_m}$ is equal to $P'(r_1,\ldots,r_\ell,x_{\ell+1},\ldots,x_m)$, with probability $1 - \frac{1}{10|H|^m}$ (here we use our assumption that, without loss of generality, $\varepsilon < 1/3$). Therefore, by the union bound, all points are read correctly with probability at least 0.9, and in this case $\sum_{x_{\ell+1},\ldots,x_m \in H} z_{r_1,\ldots,r_\ell,x_{\ell+1},\ldots,x_m} = Q'(r_1,\ldots,r_\ell)$. Thus, with probability $0.9 \cdot (1 - \frac{dm}{\mathbb{F}}) \geq 2/3$, the verifier rejects when testing that $\tilde{Q}(r_1,\ldots,r_\ell)$ equals $\sum_{x_{\ell+1},\ldots,x_m \in H} z_{r_1,\ldots,r_\ell,x_{\ell+1},\ldots,x_m}$. □

3.3.2.2 MAP Lower Bound for TensorSum

Next, we give an (almost) matching lower bound on the MAP complexity of *Sub-Tensor Sum*. Formally, we show

Lemma 3.12. *For every $\varepsilon \in (0, 1 - \frac{dm}{|\mathbb{F}|})$, if $d \geq 2(|H|-1)$, then every* MAP *for* TensorSum *(with respect to proximity parameter ε) that has proof complexity $p \geq 1$ must have query complexity $q = \Omega\left(\frac{|H|^m}{p \cdot \log|\mathbb{F}|}\right)$.*

As an immediate corollary of Lemma 3.12 we obtain the following:[12]

Corollary 3.14. *For every $\varepsilon \in (0, 1 - \frac{dm}{|\mathbb{F}|})$, if $d \geq 2(|H| - 1)$,*

$$\mathsf{PT}_\varepsilon(\mathsf{TensorSum}) = \Omega\left(\frac{|H|^m}{\log(|\mathbb{F}|)}\right).$$

In order to prove Lemma 3.12, we first extend the framework of [BBM11] from the property testing model to the MAP model. More specifically, we show a methodology for proving lower bounds on MAPs via MA communication complexity lower bounds. We refer the reader to Section 3.A.2 for background on MA communication complexity.

Let Π be a property and let ψ be a simple combining operator (see Definition 3.7). For every proximity parameter $\varepsilon > 0$, denote by $\mathcal{C}^{\Pi}_{\psi,\varepsilon}$ the communication complexity problem in which Alice gets as input a function f and Bob gets as input a function g and they

[12] The corollary can be derived by setting $p = 1$, and the fact that any property tester is an MAP.

need to decide between a YES-instance, wherein $\psi(f,g) \in \Pi$, and a NO-instance, wherein $\psi(f,g)$ is ε-far from Π.[13] We prove the following lemma.

Lemma 3.13 (MAP lower bounds via MA communication complexity)**.** *For any* simple *combining operator* ψ*, any property* Π *and any proximity parameter* $\varepsilon > 0$*, if* $\Pi \in \mathsf{MAP}(p,q)$*, then* $\mathcal{C}^{\Pi}_{\psi,\varepsilon}$ *has an* MA *communication complexity protocol with a proof of length* p *and total communication* $2q|\psi|$*.*

Proof. Let V be an MAP verifier for Π with proof complexity p and query complexity q. We construct an MA communication complexity protocol for $\mathcal{C}^{\Pi}_{\psi,\varepsilon}$. Recall that Alice and Bob get as input function f and g (respectively) and have free access to a proof string $w \in \{0,1\}^p$.

The (honest) proof string for the protocol is simply the proof string w of the MAP with respect to $h \stackrel{\text{def}}{=} \psi(f,g)$. As their first step, Alice and Bob emulate the execution of the MAP protocol with respect to the proof string w using their common random string as the source of randomness (for the emulated verifier). Whenever the MAP verifier V queries the input at a point x, Alice and Bob compute $f(x)$ and $g(x)$ (respectively) and send their values to each other. Since ψ is a *simple* combining operator, each player can compute $h(x)$ from $x, f(x)$ and $g(x)$, and feed it as an answer to the emulated MAP verifier. The players accept if V accepts, and reject otherwise.

Observe that both players use the same common random string as the source of randomness, and forward the same values to the MAP verifier (i.e., both the proof string and the oracle answers). Therefore, they emulate the verifier identically.

Note that by the definition of the communication complexity problem, if $(f,g) \in \mathcal{C}^{\Pi}_{\psi,\varepsilon}$, then $h \in \Pi$; hence the verifier will accept. On the other hand, if the pair $(f,g) \notin \mathcal{C}^{\Pi}_{\psi,\varepsilon}$, then h is ε-far from Π, so the verifier will reject.

During the entire reduction, the players communicated $2|\psi|$ bits for every query of the verifier. Hence the total number of bits that were communicated is $2|\psi| \cdot q$. □

We proceed by stating Klauck's lower bound on the MA communication complexity of set-disjointness [Kla03], and use Lemma 3.13 to show a lower bound on the MAP complexity of the *Sub-Tensor Sum* property.

Theorem 3.15 ([Kla03])**.** *Every* MA *communication complexity protocol for* DISJ_n *with proof complexity* p *and communication complexity* c *satisfies* $p \cdot c = \Omega(n)$*.*

Proof of Lemma 3.12. Denote $k = |H|^m$ and by $f \cdot g$ the function $h(x) \stackrel{\text{def}}{=} f(x) \cdot g(x)$. Let $\mathcal{C}^{\mathsf{TensorSum}}_{\cdot,\varepsilon}$ be the communication complexity problem wherein Alice gets a function

[13] When proving property testing lower bounds via standard (i.e., non-MA) communication complexity lower bounds (using [BBM11] framework) one may also map YES-instances (respectively, NO-instances) of communication complexity problems to NO-instances (respectively, YES-instances) of property testing problems. This is possible due to the *symmetrical* definition of standard communication complexity (in fact, the above was used in the proof of Lemma 3.6). In contrast, the definition of MA communication complexity is *asymmetrical*; therefore when using our extension of the framework to MA one must map YES-instances to YES-instances, and NO-instances to NO-instances.

$f : \mathbb{F}^m \to \mathbb{F}$, Bob gets a function $g : \mathbb{F}^m \to \mathbb{F}$, and their goal is to decide whether $f \cdot g \in \mathsf{TensorSum}$ or $f \cdot g$ is ε-far from $\mathsf{TensorSum}$.

Recall that by Theorem 3.15 we know that every MA communication complexity protocol for DISJ_k with proof complexity p and communication complexity c satisfies $p \cdot c = \Omega(k)$. On the other hand, by Lemma 3.13 we know that if $\mathsf{TensorSum} \in \mathsf{MAP}(p, q)$, then $\mathsf{CC}(\mathcal{C}_{\cdot,\varepsilon}^{\mathsf{TensorSum}})$ has an MA communication complexity protocol with a proof of length p and a total of $2q \log |\mathbb{F}|$ communication.

Hence, to prove the lemma, it suffices to reduce DISJ_k to $\mathcal{C}_{\cdot,\varepsilon}^{\mathsf{TensorSum}}$ (this reduction takes place entirely within the setting of MA communication complexity). Toward this end, suppose that π is an MA communication complexity protocol for $\mathcal{C}_{\cdot,\varepsilon}^{\mathsf{TensorSum}}$ with proof complexity p and communication complexity c. We use π to construct an MA protocol for DISJ_k.

Let $a \in \{0,1\}^k$ and $b \in \{0,1\}^k$ be the respective inputs of Alice and Bob for the set-disjointness problem. Recall that $\mathbb{F}$ (a finite field), d (the individual degree), m (the dimension) and $H \subset \mathbb{F}$ are parameters of the $\mathsf{TensorSum}$ problem. The reduction to $\mathsf{TensorSum}$ proceeds as follows. First, Alice and Bob compute the low degree extension $\hat{a}$ and $\hat{b}$ of their respective inputs with respect to $\mathbb{F}, m, d$ and H. Namely, they associate their inputs a and b with indicator functions $a, b : H^m \to \{0,1\}$ by mapping $[k]$ to H^m in some canonical way. Then, they compute the (unique) polynomials $\hat{a}, \hat{b} : \mathbb{F}^m \to \mathbb{F}$ of individual degree $|H| - 1$ that agree with a and b (respectively) on H^m.

Denote by w the proof for the protocol π with respect to the input pair $(\hat{a}, \hat{b})$. The proof for the set disjointness problem is simply w. Alice and Bob proceed by running π on input $(\hat{a}, \hat{b})$, with respect to the proof w and proximity parameter ε and return its output.

Observe that if $(a, b) \in \mathsf{DISJ}_k$, then $\sum_{i \in [k]} a_i b_i = 0$ (where the summation is over the integers). Hence,

$$\sum_{x_1,\ldots,x_m \in H} \hat{a}(x_1, \ldots, x_m) \cdot \hat{b}(x_1, \ldots, x_m) = \sum_{x_1,\ldots,x_m \in H} a(x_1, \ldots, x_m) \cdot b(x_1, \ldots, x_m) = 0$$

(where the first summation is over $\mathbb{F}$, and the second summation is over the integers). Thus, $\hat{a} \cdot \hat{b} \in \mathsf{TensorSum}_{\mathbb{F},m,d,H}$ (here we use the lemma's hypothesis that $d \geq 2(|H| - 1)$ since $\hat{a} \cdot \hat{b}$ is the product of two polynomials of individual degree $|H| - 1$). We conclude that there exists a proof w of length p such that the MA communication complexity protocol for DISJ_k accepts with high probability.

On the other hand, if $(a, b) \notin \mathsf{DISJ}_k$, then (by the promise of having an intersection of size at most 1) it holds that $\sum_{i \in [k]} a_i b_i = 1$ (where the summation is over the integers). Hence

$$\sum_{x_1,\ldots, \in H} \hat{a}(x_1, \ldots, x_m) \cdot \hat{b}(x_1, \ldots, x_m) = \sum_{x_1,\ldots,x_m \in H} a(x_1, \ldots, x_m) \cdot b(x_1, \ldots, x_m) = 1$$

(where the first summation is over $\mathbb{F}$, and the second summation is over the integers). Thus, $\hat{a} \cdot \hat{b}$ is an m-variate polynomials of (individual) degree d ($\geq 2(|H| - 1)$) whose

sum over H^m is non-zero. By the Schwartz-Zippel lemma (see Section 3.A.4), and since $\varepsilon < 1 - \frac{dm}{|\mathbb{F}|}$, the function $\hat{a} \cdot \hat{b}$ is at least ε-far from TensorSum.

We conclude that every MAP verifier for TensorSum with q queries and p proof length must satisfy $q \cdot p \geq \Omega\left(\frac{k}{\log(|\mathbb{F}|)}\right)$. □

3.3.2.3 Proof of Theorem 3.13

In this section we complete the proof of Theorem 3.13, which states that for every constant $\alpha > 0$, there exists a property Π_α such that for every sublinear function $p : \mathbb{N} \to \mathbb{N}$, the query complexity of Π for MAP verifiers that use proofs of length p is upper bounded by $\frac{n^{1-\alpha+o(1)}}{p} \cdot \mathsf{poly}(1/\varepsilon)$ and lower bounded by $\tilde{\Omega}\left(\frac{n^{1-\alpha}}{p}\right)$.

Toward this end, we need to set the parameters of the TensorSum problem. Our parameters are governed by $n = |\mathbb{F}|^m$ (i.e., the size of the object equals n), $dm < |\mathbb{F}|/10$ (so that we can apply the Schwartz-Zippel lemma) and $d = 2(|H| - 1)$ (see Lemma 3.12). Since $p \cdot q = \tilde{\Omega}(|H|^m)$, and the object size is $|\mathbb{F}|^m$, we need to maximize the ratio $|H|/|\mathbb{F}|$ to obtain a better lower bound (while recalling that $|H| \leq d/2 - 1$).

For every constant $\alpha > 0$ and every integer $n \in \mathbb{N}$, let $\mathbb{F}$ be a finite field of size $(\log n)^{1/\alpha}$, let $m = \alpha \cdot \frac{\log n}{\log\log(n)}$, let H be some fixed (arbitrary) subset of $\mathbb{F}$ of size $|\mathbb{F}|^{1-\alpha}$ and let $d = 2(|H| - 1)$. Note that $|\mathbb{F}|^m = n$ and $|H|^m = n^{1-\alpha}$.

Lemma 3.11 guarantees the existence of an MAP for $\mathsf{TensorSum}_{\mathbb{F},m,d,H}$ with proof complexity $(d+1)^\ell \cdot \log(|F|)$ and query complexity $|H|^{m-\ell} \cdot dm^2 \log(|H|)$ for every $\ell \in [m]$. Thus, for every parameter $p \in \{(d+1)^i \cdot \log(|\mathbb{F}|) : i \in \mathbb{N}\}$ (which corresponds to the proof length), we set:

$$\ell = \frac{\log(p) - \log\log(|F|)}{\log(d+1)}.$$

and apply Lemma 3.11. We obtain an MAP protocol for computing $\mathsf{TensorSum}_{\mathbb{F},m,d,H}$ with a proof of length

$$(d+1)^\ell \cdot \log(|F|) = p$$

and query complexity:

$$|H|^{m-\ell} \cdot dm^2 \log(|H|) \cdot \mathsf{poly}(1/\varepsilon) = \frac{n^{1-\alpha}}{|H|^\ell} \cdot \mathsf{polylog}(n) \cdot \mathsf{poly}(1/\varepsilon). \tag{3.3}$$

By our setting of ℓ we have:

$$|H|^\ell = |H|^{\frac{\log p - \log\log|\mathbb{F}|}{\log(d+1)}} \geq 2^{\frac{\log|H|}{\log(2|H|)} \cdot (\log p - \log\log|\mathbb{F}|)} = \left(\frac{p}{\log|\mathbb{F}|}\right)^{1-\frac{1}{1+\log H}} \geq \frac{p}{n^{o(1)}} \tag{3.4}$$

where the first inequality follows from $d = 2(|H|-1) \leq 2|H|-1$ and the second inequality follows from our setting of $|H|$ and $|\mathbb{F}|$ (and since $p \leq n$). Combining Eq. (3.3) and Eq. (3.4) we have that the query complexity of the MAP is $\frac{n^{1-\alpha+o(1)}}{p} \cdot \mathsf{poly}(1/\varepsilon)$.

On the other hand, by Lemma 3.12, for every MAP for TensorSum with proof complexity p and query complexity q, it holds that $p \cdot q \geq \Omega\left(\frac{|H|^m}{\log|\mathbb{F}|}\right) = \tilde{\Omega}(n^{1-\alpha})$. The theorem follows.

3.3.3 MAP vs. IPP$[O(1)]$

In this section and the following one, we consider the power of MAP in comparison to the more general notion of IPP (for a formal definition of IPP, see Section 3.2.2.) Roughly speaking, in this section we show a property that requires $\sqrt{n}$ queries by an MAP verifier that uses a proof of length $\sqrt{n}$ but requires only $\mathsf{polylog}(n)$ queries by an IPP[3] verifier (i.e., an IPP with only 3-messages) that also uses a proof of length $\sqrt{n}$.

Theorem 3.16. *For every $\alpha > 0$, there exists a property Π_α such that:*

1. *The MAP complexity of Π_α is $\tilde{\Omega}\left(n^{1/2-\alpha}\right)$; and*
2. *There is an IPP[3] for Π_α with $\mathsf{polylog}(n) \cdot \mathsf{poly}(1/\varepsilon)$ query complexity and communication complexity $\tilde{O}(n^{1/2-\alpha+o(1)})$.*

The property that we use is the TensorSum property (introduced in Section 3.3.2). Note that the first part of Theorem 3.16 was already shown in Theorem 3.13, and so, to prove Theorem 3.16, what remains to be shown is that TensorSum can be tested by a 3-message IPP verifier that uses roughly $\sqrt{n}$ communication and $\mathsf{polylog}(n)$ queries.

Lemma 3.14. *If $dm < |\mathbb{F}|/10$, then there is a 3-message IPP for $\mathsf{TensorSum}_{\mathbb{F},d,m,H}$ (where $\mathbb{F}$ is a finite field, m is the dimension, d is the degree and $H \subset \mathbb{F}$) with communication complexity $O\left((d+1)^{m/2}\log(|\mathbb{F}|)\right)$ and query complexity $O\left(dm \cdot \mathsf{poly}(1/\varepsilon)\right)$.*

We note that Theorem 3.16 follows from Lemma 3.14 (and Lemma 3.12) by setting the parameters $\mathbb{F}, m, d, H$ as in Section 3.3.2.3. Namely, fix a finite field $\mathbb{F}$ of size $(\log n)^{1/\alpha}$, a dimension $m = \alpha \cdot \frac{\log n}{\log\log(n)}$, an arbitrary subset $H \subset \mathbb{F}$ of size $|F|^{1-\alpha}$ and set $d = 2(|H|-1)$. We proceed to prove Lemma 3.14

Proof of Lemma 3.14. The first part of the protocol closely resembles the MAP that was presented in Lemma 3.11. Indeed, the first message from the prover to the verifier is the polynomial Q that is (allegedly) the sum of P on H^ℓ sub-tensors of H^m, each of dimension $m-\ell$. The verifier checks that P is close to a low degree polynomial and that Q sums to 0, but the consistency check of P and Q is different. Recall that in Lemma 3.11, the verifier chose a random sub-tensor and checked the consistency of Q and P by reading all points in the sub-tensor. Using two additional messages we replace these queries by having the prover provide them. That is, after the prover "commits" to the sum of all sub-tensors, the verifier chooses one of them at random and sends its choice to the prover. Then, the prover provides the value of *all* points in that sub-tensor via a polynomial $W : \mathbb{F}^{m-\ell} \to \mathbb{F}$ of individual degree $|H| - 1$. The verifier can readily check the that the two polynomials Q and W sent by the prover are consistent with each other (using no queries to P), and that the second polynomial (i.e., W) is consistent with P using only a constant number of queries.

Similarly to the protocol of Section 3.3.2, the protocol uses a parameter ℓ except that in this case, an optimal result is obtained by fixing $\ell = m/2$ (but for simplicity of notations we keep ℓ as a parameter). The IPP[3] protocol, in which the prover is denoted

by $\mathcal{P}$ and the verifier is denoted by $\mathcal{V}$, is described in Section 3.3.3. It can be readily verified that by setting $\ell = m/2$, the query and communication complexities are as stated. We proceed to prove that completeness and soundness hold.

IPP[3] for TensorSum

Parameters: $\mathbb{F}$ (field), m (dimension), d (individual degree), $H \subset \mathbb{F}$ and $\ell = m/2$.

Input: a proximity parameter $\varepsilon \in (0, 1/3)$, and oracle access to a function $P : \mathbb{F}^m \to \mathbb{F}$.

1. $\mathcal{V}$ runs the low individual d-degree test (see Theorem 3.31) on P with respect to the proximity parameter ε. If the test fails then $\mathcal{V}$ rejects.
2. $\mathcal{P}$ sends to $\mathcal{V}$ an individual degree d multivariate polynomial $\tilde{Q} : \mathbb{F}^\ell \to \mathbb{F}$ of individual degree d (by specifying its $(d+1)^\ell$ coefficients), which allegedly equals
$$Q(x_1, \ldots, x_\ell) \stackrel{\text{def}}{=} \sum_{x_{\ell+1}, \ldots, x_m \in H} P(x_1, \ldots, x_m).$$
3. If $\sum_{x_1, \ldots, x_\ell \in H} \tilde{Q}(x_1, \ldots, x_\ell) \neq 0$, then $\mathcal{V}$ rejects.
4. $\mathcal{V}$ selects uniformly at random $r_1, \ldots, r_\ell \in_R \mathbb{F}$ and sends $r_1, \ldots, r_\ell$ to $\mathcal{P}$.
5. $\mathcal{P}$ sends to $\mathcal{V}$ an individual degree d multivariate polynomial $\tilde{W} : \mathbb{F}^{m-\ell} \to \mathbb{F}$ of individual degree d (by specifying its $(d+1)^{m-\ell}$ coefficients), which allegedly equals
$$W(x_{\ell+1}, \ldots, x_m) \stackrel{\text{def}}{=} P(r_1, \ldots, r_\ell, x_{\ell+1}, \ldots, x_m).$$
6. $\mathcal{V}$ selects at random $s_{\ell+1}, \ldots, s_m \in_R \mathbb{F}$, reads the value $z_{r_1, \ldots, r_\ell, s_{\ell+1}, \ldots, s_m}$ of the polynomial $P(r_1, \ldots, r_\ell, s_{\ell+1}, \ldots, s_m)$ using the self-correction algorithm (see Theorem 3.29) with soundness error $1/10$ and rejects if $z_{r_1, \ldots, r_\ell, s_{\ell+1}, \ldots, s_m} \neq W(s_{\ell+1}, \ldots, s_m)$.
7. $\mathcal{V}$ accepts if $\tilde{Q}(r_1, \ldots, r_\ell) = \sum_{x_{\ell+1}, \ldots, x_m \in H} \tilde{W}(x_{\ell+1}, \ldots, x_m)$ and rejects otherwise.

Figure 3.3: IPP[3] for TensorSum

Completeness. If $P \in$ TensorSum, then P has individual degree d and the low degree tests passes. In this case $\tilde{Q} = Q$ and $\tilde{W} = W$ and therefore all the verifier's tests pass (since $\sum_{x_1, \ldots, x_\ell \in H} Q(x_1, \ldots, x_\ell) = 0$ holds as well).

Soundness. Let $\varepsilon > 0$ and let $P : \mathbb{F}^m \to \mathbb{F}$ be ε-far from TensorSum. If P is ε-far from having individual degree d, then the low degree test rejects with probability at least $1/2$ and so we assume that P is ε-close to an individual degree d polynomial P'. The

(cheating) prover sends two polynomials $\tilde{Q}$ and an $\tilde{W}$. We proceed to prove two claims regarding these polynomials.

Claim 3.14.1. *If $\tilde{Q}(x_1,\ldots,x_\ell) \equiv \sum_{x_{\ell+1},\ldots,x_m\in H} P'(x_1,\ldots,x_m)$ (as formal polynomials over $x_1,\ldots,x_\ell$), then the verifier rejects with probability 1.*

Proof. Observe that $\sum_{x_1,\ldots,x_m\in H} P'(x_1,\ldots,x_m) \neq 0$, as otherwise P is ε-close to TensorSum. Therefore, if the polynomials $\tilde{Q}(x_1,\ldots,x_\ell)$ and $\sum_{x_{\ell+1},\ldots,x_m\in H} P'(x_1,\ldots,x_m)$ are equal, then the verifier rejects when testing whether $\sum_{x_1,\ldots,x_\ell\in H} \tilde{Q}(x_1,\ldots,x_\ell) = 0$. □

Claim 3.14.2. *For every value of $r_1,\ldots,r_\ell \in \mathbb{F}$, if the prover sends an individual-degree d polynomial $\tilde{W}(x_{\ell+1},\ldots,x_m)$ (which depends on $r_1,\ldots,r_\ell$) that differs from the polynomial $P'(r_1,\ldots,r_\ell,x_{\ell+1},\ldots,x_m)$ (as formal polynomials in $x_{\ell+1},\ldots,x_m$), then the verifier rejects with probability at least 2/3.*

Proof. Assume that $\tilde{W}(x_{\ell+1},\ldots,x_m) \not\equiv P'(r_1,\ldots,r_\ell,x_{\ell+1},\ldots,x_m)$. Thus, the polynomials $\tilde{W}(x_{\ell+1},\ldots,x_m)$ and $P'(r_1,\ldots,r_\ell,x_{\ell+1},\ldots,x_m)$ are two different $(m-\ell)$-variate polynomials of individual degree d and, by the Schwartz-Zippel Lemma, they can agree on at most a $\frac{d(m-\ell)}{|\mathbb{F}|} < 0.1$ fraction of their domain. Therefore, with probability 0.9 over the verifier's choice of $s_{\ell+1},\ldots,s_m \in \mathbb{F}$, it holds that

$$\tilde{W}(s_{\ell+1},\ldots,s_m) \neq P'(r_1,\ldots,r_\ell,s_{\ell+1},\ldots,s_m).$$

Using the self-correction procedure, with probability at least 0.9, the verifier correctly obtains the value $z_{r_1,\ldots,r_\ell,s_{\ell+1},\ldots,s_m} = P'(r_1,\ldots,r_\ell,s_{\ell+1},\ldots,s_m)$. Hence, with probability at least $0.9^2 > 2/3$, the verifier rejects when testing whether $z_{r_1,\ldots,r_\ell,s_{\ell+1},\ldots,s_m} = \tilde{W}(s_{\ell+1},\ldots,s_m)$. □

By Claim 3.14.2, we can assume that

$$\tilde{W}(x_{\ell+1},\ldots,x_m) \equiv P'(r_1,\ldots,r_\ell,x_{\ell+1},\ldots,x_m) \tag{3.5}$$

(since otherwise the verifier rejects). On the other hand, by Claim 3.14.1 and using the Schwartz-Zippel Lemma, with probability at least $1-\frac{d\ell}{|\mathbb{F}|}$ over the choice of $r_1,\ldots,r_\ell \in_R \mathbb{F}$, it holds that

$$\tilde{Q}(r_1,\ldots,r_\ell) \neq \sum_{x_{\ell+1},\ldots,x_m\in H} P'(r_1,\ldots,r_\ell,x_{\ell+1},\ldots,x_m) = \sum_{x_{\ell+1},\ldots,x_m\in H} \tilde{W}(x_{\ell+1},\ldots,x_m)$$

where the last equality is due to Eq. (3.5). Hence, the verifier rejects with probability $1-\frac{d\ell}{|\mathbb{F}|} > 0.9$ when testing whether $\tilde{Q}(r_1,\ldots,r_\ell) = \sum_{x_{\ell+1},\ldots,x_m\in H} W(x_{\ell+1},\ldots,x_m)$. This completes the proof of Lemma 3.14. □

3.3.4 Exponential Separation between MAP and IPP

In this section we show an exponential separation between MAP and general IPP. Namely, we show a property that has MAP complexity roughly $\sqrt{n}$ but has IPP complexity $\mathsf{polylog}(n)$. In contrast to the IPP of Section 3.3.3 (which used $O(1)$ messages) here we use an IPP with *poly-logarithmically* many messages.

Theorem 3.17. *For every $\alpha > 0$, there exists a property Π_α such that:*

1. *The* MAP *complexity of Π_α is $\tilde{\Omega}\left(n^{1/2-\alpha} \cdot \mathsf{poly}(1/\varepsilon)\right)$; and*
2. *Π_α has an* IPP *with query complexity $\mathsf{polylog}(n) \cdot \mathsf{poly}(1/\varepsilon)$ and communication complexity $\mathsf{polylog}(n)$.*

Moreover, the PT *complexity of Π_α is $\tilde{\Theta}(n^{1-\alpha})$.*

To prove Theorem 3.17, we yet again use the TensorSum problem. The first part of the theorem follows directly from Theorem 3.13 and the query complexity of property testers (without a proof) is implied by Corollary 3.14.[14] Thus, to prove the theorem, all that remains is to show an IPP protocol for TensorSum.

Lemma 3.15. *If $d \cdot m < \mathbb{F}/10$, then there exists an m-round* IPP *for $\mathsf{TensorSum}_{\mathbb{F},m,d,H}$ with communication complexity $O(dm \log |F|)$, and query complexity $O(dm \cdot \mathsf{poly}(1/\varepsilon))$.*

Proof. The proof of Lemma 3.15 follows by adapting the well-known sum-check protocol of Lund *et al.* [LFKN92] to the settings of interactive proofs of proximity. Recall that the sum-check protocol is an interactive protocol that enables verification of the a claim of the form:

$$\sum_{x_1,\ldots,x_m \in H} P(x_1,\ldots,x_m) = 0.$$

where P is a low-degree polynomial. The difference between our setting and the classical setting of the sum-check protocol of [LFKN92] is that in the latter the verifier has explicit and direct access to P.[15] In our setting the verifier only has *oracle access* to a function that is *allegedly* a low-degree polynomial. However, we observe that the sum-check protocol can be extended to this setting by having the verifier (1) test that the function is close to a low-degree polynomial P, (2) obtain values from P via self-correction, and (3) run the sum-check protocol as-is with respect to the self-corrected P. The IPP protocol is described in Fig. 3.4, where the prover is denoted by $\mathcal{P}$, the verifier is denoted by $\mathcal{V}$ and all arithmetic is over the field $\mathbb{F}$. (For a high level description of the sum-check protocol, see Section 3.A.5.)

We note that during the run of the IPP the prover sends m degree d univariate polynomial, and the verifier sends m elements in $\mathbb{F}$. Thus, the total communication complexity of the IPP is $O(dm \log |F|)$. The only queries that the verifier performs are for the low degree test and the self-correction, which total in $O(dm \cdot \mathsf{poly}(1/\varepsilon))$ queries.

[14] We note that the property testing upper bound of $\tilde{O}(n^{1-\alpha})$ can be obtained by a verifier that tests for low degree and reads all points in H^m using self correction.

[15] An additional minor difference is that in the [LFKN92] protocol the set H is fixed to $\{0,1\}$, but this is common in the PCP literature (most notably in [BFLS91]).

IPP for TensorSum

Parameters: $\mathbb{F}$ (field), m (dimension), d (individual degree) and $H \subset \mathbb{F}$.

Input: a proximity parameter $\varepsilon \in (0, 1/3)$, and oracle access to a function $P : \mathbb{F}^m \to \mathbb{F}$.

1. $\mathcal{V}$ runs the individual degree d test (see Theorem 3.31) on P with respect to proximity parameter ε, and rejects if the test fails.
2. Let $\nu_0 \stackrel{\text{def}}{=} 0$.
3. For $i \leftarrow 1, \ldots, m$:
   (a) $\mathcal{P}$ sends to $\mathcal{V}$ a degree d univariate polynomial $\tilde{P}_i : \mathbb{F} \to \mathbb{F}$ (by specifying its $d+1$ coefficients), which allegedly equals:
   $$P_i(z) \stackrel{\text{def}}{=} \sum_{x_{i+1},\ldots,x_m \in H} P(r_1, \ldots, r_{i-1}, z, x_{i+1}, \ldots, x_m).$$
   (b) $\mathcal{V}$ verifies that $\sum_{z \in H} \tilde{P}_i(z) = \nu_{i-1}$.
   (c) $\mathcal{V}$ selects uniformly at random $r_i \in_R \mathbb{F}$ and sets $\nu_i \stackrel{\text{def}}{=} \tilde{P}_i(r_i)$.
   (d) If $i \neq m$, then $\mathcal{V}$ sends r_i to $\mathcal{P}$.
4. $\mathcal{V}$ obtains the value of z^* of $P(r_1, \ldots, r_m)$ via self-correction (see Theorem 3.29) with soundness error 0.1.
5. $\mathcal{V}$ verifies that $z^* = \nu_m$.

Figure 3.4: IPP for $\mathsf{TensorSum}_{m,d,\mathbb{F},S,c}$

Completeness. If $P \in \mathsf{TensorSum}$, then the low degree test always passes, and since we have $\sum_{x \in H^m} P(x) = 0$, and the prover supplies the correct polynomials (i.e., $\tilde{P}_i = P_i$ for every $i \in [m]$), the verifier always accepts.

Soundness. Suppose that $P : \mathbb{F}^m \to \mathbb{F}$ is ε-far from $\mathsf{TensorSum}$. Let $\mathcal{P}^*$ be a cheating prover that attempts to convince the verifier of the false statement $P \in \mathsf{TensorSum}$. If P is ε-far from having individual degree d, then the verifier rejects with probability $1/2$. Thus, we focus on the case that P is ε-close to a polynomial P' of individual degree d.

For every $i \in [m]$, let:

$$P'_i(z) \stackrel{\text{def}}{=} \sum_{x_{i+1},\ldots,x_m \in H} P'(r_1, \ldots, r_{i-1}, z, x_{i+1}, \ldots, x_m)$$

(where the values r_i are those sent from the verifier to the prover). The next two claims

relate the polynomials P'_i to the polynomials $\tilde{P}_i$ sent by the prover $\mathcal{P}^*$. Recall that both polynomials depend only on $r_1, \ldots, r_{i-1}$.

Claim 3.15.1. *If $\tilde{P}_1 \equiv P'_1$, then the verifier rejects with probability 1.*

Proof. Observe that $\sum_{x \in H^m} P'(x) \neq 0$ must hold, since otherwise $P \in \mathsf{TensorSum}$. Therefore $\sum_{z \in H} P'_1(z) = 0$, and so, if $\tilde{P}_1 \equiv P'_1$, then the verifier rejects when testing that $\sum_{z \in H} \tilde{P}_1(z) = 0$. □

Claim 3.15.2. *For every $i \in [m-1]$ and every $r_1, \ldots, r_{i-1} \in \mathbb{F}$, if $\tilde{P}_i \not\equiv P'_i$ then, with probability at least $1 - d/|\mathbb{F}|$ over the choice of r_i, if $\tilde{P}_{i+1} \equiv P'_{i+1}$ then the verifier rejects.*

Proof. If $\tilde{P}_{i+1} \equiv P'_{i+1}$ then $\sum_{z \in H} \tilde{P}_{i+1}(z) = \sum_{z \in H} P'_{i+1}(z) = P'_i(r_i)$. Thus, since the polynomials $\tilde{P}_i$ and P'_i differ, with probability at least $1 - d/|\mathbb{F}|$ over the choice of $r_i \in_R \mathbb{F}$ it holds that $\tilde{P}_i(r_i) \neq P'_i(r_i)$, and in this case the verifier will reject when testing whether $\sum_{z \in H} \tilde{P}_{i+1}(z) = \nu_i$, since $\nu_i = \tilde{P}_i(r_i)$. □

By Claim 3.15.3 and an application of the union bound, with probability $1 - dm/|\mathbb{F}|$, if there exists an $i \in [m-1]$ such that $\tilde{P}_i \not\equiv P'_i$ but $\tilde{P}_{i+1} \equiv P'_{i+1}$ then the verifier rejects. By Claim 3.15.1, we can assume that $\tilde{P}_1 \not\equiv P'_1$ and so we need only consider the case that for every $i \in [m]$ it holds that $\tilde{P}_i \not\equiv P'_i$. The following claim shows that also in this case the verifier rejects with probability at least $2/3$. The theorem follows.

Claim 3.15.3. *For every $r_1, \ldots, r_{m-1} \in \mathbb{F}$, if $\tilde{P}_m \not\equiv P'_m$, then the verifier rejects with probability at least $2/3$ (over the choice of r_m and the self-correction procedure).*

Proof. If $\tilde{P}_m \not\equiv P'_m$ then these are two distinct degree d polynomials, which can agree on at most d points. Thus, with probability $1 - d/|\mathbb{F}|$, it holds that $\tilde{P}_m(r_m) \neq P'_m(r_m)$ (over the choice of $r_m \in_R \mathbb{F}$). Now, the self-correction algorithm guarantees that the verifier computes $z^* = P'(r_1, \ldots, r_m) = P'_m(r_m)$ correctly with probability 0.9. In such case, the verifier rejects with probability $1 - d/|\mathbb{F}|$ when testing that $z^* = \tilde{P}_m(r_m)$. It follows that the verifier rejects with probability $0.9 \cdot (1 - d/|\mathbb{F}|) > 2/3$. □

This completes the proof of Lemma 3.15. □

3.4 General Transformations

In this section we show general transformations on MAP proof-systems. In Section 3.4.1 we show general transformations from MAPs with restricted proofs into PT. In Section 3.4.2 we show a general transformation from MAPs that have two-sided error into MAPs that have one-sided error.

3.4.1 From MAP to PT

In this section we show that MAPs with restricted proofs can be emulated by property testers. We show two such results. Theorem 3.18 shows that every MAP that uses a *very short* proof can be emulated by a property tester, and Theorem 3.19 shows that even MAPs with long proofs *in which the verifier's queries are proof oblivious* (see Definition 3.2) can also be emulated. We note that in both constructions the tester may be inefficient in terms of *computational* complexity (even if the original MAP verifier can be implemented efficiently).

Theorem 3.18. *If the property* Π *has an* MAP *verifier that makes* q *queries and uses a proof of length* p*, then* Π *has a property tester that makes* $\tilde{O}(2^p \cdot q)$ *queries. Moreover, if the* MAP *tester has one-sided error, then the resulting property tester has one-sided error.*

Proof. Let V be an MAP verifier for Π with query complexity q and proof complexity p. We start by running the verifier $O(p)$ times using fresh (independent) randomness, but the same proof string, and ruling by majority vote. We obtain an MAP verifier V' for Π that has soundness error $2^{-(p+2)}$, uses $q' \stackrel{\text{def}}{=} O(p \cdot q)$ queries and a proof of length p.

We use V' to construct a property tester T for Π. The tester T, given oracle access to a function f, simply enumerates over all possible 2^p proof strings for V'. For each proof string $w \in \{0,1\}^p$, the tester T emulates V' (using fresh randomness) while feeding it the proof string w, and forwarding its oracle queries to f. If for some string w the verifier accepts, then T accepts. Otherwise, it rejects. Clearly, T has query complexity $2^p \cdot q'$.

If $f \in \Pi$, then there exists a proof string w that will make V' accept, with probability at least $1 - 2^{-(p+2)}$. Therefore, T accepts in this case with probability at least $2/3$. On the other hand, if f is ε-far from Π, then no string w will make V' accept with probability greater than $2^{-(p+2)}$. Thus, by the union bound, T will accept with probability at most $2^p \cdot 2^{-(p+2)} < 1/3$.

The furthermore clause of Theorem 3.18, follows by noting that both the parallel repetition and proof enumeration steps preserve one-sided error. □

The tester of Theorem 3.18 makes $O(p \cdot q)$ queries for every one of the possible 2^p proof strings. However, the fact that these queries were chosen independently (i.e., based on fresh randomness) is not used in the soundness argument. Indeed, for soundness we simply applied a union bound, which would have worked just as well if the queries were not independent (i.e., were determined based on the same randomness). This leads us to consider using the *same* sequence of queries for all of the proofs in the emulation step. The problem that we run into is in the completeness condition. Namely, a sequence of queries that was generated with respect to a particular proof may not be "good" for a different proof. More precisely, if the distribution of queries that the MAP verifier generates (heavily) depends on the proof, then the only guarantee that we have is that the MAP verifier will be correct when emulated with a distribution of queries that matches the specific good proof.[16] Hence, we may indeed have to generate a different sequence of queries for every possible proof string.

[16] For an example of such MAPs, see Theorem 3.7 and Theorem 3.20.

However, as proved in the following theorem, if the tester makes *proof oblivious queries* (see Definition 3.2), then the foregoing problem can be avoided and indeed it suffices to make only one sequence of queries, and reuse this sequence for all the 2^p emulations.

Theorem 3.19. *If the property* Π *has an* MAP *verifier that makes* q proof oblivious queries *and uses a proof of length* p*, then* Π *has a property tester that makes* $O(p \cdot q)$ *queries. Moreover, if the* MAP *verifier has one-sided error, then the resulting property tester has one-sided error.*

Proof. Let V be an MAP verifier for Π with query complexity q and proof complexity p, and let V' be exactly as in the proof of Theorem 3.18 (i.e., an MAP verifier for Π with soundness error $2^{-(p+2)}$, using $q' = O(p \cdot q)$ queries and a proof of length p).

As hinted above, the construction of the property tester T differs from that in Theorem 3.18. The tester T is given oracle access to f. It first emulates V' using an arbitrary (dummy) proof string, denoted w_0, a random string r, and by forwarding V''s queries to f. The key observation here is that the distribution of the queries does not depend on the proof at all, and so an arbitrary proof would suffice for our needs. Thus, T obtains a sequence $\bar{a}_r^f = (a_1, \ldots, a_{q'})$ of answers (corresponding to queries specified by r and the previous answers). Now, T enumerates over all possible 2^p proof strings for V', and for each proof string $w \in \{0,1\}^p$ it emulates V' while feeding it the proof string w, the random string r, and the answer sequence $\bar{a}_r^f$. If for some string w the verifier accepts, then T accepts. Otherwise, it rejects.

If $f \in \Pi$, then there exists a proof string w that will make V' accept with probability at least $2/3$. The key point is that since the distribution of the queries does not depend on w. Hence, the queries actually made by T (using the dummy proof w_0) are identical to those V' would have made using the proof w (and the same randomness as T). Hence, T accepts in this case with probability at least $2/3$ (and in case V' has one-sided error, then T accepts with probability 1). On the other hand, similarly to the proof of Theorem 3.18, if f is ε-far from Π then no string w will make V' accept with probability greater than $2^{-(p+2)}$. Thus, by the union bound, T will accept in this case with probability at most $2^p \cdot 2^{-(p+2)} < 1/3$. □

3.4.2 From Two-Sided Error MAP to One-Sided Error MAP

In this section we show a general result transforming any MAP (which may have two-sided error) into an MAP with *one-sided* error, while incurring only a poly-logarithmic overhead to the query and proof complexities. The construction is based on the ideas introduced in Lautemann's [Lau83] proof that BPP is contained the polynomial hierarchy coupled with the observation that MAPs may have very low randomness complexity (adapted from [GS10b], which in turns follows an idea of Newman [New91]). We note that both the verifier and the proof generation algorithm in this construction may be *inefficient* in the computational complexity sense. (This is a consequence of each one of the two parts of the transformation).

Theorem 3.20. *Let Π be a property of functions $f_n : D_n \to R_n$, where $|R_n| \leq \mathsf{exp}\big(\mathsf{poly}(n)\big)$. If Π has a* two-sided error MAP *with q queries and a proof of length p, then Π has a* one-sided error MAP *with $O(q \cdot \mathsf{polylog}(n))$ queries and a proof of length $O(p + \mathsf{polylog}(n))$.*

We note that typically $|R_n| \leq n$ and that properties for which $|R_n| > \mathsf{exp}(\mathsf{poly}(n))$ seem quite pathological. Before proceeding to the proof of Theorem 3.20, we note that as a direct application of the theorem we obtain the following relation between two-sided error property testers and one-sided error MAP (denoted MAP_1).

Corollary 3.21. *For every function $q : \mathbb{N} \times \mathbb{R}^+ \to \mathbb{N}$ it holds that:*

$$\mathsf{PT}(q) \subseteq \mathsf{MAP}_1(\mathsf{polylog}(n), q \cdot \mathsf{polylog}(n)).$$

The proof of Theorem 3.20 is based on two lemmas. The first, Lemma 3.16, shows that a two-sided error MAP verifier that has low *randomness complexity*, can be transformed into a one-sided error MAP. The proof of this lemma is based on the technique of Lautemann [Lau83]. The second lemma (Lemma 3.17) shows that the Goldreich-Sheffet [GS10b] technique for reducing the randomness of *property testers* can also be used to reduce the randomness of MAP verifiers.

Lemma 3.16. *If the property Π has a* two-sided MAP *verifier that makes q queries, uses a proof of length p, and has randomness complexity r, then Π has a* one-sided MAP *verifier that makes $O(q \cdot r \log r)$ queries and uses a proof of length $O(p + r^2 \log r)$.*

Proof. Following [Lau83], the construction involves two main steps. The first step is a parallel repetition step that significantly reduces both the completeness and soundness errors of the MAP. At this point, almost the entire set of possible random strings lead to accepting inputs that have the property and rejecting inputs that are far from the property. The main observation is that there must exist relatively few "shifts" $s_1, \ldots, s_t$ such that for an input that has the property, for *every* random string r there exists a shift s_i such that $r \oplus s_i$ leads to accepting, whereas if the input is far from the property, then with high probability over the choice of r, no shift will result in accepting. Details follow.

Let $V_{(2)}$ be a *two-sided* error MAP verifier for a property Π with query complexity $q \stackrel{\text{def}}{=} q(n, \varepsilon)$, proof complexity $p \stackrel{\text{def}}{=} p(n)$ and randomness complexity $r \stackrel{\text{def}}{=} r(n, \varepsilon)$. To prove the theorem we construct a *one-sided error* MAP verifier $V_{(1)}$ for Π.

Let $V_{(2)'}$ be the two-sided error MAP obtained by taking the majority of $m = \Theta(\log r)$ repetitions of $V_{(2)}$ using fresh random coins but using *the same proof string* for all repetitions. By the Chernoff bound, this amplification yields both completeness and soundness errors that are at most $\delta \stackrel{\text{def}}{=} 2^{-\Omega(m)}$, which may be made smaller than $\frac{1}{c \cdot rm}$ for any desired constant $c > 0$. Note that $V_{(2)'}$ has query complexity $q' \stackrel{\text{def}}{=} qm$, proof complexity $p' \stackrel{\text{def}}{=} p$, and randomness complexity $r' \stackrel{\text{def}}{=} rm$.

Denote by $V^f_{(2)'}(w; s)$ the (deterministic) output of $V^f_{(2)'}(w)$ when invoked with the random string s. We construct the one-sided error MAP verifier $V_{(1)}$ as follows. The

proof string for $V_{(1)}$ consists of the original proof string w for $V_{(2)}$ as well as a sequence of strings $(s_1, \ldots, s_t)$ each of length r', where $t = \Theta(r)$ such that $\delta^t < 2^{-r'}$ and $\delta t < \frac{1}{3}$. Given the proof string $(w, s_1, \ldots, s_t)$, the verifier $V_{(1)}$ chooses a random string $s \in_R \{0,1\}^{r'}$ and runs $V^f_{(2)'}(w; s \oplus s_i)$ for each $i \in [t]$. If for some $i \in [t]$ the test accepts, then $V_{(1)}$ accepts; otherwise it rejects. The proof and query complexities can be readily verified, and so we proceed to prove the completeness and soundness of $V_{(1)}$.

Completeness. Let $f \in \Pi$ of size n and let $\varepsilon > 0$. Then, by the completeness of $V_{(2)'}$, there exists a proof string w such that $\Pr_{s \in \{0,1\}^{r'}}[V^f_{(2)'}(w; s) = 1] \geq 1 - \delta$. We show that there exists a sequence $(s_1, \ldots, s_t)$ such that $\Pr_{s \in \{0,1\}^{r'}}[V^f_{(1)}(w, s_1, \ldots, s_t; s) = 1] = 1$.

To show that such a sequence $(s_1, \ldots, s_t)$ exists we use the probabilistic method. Specifically, we consider a sequence that is chosen uniformly at random, that is, each $s_i \in_R \{0,1\}^{r'}$. By the union bound,

$$\Pr_{s_1,\ldots,s_t}\left[\exists s \text{ s.t. } \forall i \in [t],\, V^f_{(2)'}(w; s \oplus s_i) = 0\right] \leq \sum_s \Pr_{s_1,\ldots,s_t}\left[\forall i \in [t],\, V^f_{(2)'}(w; s \oplus s_i) = 0\right], \tag{3.6}$$

but since the s_i's are independent, for every $s \in \{0,1\}^{r'}$,

$$\Pr_{s_1,\ldots,s_t}\left[\forall i \in [t],\, V^f_{(2)'}(w; s \oplus s_i) = 0\right] = \prod_{i=1}^{t} \Pr_{s_i}\left[V^f_{(2)'}(w; s \oplus s_i) = 0\right] \leq \delta^t. \tag{3.7}$$

Combining Equations (3.6) and (3.7) we obtain that:

$$\Pr_{s_1,\ldots,s_t}\left[\exists s \text{ s.t. } \forall i \in [t],\, V^f_{(2)'}(w; s \oplus s_i) = 0\right] \leq 2^{r'} \cdot \delta^t < 1.$$

and (zero-error) completeness follows.

Soundness. Let f of size n be ε-far from having the property Π for $\varepsilon > 0$. Then, by the soundness of $V_{(2)'}$, for every proof string w, the verifier $V_{(2)'}$ accepts f with probability at most δ. Hence, by the union bound,

$$\Pr_s\left[\exists i \in [t] \text{ s.t. } V^f_{(2)'}(w; s \oplus s_i) = 1\right] \leq \sum_{i \in [t]} \Pr_s\left[V^f_{(2)'}(w; s \oplus s_i) = 1\right] \leq t \cdot \delta < 1/3$$

and the lemma follows. □

Lemma 3.17. *Let Π be a property of functions $f_n : D_n \to R_n$, where $|R_n| \leq \exp(\mathsf{poly}(n))$. If Π has an MAP verifier that makes q queries, uses a proof of length p, and has randomness complexity r, then Π has an MAP verifier that makes q queries, uses a proof of length p and has randomness complexity $O(\log n)$.*

Proof. The proof follows the proof of [GS10b] with a minor modification to handle the dependence of the verifier on the proof. Namely, using the probabilistic method, we show the existence of a small subset of the random strings that behaves similarly to the entire set.

Let Π be a property of functions $f_n : D_n \to R_n$, where $|R_n| = \exp\big(\mathsf{poly}(n)\big)$ (and where $D_n = [n]$, cf. Section 3.2), and let V be the MAP verifier of the lemma statement. Fix an input length n and let $D \stackrel{\text{def}}{=} D_n$, $R \stackrel{\text{def}}{=} R_n$ and $p \stackrel{\text{def}}{=} p(n)$. Consider a $2^r \times |R|^{|D|} \cdot 2^p$ matrix where the rows correspond to all possible random strings γ used by the verifier and the columns correspond to pairs (f, w) of functions $f : D_n \to R_n$ and possible proofs $w \in \{0,1\}^p$. The entry $(\gamma, (f, w))$ of the matrix corresponds to the output of $V^f(w; \gamma)$, that is, the output of the verifier when given oracle access to f, the proof string w and random coins γ.

Note that for every function $f \in \Pi$, by the completeness of V, there exists a proof string w such that the average of the (f, w) column is at least $2/3$. Similarly, by the soundness of V, for functions that are ε-far from Π and *every* proof string w the average of the (f, w) column is at most $1/3$.

We show that there exists a multi-set, S, of size $\mathsf{poly}(n)$ of the rows such that the average of every column when taken over the rows of S is at most $1/7$-far from the average taken over all rows. Thus, we obtain an MAP verifier that uses only $\log_2 |S| = O(\log n)$ random coins, by simply running the original tester V but with respect to random coins selected uniformly from S (rather than from $\{0,1\}^r$). To obtain soundness and completeness error $1/3$ we use $O(1)$ parallel repetitions.

We use the probabilistic method to show the existence of a small multi-set S as above. Consider a multi-set S of the rows, of size t, chosen uniformly at random and fix some function f and proof string w. By the Chernoff bound, with probability $2^{-\Omega(t)}$ over the choice of S, the average over the rows in S of the (f, w)-column is $1/7$-close to the average over all rows. Thus, by setting $t = \log(|R|^{|D|} \cdot 2^p)$ and applying the union bound, we obtain that there exists a multi-set S as desired.

Since the new verifier selects at random from S, it can be implemented using $\log_2 t$ random coins. We complete the proof by noting that the proof length p can always be made to satisfy $p \leq n$ (since a proof of length n suffices to test any property using only $O(1/\varepsilon)$ queries, see discussion in Section 3.1.2), that the domain size is n and that $|R| \leq \exp(\mathsf{poly}(n))$ (by the hypothesis). □

Theorem 3.20 follows by applying the randomness reducing transformation of Lemma 3.17, and then applying Lemma 3.16 to the resulting MAP verifier.

3.5 An Extremely Hard Property for MAPs

As noted in the introduction, every property has an MAP that uses a proof of length n and makes only $O(1/\varepsilon)$ queries (where the proof is simply the object itself). In contrast, in this section we show that for "almost all" properties Π, every MAP for Π that uses a proof that is even $n/100$ bits long, requires $\Omega(n)$ queries.

Our result is actually slightly stronger. Roughly speaking, we show that for every t, a random property of size 2^t can be tested (without a proof) using $O(t)$ queries, but any MAP that uses a proof of length even $t/100$ must make $\Omega(t)$ queries in order test this property.

In the following we consider properties that are sets of strings rather than functions. We note that a function formulation (as in Definition 3.1) can be easily obtained by mapping every string $x \in \{0,1\}^n$ to the function $f_x : [n] \to \{0,1\}$, defined as $f_x(i) = x_i$.

Theorem 3.22. *Let $t = t(n) < n/10$. Every property $\Pi = \cup_{n\in\mathbb{N}}\Pi_n$ (where $\Pi_n \subseteq \{0,1\}^n$) of size 2^t can be tested with $O(t/\varepsilon)$ queries (without using a proof), but for every $n \in \mathbb{N}$, for 99% of sets $\Pi_n \subseteq \{0,1\}^n$ of size 2^t, it holds that every MAP for testing $\varepsilon < 1/4$ proximity to Π_n that uses a proof of length p must make at least $t - p - O(\log n)$ queries.*

The rest of this section is devoted to the proof of Theorem 3.22, which is inspired by [GGR98, Section 4.1] and uses also ideas from [RVW13, Section 4]. We remark that while Theorem 3.22 holds for almost all properties, finding an *explicit* property for which a similar statement holds is an interesting open question.

The key idea in the proof of Theorem 3.22 is to show that MAPs that use a relatively short proof and make relatively few queries can be represented by a small class of functions. Since this class of functions is small, we argue that a (small) *random* set $S \subseteq \{0,1\}^n$, viewed as a property, will fool every MAP, in the sense that no MAP verifier can distinguish between a random element in S and a random element in $\{0,1\}^n$.

The foregoing intuition is formalized by the following lemma shows that exists a set of *randomized decision trees* (see definition below) such that for every MAP, there exists a subset of the decision trees such that the MAP accepts an input x (with probability at least $2/3$) if and only if at least one of the randomized decision trees accepts x (with probability at least $2/3$).

Lemma 3.18. *Let $\varepsilon \in (0, 1/4)$. For every $n \in \mathbb{N}$ and for every $p, q \leq n$, there exists a class of functions $\mathcal{F}^{(n)}_{p,q}$ of size $2^{(\mathsf{poly}(n)\cdot 2^{p+q})}$ of functions from $\{0,1\}^n$ to $\{0,1\}$, such that the following holds. For every MAP verifier V for testing ε-proximity to $\Pi_n \subseteq \{0,1\}^n$ that uses a proof of length p and q queries, it holds that $I_V \in \mathcal{F}^{(n)}_{p,q}$, where $I_V(x)$ is defined as the indicator function for the event that there exists some $\pi \in \{0,1\}^p$ such that $\Pr[V^x(n, \varepsilon, \pi) = 1] \geq 2/3$.*

Note that the order of quantifiers in Lemma 3.18 is such that the class of functions is the same for *every* MAP verifier (and depends only on p and q). This will be crucial in showing that a random set fools *every* MAP verifier. Also note that if $p+q \ll n$, then the size of $\mathcal{F}$ is quite small relative to the class of all functions from $\{0,1\}^n$ to $\{0,1\}$ (which has size 2^{2^n}).

Proof of Lemma 3.18. To facilitate the proof of Lemma 3.18, it will be useful to describe standard testers (which do not use a proof) as *randomized decision trees*. Our main observation is that, roughly speaking, an MAP can be expressed as an OR of randomized decision trees.

Recall that a randomized decision tree is a model of computation for computing a randomized function $f : \{0,1\}^n \to \{0,1\}$. The randomized decision tree is a rooted ordered binary tree. Each internal vertex of the tree is labeled with a value $i \in \{1,\ldots,n,*\}$ and the leaves of the tree are labeled with 0 or 1. (We think of a node that is labeled with $i \in [n]$ as representing the reading of the i^{th} bit, and of a node that is labeled with $*$ as representing a random coin toss.) Given an input $x \in \{0,1\}^n$, the decision tree is recursively evaluated as follows. If the root's label is $*$, then one of its two children is selected uniformly at random, and we recurse on that child. Otherwise (i.e., $i \in [n]$), if $x_i = 0$, then we recurse on the left subtree, and if $x_i = 1$, then we recurse on the right subtree. Once a leaf is reached, we output the label of that leaf and halt. If T is a randomized decision tree, we denote by $T(x)$ the (random variable that corresponds to) the output of T on input x.

The size of the decision tree is defined as the number of vertices in the tree, and the depth of the tree is defined as the longest path between the root of the tree and one of its leaves. (See [BdW02] for an extensive survey of decision tree complexity.) Let RDT_s be the set of all randomized decision trees of size s. For every $T_1,\ldots,T_t \in \mathsf{RDT}_s$ let $f_{T_1,\ldots,T_t} : \{0,1\}^n \to \{0,1\}$ be the function defined as $f_{T_1,\ldots,T_t}(x) = 1$ if and only if there exists $i \in [t]$ such that $\Pr[T_i(x) = 1] \geq 2/3$. Consider the class of functions

$$\mathcal{F}_{s,t} = \left\{ f_{T_1,\ldots,T_t} \,:\, T_1,\ldots,T_t \in \mathsf{RDT}_s \right\}.$$

We show that $\mathcal{F}_{\mathsf{poly}(n)\cdot 2^q,2^p}$ satisfies the conditions of the lemma.

Let V be an MAP verifier of ε-proximity for Π_n that uses a proof of length p bits, q queries, and r random bits. The main observation is that for every fixed proof string $\pi \in \{0,1\}^p$, the (randomized) decision $V^x(n,\varepsilon,\pi)$ can be expressed as a randomized decision tree $T_{V,\pi}$ of depth $r+q$ (and size 2^{r+q}), which is defined as follows. The first r vertices in every path from the root to a leaf in the tree are labeled by $*$ (these vertices correspond to the random coin tosses of V). Every other internal vertex is labeled by some $i \in [n]$, corresponding to a query to x_i made by V. The two edges leaving every vertex, labeled by 0 and 1, correspond to the actual value of x_i, and these edges lead to a vertex that is labeled by the next query made by V, given the answer x_i to the query i. Given an input x and a random string $\rho \in \{0,1\}^r$, the leaf that is reached by evaluating the decision tree on input x and the random string ρ is labeled with the value $V^x(n,\varepsilon,\pi;\rho)$. (Recall that $V^x(n,\varepsilon,\pi;\rho)$ denotes the output of the verifier V given oracle access to x, direct access to n, ε, π and the random string ρ.) We are interested in $\Pr\left[V^x(n,\varepsilon,\pi) = 1\right]$.

Let $I_V : \{0,1\}^n \to \{0,1\}$ be defined as $I_V(x) = 1$ if and only if there exists $\pi \in \{0,1\}^p$ such that $\Pr[V^x(n,\varepsilon,\pi) = 1] \geq 2/3$. Since the randomized functions $V^x(n,\varepsilon,\pi)$ and $T_{V,\pi}(x)$ are identically distributed, it holds that $I_V \in \mathcal{F}_{2^{r+q},2^p}$.

By Lemma 3.17, we may assume without loss of generality that V has randomness complexity $r = O(\log n)$. The lemma follows by noting that $|\mathsf{RDT}_s| \leq (n+1)^s$ and therefore $|\mathcal{F}_{s,t}| \leq |\mathsf{RDT}_s|^t \leq (n+1)^{s\cdot t}$. □

Before proceeding to the proof of Theorem 3.22, we state a few standard propositions

(Propositions 3.19, 3.20 and 3.22) whose proofs are deferred to Section 3.B.1. We start by noting that sparse properties can be efficiently tested.

Proposition 3.19 (folklore)**.** *Every property $\Pi = \cup_{n\in\mathbb{N}}\Pi_n$ (where $\Pi_n \subseteq \{0,1\}^n$) can be tested by making $O(\log|\Pi_n|/\varepsilon)$ queries (without a proof).*

We note that Proposition 3.19 has standard proofs via learning theory techniques.[17] In Section 3.B.1 we provide an alternative proof that uses the notion of MAPs in a somewhat surprising, but very natural way.

The following (standard) proposition shows that, with high probability, a random n-bit string will be far from any small subset of $\{0,1\}^n$.

Proposition 3.20 (folklore)**.** *For every constant $\varepsilon \in (0,1/4]$ and set $S \subseteq \{0,1\}^n$, it holds that $\Pr_{x\in_R\{0,1\}^n}[x \text{ is } \varepsilon\text{-close to } S] \leq |S| \cdot 2^{-n/8}$.*

For the last claim that we need, recall the definition of a PRG.

Definition 3.21. *A set $S \subseteq \{0,1\}^n$ is called a* pseudorandom generator (PRG) *for fooling a class $\mathcal{F}$ of functions from $\{0,1\}^n$ to $\{0,1\}$ if for every $f \in \mathcal{F}$ it holds that*

$$\left|\Pr_{x\in_R S}[f(x)=1] - \Pr_{x\in_R\{0,1\}^n}[f(x)=1]\right| < 1/10.$$

(note that the choice of the constant $1/10$ is arbitrary.)

The following (well-known) lemma shows that for every class of functions $\mathcal{F}$, a random set of size $O(\log|\mathcal{F}|)$ is a PRG that fools $\mathcal{F}$.

Proposition 3.22 (implicit in [GK92], see also [Gol08, Exercise 8.1])**.** *Let $\mathcal{F}$ be a class of functions from $\{0,1\}^n$ to $\{0,1\}$, of size at most $2^{2^{n/4}}$. Then, 99% of subsets of $\{0,1\}^n$ of size $s = O(\log|\mathcal{F}|)$ are PRGs that fool $\mathcal{F}$.*

We are now ready to prove Theorem 3.22.

Proof of Theorem 3.22. Fix $\epsilon \in (0,1/4)$. Let $t,p,q : \mathbb{N} \to \mathbb{N}$ be functions such that $t = t(n) < n/10$, $p = p(n) \leq n$, $q = q(n) \leq n$ and $t = p + q + O(\log n)$.

Let $S \stackrel{\text{def}}{=} \cup_{n\in\mathbb{N}} S_n$ where for every $n \in \mathbb{N}$, the set $S_n \subseteq \{0,1\}^n$ is a random subset of $\{0,1\}^n$ of size $2^{t(n)}$. By Proposition 3.19, (for any choice of S) the property S can be tested using $O(\log(|S_n|)/\varepsilon) = O(t/\varepsilon)$ queries (without a proof).

Fix $n \in \mathbb{N}$ and let $\mathcal{F}^{(n)}_{p,q}$ be the class of functions of size $2^{(\mathsf{poly}(n)\cdot 2^{p+q})}$ guaranteed by Lemma 3.18, with respect to p and q. Since $O(\log|\mathcal{F}^{(n)}_{p,q}|) = O(2^{p+q}\cdot\mathsf{poly}(n)) = 2^t$, by Proposition 3.22 (applied to the class $\mathcal{F}^{(n)}_{p,q}$), with probability 0.99 over the choice of S_n, it holds that for every $f \in \mathcal{F}^{(n)}_{p,q}$:

$$\left|\Pr_{x\in_R S_n}[f(x)=1] - \Pr_{x\in_R\{0,1\}^n}[f(x)=1]\right| < 1/10. \tag{3.8}$$

[17] Either by an explicit reduction of property testing to learning (see [GGR98, Section 3]), or by applying Occam's razor directly to the testing problem.

Let S_n be a set for which Eq. (3.8) holds and assume toward a contradiction that there exists an MAP verifier V that uses a proof of length p and q queries, and tests ε-proximity to S_n.

By Lemma 3.18, it holds that $I_V \in \mathcal{F}^{(n)}_{p,q}$, where the function I_V is defined as $I_V(x) = 1$ if and only if there exists $\pi \in \{0,1\}^p$ such that $\Pr[V^x(n,\varepsilon,\pi)] \geq 2/3$. We proceed to show that I_V is a distinguisher for the PRG S_n, in contradiction to Eq. (3.8).

By the completeness of the MAP, for every $x \in S_n$ it holds that $I_V(x) = 1$ and therefore

$$\mathop{\mathbf{E}}_{x \in_R S_n}[I_V(x)] = 1.$$

On the other hand, by the soundness of the MAP, for every x that is ε-far from S_n it holds that $I_V(x) = 0$ and so

$$\mathop{\mathbf{E}}_{x \in_R \{0,1\}^n}[I_V(x)] \leq \mathop{\mathbf{E}}_{\substack{x \text{ that is} \\ \varepsilon\text{-far from } S_n}}[I_V(x)] + \Pr_{x \in_R \{0,1\}^n}\left[x \text{ is } \varepsilon\text{-close to } S_n\right] \leq |S_n| \cdot 2^{-n/8} \leq 2^{-\Omega(n)},$$

where the second inequality follows from Proposition 3.20 (and the fact that $I_V(x) = 0$ for every x that is ε-far from S_n), and the last inequality follows from our setting of $t \leq n/10$. Therefore,

$$\mathop{\mathbf{E}}_{x \in_R S_n}[I_V(x)] - \mathop{\mathbf{E}}_{x \in_R \{0,1\}^n}[I_V(x)] \geq 1 - 2^{-\Omega(n)},$$

in contradiction to Eq. (3.8). □

3.6 MAPs for Parametrized Concatenation Problems

In this section we give a scheme for constructing efficient MAPs for *parameterized concatenation problems*. For starters, we review the notion of (non-parameterized) concatenation problems: The k-concatenation problem of a property Π is defined as the property $\Pi^{\times k} \stackrel{\text{def}}{=} \{(x_1,\ldots,x_k) : \forall i \in [k],\ x_i \in \Pi \text{ and } |x_i| = |x_1|\}$. For every $i \in [k]$, we will refer to x_i as the i^{th} block or sub-input.

Concatenation problems (in the context of property testing) were recently studied by Goldreich [Gol14], who showed that the query complexity of the concatenation problem $\Pi^{\times k}$ (of a property Π) is roughly the same as the query complexity of the problem of testing a single instance of Π, regardless of the number of concatenations. More precisely, the query complexity of testing proximity of an input of length $n \cdot k$ (for $\Pi^{\times k}$) is the same, up to a polylogarithmic factor, as the query complexity of testing proximity of an input of length n (for Π), provided that the query complexity of Π increases at least linearly with $1/\varepsilon$ (which is typically the case).

We consider a generalization of the notion of a concatenation problem by allowing the underlying property to depend on some parameter, which may differ between the different blocks. Consider a family of properties $\{\Pi^\alpha\}_{\alpha \in A}$, where α is the parameter and A is some domain. As we shall show, some natural properties can be expressed as

a concatenation $\Pi^{\alpha_1} \times \ldots, \Pi^{\alpha_k}$ of a property Π^{α}, with respect to different values of the parameter. For example, testing whether a given string x has Hamming weight w can be expressed as the question of testing whether x can be partitioned into k blocks such that the i^{th} block has Hamming weight w_i and $\sum_{i\in[k]} w_i = w$. (Other natural examples are reviewed below.)

In this section it will be convenient for us to view the input length $n \in N$, the proximity parameter $\varepsilon \in (0,1)$, and the number of concatenations k as fixed. We note that although we fix n, ε, and k, these parameters should be viewed as generic, and so we allow ourselves to write asymptotic expressions such as $\mathsf{poly}(n)$, $\mathsf{poly}(\varepsilon)$, etc. If $\Pi \subseteq \{0,1\}^n$, then we say that a verifier V is an $\mathsf{MAP}(p,q)$ for Π with respect to proximity ε if V can distinguish between inputs that are in Π and inputs that are ε-far from Π using a proof of length p and q queries. (See the end of Section 3.6.1 for a discussion of the issues involved in providing a uniform treatment of parameterized concatenation problems.)

Additionally, throughout this section we study properties that are more naturally expressed as sets of strings (rather than functions), therefore we present them as such. Note that a function formulation (as in Definition 3.1) can be easily obtained by the (trivial) mapping that maps the string $x \in \Sigma^n$ to the function $f_x : [n] \to \Sigma$ defined as $f_x(i) = x_i$. We proceed to define parameterized concatenation problems.

Definition 3.23. *Let A be a finite set, and $n, k, n/k \in \mathbb{N}$. For every $\alpha \in A$, let $\Pi^{\alpha}_{n/k} \subseteq \{0,1\}^{n/k}$ be a property of n/k-bit strings that is parameterized by α. For every subset $\bar{A} \subseteq A^k$, we say that the property $\Pi^{\bar{A}}_n$ is a* parameterized k-concatenation property *(of n-bit strings), where $\Pi^{\bar{A}}_n$ is defined as*

$$\Pi^{\bar{A}}_n \stackrel{\text{def}}{=} \bigcup_{(\alpha_1,\ldots,\alpha_k)\in\bar{A}} \Pi^{\alpha_1}_{n/k} \times \ldots \times \Pi^{\alpha_k}_{n/k}.$$

If we consider the task of testing $\Pi^{\bar{A}}_n$, it is not a priori clear (for the tester) what value of the parameter α_i to use for each block. This is where MAPs can help us. That is, the proof of proximity will simply tell the MAP verifier the correct value of the parameter for each block. Using this idea, in Section 3.6.1 we construct an MAP for any parameterized concatenation problem. In Sections 3.6.2 to 3.6.3, we demonstrate the applicability of this technique by using it to construct efficient MAPs (which manage to bypass some lower bounds for testers that do not use a proof) for a couple of natural properties:

1. **Approximate Hamming weight:** The first application of our scheme is an efficient MAP for the problem of approximating the Hamming weight of a given string. In this problem, which is parameterized by $w \in [n]$, the tester needs to distinguish between inputs that have Hamming weight exactly w and those that have Hamming weight $\notin [w - \varepsilon n, w + \varepsilon n]$.

   We complement this MAP with a (non-tight) *lower bound* on the MAP complexity of the approximate Hamming weight property. We leave the question of resolving the gap between the upper and lower bounds to future work. See Section 3.6.2.

2. **Graph orientation problems:** In addition, we show an MAP in the graph orientation model (see Section 3.6.3 for details on this model). Specifically, our MAP distinguishes between orientations (of a specific undirected graph) that are Eulerian and those that are far from Eulerian. Our MAP has lower query complexity than the best possible property tester for this problem, and the gap in query complexity increases with the size of the proof. See Section 3.6.3.

Properties with/without distance. Note that all of the explicit properties studied in Section 3.3 are properties of low-degree polynomials and error-correcting codes. The MAPs that we have shown for these properties crucially relied on the fact that these properties have *distance* (i.e., properties wherein every two objects that have the property are far from each other), and moreover, they allow for a local form of self-correction.[18] We note that in contrast, all of the properties that we study in this section are without distance (as is the property of bipartiteness studied in Section 3.7). For example, the Hamming weight property is without distance since there are pairs of strings at distance 2 that have the same Hamming weight.

3.6.1 The Generic Scheme

In this section we show a generic scheme for parameterized concatenation problems.

Theorem 3.23. *Ler $c_1, c_2 \geq 0$ be constants. Let $\Pi_n^{\bar{A}}$ be a parameterized k-concatenation property (of n-bit strings) with respect to A, $\bar{A}$, and $\{\Pi_{n/k}^{\alpha}\}_{\alpha \in A}$, as in Definition 3.23. Suppose that for every $\alpha \in A$, the property $\Pi_{n/k}^{\alpha}$ can be tested with respect to any proximity parameter $\varepsilon' > 0$ (without using a proof) with query complexity $O\left((n/k)^{c_1} \cdot (\varepsilon')^{-c_2}\right)$. Then, the property Π has an MAP, with respect to proximity parameter ε, that uses a proof of length $k \cdot \log |A|$ and has query complexity:*

$$\begin{cases} \tilde{O}\left((n/k)^{c_1} \cdot \varepsilon^{-\max(1,c_2)}\right) & \text{if } c_1 > 0 \text{ and } c_2 \geq 0 \\ \tilde{O}\left((n/k)^{1-1/c_2} \cdot \varepsilon^{-1}\right) & \text{if } c_1 = 0 \text{ and } c_2 \geq 1. \end{cases}$$

Furthermore, if the testers for $\{\Pi_{n/k}^{\alpha}\}_{\alpha \in A}$ have a one-sided error, then the resulting MAP has a one-sided error.

Proof. The key idea is to use the proof in order to "break" the problem of testing property Π into the concatenation problem of testing several sub-properties with smaller inputs. Then, instead of solving each sub-problem independently, we efficiently verify that the (smaller) sub-inputs together are not too far from their corresponding sub-properties.

More specifically, we partition the input x (of length n) into k blocks $x_1, \ldots, x_k$ of length n/k each. If $x \in \Pi_n^{\bar{A}}$, then there must exist $(\alpha_1, \ldots, \alpha_k) \in \bar{A}$ such that $x_i \in \Pi_{n/k}^{\alpha_i}$ for each $i \in [k]$. The proof is simply $(\alpha_1, \ldots, \alpha_k)$; that is, the "hidden" parameter for each

[18] An important natural subset of this type of properties with distance is the set of properties of algebraic objects; see [KS08] for an extensive study of algebraic properties.

sub-property. The verifier, given this alleged proof, checks that indeed $(\alpha_1, \ldots, \alpha_k) \in \bar{A}$ (i.e., the parameterization of the sub-properties is valid), and is then left with the task of ascertaining that the k blocks are not "far" from $\Pi_{n/k}^{\alpha_1} \times \cdots \times \Pi_{n/k}^{\alpha_k}$.

Toward this end, similarly to the approach in [Gol14, Section 5], we note that given an input that is far from $\Pi_{n/k}^{\alpha_1} \times \cdots \times \Pi_{n/k}^{\alpha_k}$, the distance from the property can be either "spread" between all of the sub-inputs, or "concentrated" on a few sub-inputs — or anything in between. The main idea is that if the distance is "concentrated", then the deviation in these sub-inputs must be large, and so, we can detect that such particular sub-inputs do not have their corresponding sub-property by using a test with low query complexity. Since we only read a few bits for this test, we can afford to run it on many sub-inputs (thereby increasing our chance of catching a sub-input that is far from its corresponding sub-property). On the other hand, if the distance is "spread" among the sub-inputs, then it suffices to examine only a few sub-inputs, but for each such sub-input, we need to run a test with high query complexity. Interestingly, in the latter case it is sometimes beneficial for the verifier to simply read the entire block rather than to run the "expensive" tester.

Since the verifier does not know whether it is in one of the extreme situations or anywhere in between, naively we might want to consider the "worst of all worlds" (i.e., small spread and high query complexity per block). We improve upon the performance of the forgoing approach by using the precision sampling technique (originating in Levin [Lev85, last paragraph of Section 9], see also [Gol14, Appendix A.2]), which allows us to deal with all of the possible distributions of the distance economically (specifically, by considering only a logarithmic number of representative distributions). The resulting MAP protocol for parameterized concatenation problems is presented in Fig. 3.1.

Note that the length of the proof, which is $(\alpha_1, \ldots, \alpha_k)$, is bounded by $k \cdot \log |A|$. As for the query complexity, first recall that for any α and $\varepsilon' > 0$, the property $\Pi_{n/k}^{\alpha}$ has a tester with query complexity $T(n/k, \varepsilon') = (n/k)^{c_1} \cdot (\varepsilon')^{-c_2}$. Thus, the total number of queries is at most:

$$O\left(\sum_{j \in [\lceil \log_2 2/\varepsilon \rceil]} \frac{\log(1/\varepsilon)}{2^j \varepsilon} \cdot \log(1/\varepsilon) \cdot T\left(n/k, 2^{-j}\right)\right) = \tilde{O}\left(\frac{(n/k)^{c_1}}{\varepsilon} \sum_{j \in [\lceil \log_2(2/\varepsilon) \rceil]} 2^{j(c_2-1)}\right)$$
$$= \tilde{O}\left((n/k)^{c_1} \varepsilon^{-\max(1,c_2)}\right).$$

For the special case in which $c_1 = 0$, we tighten the analysis. Observe that, without loss of generality, for any proximity parameter ε, it holds that $T(n, \varepsilon) \leq n$ (simply since the tester can always just read the entire input). Therefore, the query complexity is

MAP for the parameterized k-concatenation problem $\Pi_n^{\bar{A}}$

Input: a proximity parameter $\varepsilon > 0$ and oracle access to a string $x \in \{0,1\}^n$.

The Proof:

- The string x is interpreted as a k sub-inputs $x = (x_1, \ldots, x_k) \in (\{0,1\}^{n/k})^k$.
- The proof consists of the parameters for the concatenated problems; namely, the values $(\alpha_1, \ldots, \alpha_k)$ such that $x_i \in \Pi_{n/k}^{\alpha_i}$, for every $i \in [k]$ (such values must exist for $x \in \Pi_n^{\bar{A}}$).

The Verifier:

1. If $(\alpha_1, \ldots, \alpha_k) \notin \bar{A}$, then reject.
2. For every $j \in [\lceil \log_2(2/\varepsilon) \rceil]$, perform the following test:
   (a) Select uniformly at random $O\left(\frac{\log(1/\varepsilon)}{2^j \varepsilon}\right)$ indices in $[k]$. Denote the chosen indices by I.
   (b) For every $i \in I$: Run the $\Pi_{n/k}^{\alpha_i}$ tester $O(\log(1/\varepsilon))$ times on input x_i, with respect to proximity parameter 2^{-j}. Reject if the majority of the tests failed.
3. If all of the previous tests passed, then accept.

Figure 3.1: MAP for Π

bounded in this case by:

$$O\left(\sum_{j \in [\lceil \log_2 2/\varepsilon \rceil]} \frac{\log(1/\varepsilon)}{2^j \varepsilon} \cdot \log(1/\varepsilon) \cdot T\left(n/k, 2^{-j}\right)\right) = \tilde{O}\left(\frac{1}{\varepsilon} \sum_{j \in [\lceil \log_2 2/\varepsilon \rceil]} \min\left(\frac{n/k}{2^j}, 2^{j(c_2-1)}\right)\right)$$
$$\leq \tilde{O}\left(\frac{1}{\varepsilon} \sum_{j \in [\lceil \log_2 2/\varepsilon \rceil]} (n/k)^{1-1/c_2}\right),$$

where the last inequality follows from the fact that $c_2 \geq 1$ (by our assumption) and thus $\min\left(n/k \cdot 2^{-j}, 2^{(c_2-1)j}\right) \leq (n/k)^{1-1/c_2}$. Therefore, the total query complexity in this case is $\tilde{O}\left((n/k)^{1-1/c_2} \cdot \varepsilon^{-1}\right)$.

We proceed to prove the completeness and soundness of the protocol.

Completeness. Suppose that $x \in \Pi_n^{\bar{A}}$ and that $(x_1, \ldots, x_k) \in \Pi_{n/k}^{\alpha_1} \times \ldots \times \Pi_{n/k}^{\alpha_k}$. The tester for each sub-property is invoked $O(\log(1/\varepsilon))$ times in Step (2b) on some $x_i \in \Pi_{n/k}^{\alpha_i}$. Therefore, with probability $1 - \mathsf{poly}(\varepsilon)$ the majority of these invocations will accept. The total number of times that this step is run is at most $O(1/\varepsilon \cdot \log^2(1/\varepsilon))$ and therefore, by the union bound, the MAP verifier accepts with probability at least $2/3$.

Soundness. Suppose that $x \in \{0,1\}^n$ is ε-far from $\Pi_n^{\bar{A}}$. Let $(\alpha_1, \ldots, \alpha_k) \in \bar{A}$ be an alleged proof for the false statement $x \in \Pi_n^{\bar{A}}$ (notice that if $(\alpha_1, \ldots, \alpha_k) \notin \bar{A}$, then the tester immediately rejects). Thus, $x = (x_1, \ldots, x_k) \in (\{0,1\}^{n/k})^k$ is ε-far from $\Pi_{n/k}^{\alpha_1} \times \ldots \times \Pi_{n/k}^{\alpha_k}$ (since otherwise x is ε-close to $\Pi_n^{\bar{A}}$).

The following claim shows that it suffices to consider $O(\log(1/\varepsilon))$ different distributions of the distance between the sub-inputs. Since the proof of the claim is similar to results of [Gol14, Section 5], we defer it to Section 3.B.2).

Claim 3.23.1 (Precision Sampling (cf. [Lev85, last paragraph of Section 9] or [Gol14, Appendix A.2]))**.** *There exists* $j \in [\lceil \log_2 2/\varepsilon \rceil]$ *such that a* $\frac{2^j \varepsilon}{4 \cdot \lceil \log_2(2/\varepsilon) \rceil}$ *fraction of* $x_1, \ldots, x_k$ *are* 2^{-j}*-far from their corresponding sub-properties* $\Pi_{n/k}^{\alpha_1}, \ldots, \Pi_{n/k}^{\alpha_k}$.

Consider the execution of iteration j, where j is the index guaranteed by Claim 3.23.1. In this iteration, since the verifier selects uniformly at random $O\left(\frac{\log(1/\varepsilon)}{2^j \varepsilon}\right)$ indices in $[k]$, with probability at least 0.9, it selects at least one $i \in [k]$ such that x_i is 2^{-j}-far from Π^{α_i}.

Suppose that such an i is indeed selected. Since the base tester for $\Pi_i^{\alpha_i}$ is run with respect to proximity 2^{-j}, it will reject x_i with probability 2/3. Since the test is repeated $O(\log(1/\varepsilon))$ times, the majority of these tests will reject with probability at least 0.9. Thus, the MAP verifier rejects x with probability at least $0.9 \cdot 0.9 \geq 2/3$. □

On providing a uniform treatment. Recall that throughout this section we have fixed n, ε and k. Before proceeding to describe the applications of Theorem 3.23, we shortly discuss issues that arise when considering a uniform (asymptotic) treatment. In some cases, in order to optimize the total complexity (i.e., the sum of the proof complexity and the query complexity) of the MAP in Theorem 3.23, it is beneficial to allow the number k of concatenations to depend on the proximity parameter ε. However, if k depends on ε, then the following two issues arise.

First, notice that if k depends on ε, then the proof string in Theorem 3.23 becomes dependent on ε too, and therefore this protocol does not fall in our definition of MAP (Definition 3.1), which requires a single proof of proximity that works for *every* value of $\varepsilon > 0$. Hence, one can consider a slight relaxation of Definition 3.1 in which we allow the proof of proximity to depend on ε. Since formally such a protocol is not an MAP, we call it an $\mathsf{MAP}_{\mathsf{PDP}}$ (where PDP stands for *proximity dependent proofs*). Note that in an $\mathsf{MAP}_{\mathsf{PDP}}$ both the contents of the proof of proximity, *and its length* may depend on the proximity parameter. See Section 3.2.1 for further discussion of $\mathsf{MAP}_{\mathsf{PDP}}$.

An additional issue that arises when the number of concatenations k depends on ε is that it is unclear how to define a k-concatenation property, as the naive definition that follows Definition 3.23 would make the property itself depend on k, and therefore also on the proximity parameter. While this issue can be overcome for the specific properties that are studied below, doing so in general would be extremely cumbersome, which is the main reason for our non-uniform treatment.

3.6.2 Approximate Hamming Weight

In this section we consider the problem of deciding whether a given string $x \in \{0,1\}^n$ has Hamming weight approximately w. More specifically, we would like a tester that accepts every string $x \in \{0,1\}^n$ that has Hamming weight $w \in [n]$, and rejects strings that have Hamming weight that is ε-far from having weight w. Namely, the tester should reject every string $x \in \{0,1\}^n$ for which $\mathsf{wt}(x) \notin [w - \varepsilon n, w + \varepsilon n]$, where $\mathsf{wt}(x)$ denotes the Hamming weight of x.

More formally, we consider a family of properties $\{\mathsf{Hamming}_n^w\}_w$, indexed by a weight $w \in \{0, \ldots, n\}$. The property $\mathsf{Hamming}_n^w$ is defined as the set that consists of all strings $x \in \{0,1\}^n$ that have Hamming weight exactly w.

By well-known sampling lower bounds (see, e.g., [BYKS01, Theorem 15], improving upon [CEG95]), the query complexity of any property tester (which does not use a proof) is $\Omega\big(\min(n, \varepsilon^{-2})\big)$. Our goal is to use MAPs in order to bypass this lower bound. We remark that $\mathsf{Hamming}^w$ was already studied by [RVW13] who showed a multiple-message IPP for $\mathsf{Hamming}^w$ with complexity $\tilde{O}(\varepsilon^{-1})$ and a *2-message* IPP with complexity $\tilde{O}\left(n^{\frac{1}{3}} \cdot \varepsilon^{-\frac{2}{3}}\right)$. (Note that for $\varepsilon = 1/\sqrt{n}$, the 2-message protocol of [RVW13] has *sublinear* complexity of $\tilde{O}\left(n^{2/3}\right)$, whereas testing without a proof requires $\Omega(n)$ queries.)

Using Theorem 3.23, we show that the performance of the [RVW13] 2-message IPP can be matched by an MAP (i.e., a 1-message IPP), while essentially preserving its complexity.[19] Thus, we show that even a *non-interactive* proof suffices to bypass the property testing lower bound.

More generally, for every constant parameter $\alpha \in (0,1)$, we show that there exists an explicit MAP for Hamming that uses a proof of length $\tilde{O}(n^\alpha)$, and makes at most $\tilde{O}\left(\sqrt{n^{1-\alpha}} \cdot \varepsilon^{-1}\right)$ queries to the input string. For every value of $\alpha \in (0,1)$, there is a range of ε for which the MAP is more efficient than the best possible property tester (which does not use a proof) for Hamming. A comparison of the efficiency of our MAP versus standard property testers, for different values of α, is provided in Table 3.1.

Before we proceed, we note that we actually prove a slightly stronger result. Namely, that for every $k \in [n]$ there is an MAP for Hamming that uses a proof of length $k \cdot \log n$, and makes at most $\tilde{O}\left(\sqrt{n/k} \cdot \varepsilon^{-1}\right)$ queries (where the more restricted statement above is obtained by setting $k = n^\alpha$). In order to minimize the total complexity (i.e., the sum of the proof complexity and the query complexity) of the MAP, we also consider $\mathsf{MAP}_{\mathsf{PDP}}$ verifiers (recall that $\mathsf{MAP}_{\mathsf{PDP}}$ is a slight relaxation of our definition of MAP that allows the proof of proximity to depend on the proximity parameter, see the discussion at the end of Section 3.6.1. With this relaxation, we can set $k = n^{\frac{1}{3}} \cdot \varepsilon^{-\frac{2}{3}}$ to obtain an $\mathsf{MAP}_{\mathsf{PDP}}$ with (total) complexity $\tilde{O}\left(n^{\frac{1}{3}} \cdot \varepsilon^{-\frac{2}{3}}\right)$. See further discussion in Section 3.2.1.

We complement the foregoing upper bound by showing a *lower bound* on the MAP complexity of Hamming. Specifically, we show that every MAP for Hamming that uses

[19] We note that an MAP for approximating the Hamming distance with similar performance was also discovered independently by (Guy) Rothblum *et al.* following the initial publication of [RVW13].

		MAP	
Parameters	Property Testing	Proof Complexity	Query Complexity
General $\alpha \in (0,1)$	$\Theta\left(\min\left(n, \varepsilon^{-2}\right)\right)$	$\tilde{O}(n^{\alpha})$	$\tilde{O}\left(\sqrt{n^{1-\alpha}} \cdot \varepsilon^{-1}\right)$ Improves for $n^{-\frac{1}{2}-\frac{\alpha}{2}} < \varepsilon < n^{-\frac{1}{2}+\frac{\alpha}{2}}$
$\alpha = 0.02$	$\Theta\left(\min\left(n, \varepsilon^{-2}\right)\right)$	$\tilde{O}\left(n^{0.02}\right)$	$\tilde{O}\left(n^{0.49} \cdot \varepsilon^{-1}\right)$ Improves for $n^{-0.51} < \varepsilon < n^{-0.49}$
$\alpha = 2/3$	$\Theta\left(\min\left(n, \varepsilon^{-2}\right)\right)$	$\tilde{O}\left(n^{2/3}\right)$	$\tilde{O}\left(n^{1/6} \cdot \varepsilon^{-1}\right)$ Improves for $n^{-5/6} < \varepsilon < n^{-1/6}$
$\alpha = 0.98$	$\Theta\left(\min\left(n, \varepsilon^{-2}\right)\right)$	$\tilde{O}\left(n^{0.98}\right)$	$\tilde{O}\left(n^{0.01} \cdot \varepsilon^{-1}\right)$ Improves for $n^{-0.99} < \varepsilon < n^{-0.01}$

Table 3.1: The complexity of testing Hamming for different values of α.

a proof of length $p \geq 1$ must use $\Omega\left(\frac{\min(n,\varepsilon^{-2})}{p}\right)$ queries. Note that the two bounds do not match (e.g., for $\varepsilon = 1/\sqrt{n}$ and $p = n^{2/3}$, the upper bound is $\tilde{O}\left(n^{2/3}\right)$ and the lower bound is $\Omega(n^{1/3})$). We leave the question of resolving this gap for future work.

Theorem 3.24. *For every $w \in \{0, \ldots, n\}$, the property $\mathsf{Hamming}_n^w$ has a (two-sided error) MAP, with respect to proximity parameter ε, that uses a proof of length $k \cdot \log n$ and $\tilde{O}\left(\sqrt{n/k} \cdot \varepsilon^{-1}\right)$ queries.*

We remark that by applying Theorem 3.20 to the MAP of Theorem 3.24, we can (somewhat surprisingly) construct a *one-sided error* MAP with proof complexity $O(k \log n + \mathsf{polylog} n)$ and query complexity $\tilde{O}\left(\sqrt{n/k} \cdot \varepsilon^{-1}\right)$. In contrast, the query complexity of every one-sided error property tester for $\mathsf{Hamming}_n^w$ (without a proof) is *linear* in the input size.

Proof of Theorem 3.24. Fix $w \in [n]$. It is well-known (and easy to show, e.g., via the Chernoff bound) that ε-proximity to $\mathsf{Hamming}_n^w$ can be tested, without a proof, using $O(\varepsilon^{-2})$ queries (with a two-sided error). Let

$$\bar{A} \stackrel{\text{def}}{=} \left\{(w_1, \ldots, w_k) \in \{0, \ldots, n/k\}^k \; : \; \sum_{i=1}^{k} w_i = w\right\}.$$

Observe that a string $x = (x_1, \ldots, x_k) \in (\{0,1\}^{n/k})^k$ has Hamming weight w if and only if, for every $i \in [k]$ the string x_i has Hamming weight w_i and $\sum_{i=1}^{k} w_i = w$. Hence,

$$\mathsf{Hamming}_n^w = \bigcup_{(w_1,\ldots,w_k)\in\bar{A}} \mathsf{Hamming}_{n/k}^{w_1} \times \ldots \times \mathsf{Hamming}_{n/k}^{w_k}.$$

The theorem follows from Theorem 3.23 (where $c_1 = 0$ and $c_2 = 2$). □

Relation to TensorSum. The Hamming problem is loosely related to the *Sub-Tensor Sum problem* (see Section 3.3.2), since in both problems we want to compute the sum of the entries of a given input string. In the Sub-Tensor Problem we want an exact answer but are given the string in an error-corrected format (where we think of the input as $f : H^m \to \mathbb{F}$ which is encoded by a low degree polynomial $\hat{f} : \mathbb{F}^m \to \mathbb{F}$ that agrees with f on H^m). In the Hamming problem we do not have the benefit of an error-correcting code but allow an approximate answer.

Next, we show a lower bound on the MAP complexity of the property $\mathsf{Hamming}_n^{n/2}$ (the set of all strings of Hamming weight exactly $n/2$, where n is the length of the string). We note that the lower bound can be extended to $\mathsf{Hamming}_n^w$ for more general values of w by reducing to $\mathsf{Hamming}_n^{n/2}$ using adequate padding (while taking care of the integrality issues that arise). We also note that the lower bound only holds for reasonable complexity measures (which are specified formally below).

The lower bound is proved using our extension of the [BBM11] framework to the MAP model that was established in Section 3.3.2.2. Recall that this extension allows us to prove lower bounds on the complexity of MAPs via MA communication complexity lower bounds. We note that since an MAP lower bound refers to a particular value of ε, it immediately implies a lower bound also on $\mathsf{MAP}_{\mathsf{PDP}}$.

One natural candidate for a communication complexity problem on which we can base our Hamming lower bound is the *Hamming Distance* communication problem, wherein Alice and Bob need to decide whether the Hamming distance of their input strings is equal to a predetermined number. However, as opposed to the MAP lower bounds that we have shown before (e.g., for TensorSum, and EIM), Hamming is a property of non-robust objects; i.e., there is no significant distance between every pair of valid objects. In order to overcome the lack of distance between valid objects in Hamming, we wish to reduce Hamming to an MA communication complexity *gap*-problem wherein the YES-instances and NO-instances are far apart. Indeed, the *Gap Hamming Distance* problem, described next, serves this purpose.

Let $n \in \mathbb{N}$, and let $t, g > 0$. The Gap Hamming Distance problem, denoted by $\mathsf{GHD}_{n,t,g}$, is the promise problem wherein Alice gets as input an n-bit string x, Bob gets as input an n-bit string y, and the players need to decide whether the Hamming distance of their strings is greater than $t+g$ (considered a YES-instance), or smaller than $t-g$ (considered a NO-instance). See Section 3.B.3 for formal definitions and background. By extending a recent result of Gur and Raz [GR13b], we show

Lemma 3.24. *Let $g, n \in \mathbb{N}$ such that $g \leq n$ and $t = \alpha \cdot n$ for some constant $\alpha \in (0,1)$. Then, every MA communication complexity protocol for $\mathsf{GHD}_{n,t,g}$, with proof complexity $p \geq 1$, has communication complexity at least $\Omega\left(\frac{\min\left(n,(n/g)^2\right)}{p}\right)$.*

The proof of Lemma 3.24, which is by a reduction to the result of [GR13b], is presented in Section 3.B.3 (see Corollary 3.33). Equipped with Lemma 3.24, we proceed to prove

the lower bound for $\mathsf{Hamming}_n^w$.

Theorem 3.25. *For every $n \in \mathbb{N}$ and $\varepsilon \stackrel{\text{def}}{=} \varepsilon(n) \in (0, 1/2)$, if $\mathsf{Hamming}_n^{n/2}$ has an MAP with respect to proximity parameter ε, with proof complexity $p = \Omega(\log n)$ and query complexity q such that $p(O(n)) = O(p(n))$ and $q(O(n)) = O(q(n))$, then $p \cdot q = \Omega\left(\min\left(n, \varepsilon^{-2}\right)\right)$.*

We note that our restriction on the form of p and q is satisfied by reasonable functions such as $f(n) = a \cdot n^b$ for any $a, b \geq 0$ as well as for $f(n) = a \cdot \mathsf{polylog}(n)$.

Proof of Theorem 3.25. Throughout the proof we fix the function w as $w(m) \stackrel{\text{def}}{=} m/2$. By Lemma 3.13, if $\mathsf{Hamming}_n^w \in \mathsf{MAP}(p, q)$, then the communication complexity (promise) problem $\mathcal{C}_{\oplus,\varepsilon}^{\mathsf{Hamming}_w}$ has an MA communication complexity protocol with a proof of length p and total communication $2q$, where (following [BBM11]) $\mathcal{C}_{\oplus,\varepsilon}^{\mathsf{Hamming}_w}$ refers to the communication complexity (promise) problem, in which Alice and Bob need to decide whether their inputs have Hamming distance exactly $n/2$ or are ε-far from having such distance. Thus, by Lemma 3.24, the theorem follows by reducing $\mathsf{GHD}_{n,n/2-\varepsilon n,\varepsilon n}$ to $\mathcal{C}_{\oplus,\varepsilon}^{\mathsf{Hamming}_w}$, which is done next. (We stress that this reduction takes place entirely in the context of MA communication complexity.)

We note that both $\mathsf{GHD}_{n,n/2-\varepsilon n,\varepsilon n}$ and $\mathcal{C}_{\oplus,\varepsilon}^{\mathsf{Hamming}_w}$ are communication complexity (promise) problems that refer to the Hamming distance $\Delta(x, y)$ between the inputs x and y (of Alice and Bob, respectively). In $\mathsf{GHD}_{n,n/2-\varepsilon n,\varepsilon n}$ the YES-instances correspond to $\Delta(x, y) \geq n/2$ and the NO-instances correspond to $\Delta(x, y) \leq n/2 - 2\varepsilon n$, whereas in $\mathcal{C}_{\oplus,\varepsilon}^{\mathsf{Hamming}_w}$ the YES-instances correspond to $\Delta(x, y) = n/2$ and the NO-instances correspond to $\Delta(x, y) \notin [n/2 - \varepsilon n, n/2 + \varepsilon n]$.

We proceed to show a reduction from $\mathsf{GHD}_{n,n/2-\varepsilon n,\varepsilon n}$ to $\mathcal{C}_{\oplus,\varepsilon}^{\mathsf{Hamming}_w}$. Since the reduction is between two MA communication complexity problems, we may allow the reduction to make use of a proof string. Specifically, the reduction is given as a proof string an integer $\tilde{d} \in \{0, \ldots, n\}$ that allegedly equals $\Delta(x, y)$, and maps a pair $(x, y) \in \{0,1\}^{n+n}$ to a pair $(x', y') \in \{0,1\}^{2n+2n}$ such that a YES (resp., NO) instance of $\mathsf{GHD}_{n,n/2-\varepsilon n,\varepsilon n}$ is mapped to a YES (resp., NO) instance of $\mathcal{C}_{\oplus,\varepsilon}^{\mathsf{Hamming}_w}$.

The reduction, given input $\tilde{d}$ and (x, y), first checks that $\tilde{d} \geq n/2$ and rejects otherwise (since $\Delta(x, y) < n/2$ does not correspond to a YES instance of $\mathsf{GHD}_{n,n/2-\varepsilon,\varepsilon n}$). Then, the reduction maps the pair $(x, y) \in \{0,1\}^{n+n}$ to the pair $(x', y') \in \{0,1\}^{2n+2n}$ by setting $x' = x \circ 0^n$ and $y' = y \circ 0^{\tilde{d}} 1^{n-\tilde{d}}$. That is, Alice (resp., Bob), given input x (resp., y) and the alleged proof $\tilde{d}$, first checks that $\tilde{d} \geq n/2$ and then computes x' (resp., y'). The parties then run the $\mathcal{C}_{\oplus,\varepsilon}^{\mathsf{Hamming}_w}$ MA communication complexity protocol on input (x', y').

If (x, y) is a YES-instance of $\mathsf{GHD}_{n,n/2-\varepsilon n,\varepsilon n}$ (i.e., $\Delta(x, y) \geq n/2$) and $\tilde{d} = \Delta(x, y)$ (i.e., the provided proof is correct), then

$$\Delta(x', y') = \Delta(x, y) + n - \tilde{d} = n,$$

and so (x', y') is a YES-instance of $\mathcal{C}_{\oplus,\varepsilon}^{\mathsf{Hamming}_w}$. On the other hand, if (x, y) is a NO-instance of $\mathsf{GHD}_{n,n/2-\varepsilon n,\varepsilon n}$ (i.e., $\Delta(x, y) \leq n/2 - 2\varepsilon n$), then for every $\tilde{d} \geq n/2$

$$\Delta(x', y') = \Delta(x, y) + n - \tilde{d} \leq n - 2\varepsilon n$$

and so (x', y') is a NO-instance of $\mathcal{C}_{\oplus,\varepsilon}^{\mathsf{Hamming}_w}$.

Let us spell out how the reduction is used to prove the theorem. Suppose that $\mathsf{Hamming}_w$ is in the class $\mathsf{MAP}(p, q)$, where p and q are as in the hypothesis. Then, by Lemma 3.13, the $\mathcal{C}_{\oplus,\varepsilon}^{\mathsf{Hamming}_w}$ problem has an MA communication complexity protocol with proof complexity p and communication complexity $2q$. Our reduction maps inputs of length n (of $\mathsf{GHD}_{n,n/2-\varepsilon n,\varepsilon n}$) to inputs of length $2n$ (of $\mathcal{C}_{\oplus,\varepsilon}^{\mathsf{Hamming}_w}$), while using an additional proof of length $\log_2 n$. Thus, the reduction implies an MA communication complexity protocol for $\mathsf{GHD}_{n,n/2-\varepsilon n,\varepsilon n}$ with proof complexity $p(2n) + \log_2 n = O(p(n))$ and communication complexity $2q(2n) = O(q(n))$. Hence, by Lemma 3.24, it holds that $p \cdot q = \Omega(\min(n, \varepsilon^{-2}))$. □

3.6.3 Graph Orientation Problems

In this section we apply Theorem 3.23 to the problem of testing graph orientations for being Eulerian in the *graph orientation* model. In the graph orientation model, introduced by Halevy *et al.* [HLNT05], an underlying *directed* graph $G = (V, E)$ with a canonical orientation (i.e., wherein each edge is directed from the vertex with the smaller lexicographical order to the vertex with the larger lexicographical order) is given as an explicit input to the tester, and the actual input, to which the tester only has oracle access, is an orientation $\overrightarrow{G} = \{ d(e) \in \{0, 1\} : e \in E \}$ of G, wherein $d(e)$ represents the direction of the edge e.

Given a property Π_G (parameterized by the fixed directed graph G) of graph orientations, a tester for Π_G is given query access to an orientation of G; that is, every query is an edge $e \in E$, and the answer to the query is the direction of e in G (i.e., $d(e) \in \{0, 1\}$). An orientation $\overrightarrow{G}$ of G is ε-close to Π_G if it can be modified to be in Π_G by inverting the direction of at most an ε-fraction of the edges of G. Note that the distance function in the orientation model naturally depends on the size of the underlying graph. Moreover, the testing algorithm may strongly depend on the structure of the underlying graph. We note that the graph orientation model falls within the standard property testing framework, as a special case of property testing of *massively parameterized* problems (see [New10] for a survey on massively parameterized properties).

We consider the graph orientation property of being *Eulerian*, which was first pointed out by Halevy *et al.* [HLNT07] as a natural property for the graph orientation model. Recall that a directed graph is Eulerian if for every vertex v in the graph, the in-degree of v is equal to its out-degree. If G is a directed graph (with canonical orientation), we denote by Euler_G the property that contains all orientations of G to (directed) Eulerian graphs. While no (non-trivial) upper bound is known for this property, Fischer *et al.* [FLM+12] showed that for general graphs, testing proximity to being Eulerian with 1-*sided error* is

hard. Specifically, They showed that for $G = K_{2,n-2}$ (i.e., the full bipartite graph with 2 vertices on one side, and $n-2$ vertices on the other side), a one-sided error tester for Euler_G must use $\Omega(n)$ queries.

Using Theorem 3.23 we show, for every $\alpha \in (0,1]$, an MAP with 1-sided error for $\mathsf{Euler}_{K_{2,n-2}}$, which uses a proof of length $\tilde{O}(n^{\alpha})$ and $\tilde{O}(n^{1-\alpha}\varepsilon^{-1})$ queries. Hence, we have a smooth (up to poly-logarithmic factors) multiplicative trade-off between the query and proof complexities of the MAP. We note that it seems that using similar techniques, it is possible to obtain, using Theorem 3.23, efficient MAPs for several problems in the graph orientation model.

Formally, let $K_{2,n-2}$ be the graph with a set of vertices $V = \{ v_1, ..., v_n \}$ and a set of edges $E = \{ (v_i, v_j) \ : \ i \in \{1,2\},\ j \in \{ 3, ..., n \} \}$.

Theorem 3.26. *The property* $\mathsf{Euler}_{K_{2,n-2}}$ *has a one-sided error* MAP*, with respect to proximity parameter* ε*, that uses a proof of length* $O(k \cdot \log n)$ *and has query complexity* $\tilde{O}\left(\frac{n}{k} \cdot \varepsilon^{-1}\right)$.

Proof. The main idea is to divide $K_{2,n-2}$ into sub-graphs of equal size, wherein v_1 and v_2 are the only vertices that appear in all sub-graphs. We require that for all $j \in \{ 3, \ldots, n \}$, the in-degree of v_j is equal to its out-degree. However, since v_1 and v_2 appear in all of the sub-graphs, we can allow their in-degree in each subgraph to be different than their out-degree in this subgraph, as long as the sum of their in-degrees is equal to the sum of their out-degrees.

We denote the in-degree of a vertex $v \in K_{2,n-2}$ by $d_{in}(v)$ and the out-degree of $v \in K_{2,n-2}$ by $d_{out}(v)$. We start by considering the following generalization of the $\mathsf{Euler}_{K_{2,n-2}}$ property. For every $a, b \in \mathbb{Z}$, let $\mathsf{Euler}^{(a,b)}_{K_{2,n-2}}$ be the set of all orientations of $K_{2,n-2}$ such that:

1. $d_{in}(v_1) - d_{out}(v_1) = a$.
2. $d_{in}(v_2) - d_{out}(v_2) = b$
3. $d_{in}(v_j) = d_{out}(v_j)$, for all $j \in \{ 3, \ldots, n \}$.

(note that a and b may be negative). Let $\bar{A}$ be the set of all sequences $\big((a_1, b_1), \ldots, (a_k, b_k)\big)$, where $a_i, b_i \in \{-(n-2), \ldots, n-2\}$ for every $i \in [k]$ and for which it holds that $\sum_{i=1}^{k} a_i = 0$ and $\sum_{i=1}^{k} b_i = 0$. Consider the property:

$$\Pi \stackrel{\text{def}}{=} \bigcup_{(a_1,b_1),\ldots,(a_k,b_k) \in \bar{A}} \mathsf{Euler}^{(a_1,b_1)}_{K_{2,n/k-2}} \times \ldots \times \mathsf{Euler}^{(a_k,b_k)}_{K_{2,n/k-2}}.$$

This property contains all sequences of k orientations of the graphs $K_{2,n/k-2}$ such that (1) the vertices on the "large" side have in degree that is equal to their out degree and (2) for the vertices on the "small" sides, the sum, over all graphs, of their in-degree equals the sum of their out-degrees. We note that there is a trivial mapping between Π and $\mathsf{Euler}_{K_{2,n-2}}$ which simply identifies the pair of vertices on the smaller side of graphs in Π as a singe pair of vertices.

By applying Theorem 3.23 with $c_1 = 1$, $c_2 = 0$, and using the trivial tester (that queries the entire orientation) for every subgraph, the property Π has an MAP with proof of length $O(k \cdot \log n)$, and query complexity $\tilde{O}\left(\frac{n}{k} \cdot \varepsilon^{-1}\right)$. By the foregoing discussion, this MAP can be easily modified to work also for the property $\mathsf{Euler}_{K_{2,n-2}}$. □

3.7 Bipartiteness in Bounded Degree Graphs

In this section we consider the problem of testing *bipartiteness* for "rapidly-mixing" graphs in the bounded-degree graph model. In a classical result, Goldreich and Ron [GR99] showed that *any* graph can be tested for bipartiteness in the bounded-degree model, using a tester with query complexity $\tilde{O}(\sqrt{N}/\varepsilon)$, where N is the number of vertices in the tested graph. Goldreich and Ron first consider the (far simpler) case in which there is a promise that the graph is "rapidly-mixing" (see definition below). More recently, Rothblum, Vadhan and Wigderson [RVW13] showed a 2-message IPP for bipartiteness, in the rapidly-mixing case, with communication and query complexities that are $\mathsf{poly}(\log N, \varepsilon^{-1})$.

Roughly speaking, using similar techniques to (the rapidly-mixing case in) [GR99], we construct an MAP protocol for testing bipartiteness of rapidly-mixing graphs, with proof complexity p and query complexity q for every p and q such that $p \cdot q \geq N$. Thus, the query complexity of our MAP improves upon that of the [GR99] bipartiteness tester (which does not use a proof) only if the proof is of length $\omega(\sqrt{N})$. In particular, we obtain an MAP verifier that uses a proof of length $N^{2/3}$ and makes only $N^{1/3}$ queries. In contrast, a lower bound of $\Omega(\sqrt{N})$ for testers (which do not use a proof) was shown by Goldreich and Ron [GR02] (and this lower bound holds also in the rapidly-mixing case).

We leave the questions of (1) extending our result to graphs that are not rapidly-mixing, and (2) obtaining an MAP for bipartiteness with query and proof complexities that are both $o(\sqrt{N})$, for future research.

The Bounded Degree Graph Model. In the bounded degree graph model, introduced by Goldreich and Ron [GR02] (see also [Gol11]), the object that is being tested is a graph $G = (V, E)$ with degree bounded by some constant d. The graph is represented by a function $g : V \times [d] \to V \cup \{\perp\}$ such that $g(u, i) = v$ if v is the i^{th} vertex incident at u, and $g(u, i) = \perp$ if u has less than i neighbors. The distance between two graphs, represented by functions $g, g' : V \times [d] \to V \cup \{\perp\}$ is measured (as usual) as the fraction of pairs (u, i) such that $g(u, i) \neq g'(u, i)$. For further details, see [Gol11].

Rapidly-Mixing Graphs. Let $G = (V, E)$ be graph with degree bounded by d and let $N \stackrel{\text{def}}{=} |V|$. A (lazy) random walk of length ℓ starting at a vertex $s \in V$ is a random walk that involves ℓ steps. At each step, if the walk is currently at vertex v with degree $d_v \leq d$, then the walk continues to each neighbor of v with probability $1/2d$ and stays at v with probability $1 - \frac{d_v}{2d} \geq 1/2$ (a so-called "lazy" step). We say that G is rapidly-mixing if for every $s, t \in V$, the probability that a (lazy) random walk of length $\Omega(\log N)$ that starts in s ends in t, is at least $1/(2N)$ and at most $2/N$. We will use the fact that

in a rapidly-mixing graph $G = (V, E)$, for every vertex $s \in V$ and subset $T \subseteq V$, the probability that a random walk of length $\Omega(\log N)$ that starts at s ends in T, is at least $|T|/(2N)$ and at most $2|T|/N$. We mention the well-known fact that expander graphs are rapidly-mixing.

We proceed to describe our MAP. Actually since we require a promise that the graph is rapidly-mixing, we will need a "promise-problem" variant of the notion of MAP. For sake of brevity we only define this notion implicitly (in the next theorem).

Theorem 3.27. *There exists a probabilistic verifier V that given oracle access to a graph G of size N (in the bounded degree model), and explicit access to N, the degree bound d, a proximity parameter $\varepsilon \in (0, 1)$, and a proof string w of length $k \cdot \log N$, makes at most $\tilde{O}(\frac{N}{k} \cdot \varepsilon^{-2})$ oracle queries, and satisfies the following two conditions:*

1. *(Completeness:) if G is bipartite, then there exists a proof string $w \in \{0,1\}^{k \log N}$ such that $V^G(N, d, \varepsilon, w) = 1$, with probability 1.*

2. *(Soundness:) if G is* rapidly-mixing *and ε-far from every bipartite graph, then for every proof string w, with probability at least $1/2$, it holds that $V^G(N, d, \varepsilon, w) = 0$.*

Note that our tester has a one-sided error.

Proof. We define the parity of a (lazy) random walk as the parity of the number of actual (i.e., non-lazy) steps that take place in it. Loosely speaking, the proof that the graph G is bipartite is a subset $S \subseteq V$ of k vertices that are allegedly on the same side of G. To verify the proof, the verifier selects roughly $O(\log N)$ starting vertices, and takes approximately N/k random walks of length $O(\log N)$ from each starting vertex s. If there exist two random walks that start in s and end in S with different parities, then two corresponding vertices in S must be on different sides and the verifier rejects. Otherwise, the verifier accepts.

Since the graph is rapidly-mixing, the probability that a random walk that starts in s ends in S is roughly $|S|/N$. The key point (which is proved formally below) is that if the graph is far from bipartite, then for many starting vertices, the probability that the random walk ends in S *with parity* 0 (or equivalently, with parity 1) is $\Omega(|S|/N)$. That is, the probability of reaching S with either parity is significant enough. The protocol is presented in Fig. 3.1.

Note that the proof and query complexities are as stated. We proceed to show that completeness and soundness hold.

Completeness. If $G = ((L, R), E)$ is a bipartite graph such that $|L| \geq |R|$, and $S \subseteq L$ is the proof string, then there is no path between two vertices in S that has an odd length. Therefore, for every vertex $s \in V$, there are no two paths with different parities that end in S.

MAP for Bipartiteness of rapidly-mixing graphs (in the bounded degree graph model)

Input: oracle access to a graph $G = (V, E)$, the size $N \stackrel{\text{def}}{=} |V|$ of the graph, a bound d on the maximal degree in G, a proximity parameter $\varepsilon \in (0, 1)$, and a parameter $k \in [N]$.

The Proof:
Let $V = (L, R)$ such that L, R are disjoint independent sets and $|L| \geq |R|$ (such a partition is guaranteed if the graph is bipartite). The proof is an (arbitrary) subset $S \subseteq L$ of size k.

The Verifier:

1. Repeat $O\left(\frac{\log N}{\varepsilon}\right)$ times:
   (a) Select uniformly at random $s \in V$.
   (b) Take $O\left(\frac{N}{k} \cdot \frac{\log N}{\varepsilon}\right)$ (lazy) random walks starting at s, each of length $\ell \stackrel{\text{def}}{=} O(\log N)$.
   (c) Reject if there are two walks that end in S, having different parities.
2. If all of the previous tests passed, then accept.

Figure 3.1: MAP for Bipartiteness of rapidly-mixing graphs

Soundness. Suppose that $G = (V, E)$ is a rapidly-mixing graph of size $N = |V|$ that is ε-far from every bipartite graph and let $S \subseteq V$. For every $v \in V$ and $\sigma \in \{0, 1\}$, let p_v^σ be the probability that a (lazy) random walk of length $\ell = O(\log N)$ that starts at v, ends in S with parity σ. Since the graph is rapidly-mixing, $p_v^0 + p_v^1 \geq \frac{|S|}{2N}$ for every $v \in V$.

The following claim shows that, for an average vertex v, the probability that one random walk that starts at v ends in S with parity 0 *and* a second random walk that starts at v ends in S with parity 1, is roughly $\Omega((|S|/N)^2)$ (i.e., roughly the same as the probability for two random walks that start at v to end in S without any restriction on the parities of the walks).

Claim 3.24.1. $\sum_{v \in V} p_v^0 p_v^1 > \frac{\varepsilon |S|^2}{64 \ell N}$.

Proof. Suppose otherwise. Consider the following partition of the graph into (V_0, V_1) where $V_0 = \{v \in V : p_v^0 \geq p_v^1\}$ and $V_1 = \{v \in V : p_v^1 > p_v^0\}$. Let $E' = E(V_0, V_0) \cup E(V_1, V_1)$ be the set of all internal edges within V_0 and within V_1. We will obtain a contradiction by showing that G is ε-close to the bipartite graph $((V_0, V_1), E \backslash E')$ that is obtained from G by removing all edges in E'.

For every $v \in V$ and $\sigma \in \{0, 1\}$, let $A_{v,m}^\sigma$ denote the event that a (lazy) random walk of length m (where m is a parameter) that starts at v, ends in S with parity σ. In particular, $\Pr[A_{v,\ell}^\sigma] = p_v^\sigma$. Then, for every $\sigma \in \{0, 1\}$ and $v \in V_\sigma$, it holds that

$$p_v^{1-\sigma} \geq \sum_{u \in V_\sigma \text{ s.t. } (v,u) \in E'} \frac{1}{2d} \cdot \Pr[A_{u,\ell-1}^\sigma], \tag{3.9}$$

since a walk from v to S with parity $1-\sigma$ can be obtained by a step to one of the neighbors of v in V_σ (which happens with probability $1/2d$ for each neighbor), and a walk of length $\ell-1$ from this neighbor u to S with parity σ (i.e., the event $A^\sigma_{u,\ell-1}$).

Intuitively, since we expect the number of *lazy* steps in a lazy random walk to be rather large (at least $\ell/2$ in expectation), the probability that the event $A^\sigma_{u,\ell-1}$ occurs is closely related to the probability that the event $A^\sigma_{u,\ell}$ occurs (indeed, we expect the discrepancy in the number of steps to be "hidden" by the (deviation of the number of) lazy steps). The foregoing intuition is formalized by observing that with very high probability at least one lazy step occurs and the probability that $A^\sigma_{u,\ell}$ occurs, conditioned on a specific step being lazy, is equal to the probability that $A^\sigma_{u,\ell-1}$ occurs. Indeed, by the union bound,

$$\begin{aligned}
p^\sigma_u &= \Pr[A^\sigma_{u,\ell}] \\
&\leq \Pr[A^\sigma_{u,\ell} \wedge \text{ no lazy steps in the walk}] + \sum_{i\in[\ell]} \Pr[A^\sigma_{u,\ell} \wedge \text{ the } i^{\text{th}} \text{ step in the walk is lazy}] \\
&\leq \Pr[\text{no lazy steps in the walk}] + \sum_{i\in[\ell]} \Pr[A^\sigma_{u,\ell} \mid \text{the } i^{\text{th}} \text{ step in the walk is lazy}]
\end{aligned}$$

We can bound the first term by $2^{-\ell}$, which by setting $\ell = \log(4n)$, is at most $1/(4N)$. As for the second term, the probability that a random walk of length ℓ from u ends in S with parity σ *conditioned on the i^{th} step being lazy* is equal to the probability that a random walk of length $\ell-1$ from u ends in S with parity σ. Hence

$$p^\sigma_u \leq \frac{1}{4N} + \ell \cdot \Pr[A^\sigma_{u,\ell-1}] \tag{3.10}$$

Using Eq. (3.9) and Eq. (3.10), we obtain that:

$$\begin{aligned}
\sum_{v\in V} p^0_v p^1_v &= \sum_{\sigma\in\{0,1\}} \sum_{v\in V_\sigma} p^\sigma_v p^{1-\sigma}_v \\
&\geq \sum_{\sigma\in\{0,1\}} \sum_{\substack{(v,u)\in E' \\ \text{s.t. } v,u\in V_\sigma}} p^\sigma_v \cdot \frac{\Pr[A^\sigma_{u,\ell-1}]}{2d} \\
&\geq \sum_{\sigma\in\{0,1\}} \sum_{\substack{(v,u)\in E' \\ \text{s.t. } v,u\in V_\sigma}} p^\sigma_v \cdot \frac{1}{2\ell d} \cdot \left(p^\sigma_u - \frac{1}{4N}\right) \\
&\geq |E'| \cdot \frac{1}{2\ell d} \cdot \frac{|S|}{4N} \cdot \frac{|S|}{8N}
\end{aligned}$$

where the last inequality follows from the fact that for every $w \in V_\sigma$ it holds that $p^\sigma_w \geq (p^\sigma_w + p^{1-\sigma}_w)/2 \geq |S|/4n$.

Hence, by our hypothesis, $|E'| \leq \frac{\varepsilon|S|^2}{64\ell N} \cdot \left(\frac{1}{2\ell d} \cdot \frac{|S|}{4N} \cdot \frac{|S|}{8N}\right)^{-1} = \varepsilon dN$. Therefore, by removing an ε fraction of the edges of G we obtain a bipartite graph, in contradiction to our assumption that G is ε-far from bipartite. This concludes the proof of Claim 3.24.1. □

We say that a vertex v is good if $p_v^0 p_v^1 \geq \frac{\varepsilon|S|^2}{128\ell N^2}$. (Intuitively, a vertex v is good if two random walks that start at v are likely to end in S with different parities.) Let $\alpha \in [0,1]$ be the fraction of good vertices in V. By Claim 3.24.1,

$$\frac{\varepsilon|S|^2}{64\ell N} < \sum_{v\in V} p_v^0 p_v^1 = \sum_{v \text{ is good}} p_v^0 p_v^1 + \sum_{v \text{ is not good}} p_v^0 p_v^1 \leq \alpha N \cdot \left(\frac{2|S|}{N}\right)^2 + N \cdot \frac{\varepsilon|S|^2}{128\ell N^2},$$

where the last inequality uses the fact that for every vertex $v \in V$ it holds that $p_v^0 \cdot p_v^1 \leq (p_v^0 + p_v^1)^2 \leq (2|S|/N)^2$. Hence, the fraction of good vertices is at least $\alpha = \Omega(\varepsilon/\log N)$.

Hence, with probability at least 0.9, at least one of the starting vertices s (which were selected in one of the $O(\log N/\varepsilon)$ iterations) is good. Assume that indeed, in one of the iterations a good vertex s is selected. Hence, $p_s^0 p_s^1 \geq \frac{\varepsilon|S|^2}{128\ell N^2}$ and $p_s^0 + p_s^1 \leq \frac{2|S|}{N}$, which implies that $p_s^0, p_s^1 = \Omega\left(\frac{|S|\varepsilon}{N\log N}\right)$. Therefore, since we take $O\left(\frac{N}{|S|} \cdot \frac{\log N}{\varepsilon}\right)$ random walks starting in s, with probability 0.9, there will be at least one walk thats ends in S with parity 0 *and* one walk that ends in S with parity 1. Hence, the tester rejects with probability at least $0.9^2 \geq 1/2$. □

Appendix for Chapter 3

3.A Background

3.A.1 Communication Complexity

Let X and Y be finite sets, and let $f : X \times Y \to \{0,1\}$ be a function. In the *two-party probabilistic communication complexity model* we have two computationally unbounded players, traditionally referred to as Alice and Bob. Both players share a random string. Alice gets as an input $x \in X$. Bob gets as an input $y \in Y$. At the beginning, neither one of the players has any information regarding the input of the other player. Their common goal is to compute the value of $f(x,y)$, while minimizing the communication between them. In each step of the protocol, one of the players sends one bit to the other player. This bit may depend on the player's input, the common random string, as well as on all previous bits communicated between the two players. At the end of the protocol, both players output $f(x,y)$ with high probability.

We say that a given protocol π computes a (possibly partial) function $f : X \times Y \to \{0,1\}$ if for every $x \in X$ and $y \in Y$ with probability at least $2/3$ Alice outputs $f(x,y)$ after interacting with Bob.[20] We define the communication complexity of the protocol $\mathsf{CC}(\pi)$ to be the maximum number of communicated bits in the protocol π when Alice and Bob are given inputs from X and Y respectively. The communication complexity of a function f is defined as:

$$\mathsf{CC}(f) = \min_{\pi \text{ that compute } f} \mathsf{CC}(\pi).$$

For a family of functions $\mathcal{F} = \{f_n : X_n \to Y_n\}_{n\in\mathbb{N}}$ we define the communication complexity of $\mathcal{F}$ as $\mathsf{CC}_n(\mathcal{F}) = \mathsf{CC}(f_n)$.

Set-Disjointness. The (unique) *set-disjointness* problem is the classical communication complexity problem wherein Alice gets an n-bit string x, Bob gets an n-bit string y, and their goal is to decide whether there exists $i \in [n]$ such that $x_i = y_i = 1$. Formally,

[20] In the case of a partial function, we consider only relevant x and y's.

Definition 3.25. *For every $n \in \mathbb{N}$, $\mathsf{DISJ}_n : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ is the communication complexity predicate given by the partial function*

$$\mathsf{DISJ}_n(x,y) = \begin{cases} 1 & if \quad \sum_{i\in[n]} x_i y_i = 0 \\ 0 & if \quad \sum_{i\in[n]} x_i y_i = 1 \end{cases}$$

(where the arithmetic is over the integers).

It is well-known (see [KS92]) that the communication complexity of the *set-disjointness* problem is linear in the size of the inputs.

3.A.2 MA Communication Complexity

In MA *communication complexity protocols*, we have a function $f : X \times Y \to \{0,1\}$ (for some finite sets X, Y), and three computationally unbounded parties: Merlin, Alice, and Bob. The function f is known to all parties. Alice gets as an input $x \in X$. Bob gets as an input $y \in Y$. Merlin sees both x, y but Alice and Bob share a private random string that Merlin cannot see.

At the beginning of an MA *communication complexity protocol*, Merlin, who sees both inputs x and y, sends a proof string $w = w(x,y)$ that asserts that $f(x,y) = 1$ to Alice and Bob. The two players exchanges messages and at the end of the protocol, (say) Alice outputs an answer $z \in \{0,1\}$. Note that the answer may depend on the proof w as well as the input (x,y). For a protocol π, denote by $\pi\big((x,y),w\big)$ the probabilistically generated answer $z \in \{0,1\}$ given by Alice on input (x,y) and proof w.

We define MA *communication complexity protocol* as follows.

Definition 3.26. *An $\mathsf{MA}(c,p)$-communication complexity protocol for f is probabilistic communication complexity protocol π between Alice and Bob in which they both get as input a p-bit proof, they can communicate at most c bits, and the protocol satisfies the following two conditions:*

1. Completeness*: for all $(x,y) \in f^{-1}(1)$, there exists a string $w \in \{0,1\}^p$ such that*

$$\Pr\left[\pi\big((x,y),w\big) = 1\right] \geq 2/3$$

(where the probability is over the common random string).

2. Soundness*: for all $(x,y) \in f^{-1}(0)$ and for any string $w \in \{0,1\}^p$ we have*

$$\Pr\left[\pi\big((x,y),w\big) = 1\right] \leq 1/3$$

(where the probability is over the common random string).

The MA Communication Complexity of Set-Disjointness. Recall that there is a well-known linear lower bound on the communication complexity of the the *set-disjointness* problem (DISJ) (see Section 3.3.1.3 for formal definitions and statement of the lower bound). A decade after the communication complexity of DISJ was settled, Klauck [Kla03, Kla11] showed the following lower bound on the MA communication complexity of set-disjointness (later proved to be tight, by Aaronson and Wigderson [AW09]).

Theorem 3.28. *Every* MA *communication complexity protocol for* DISJ_n *with proof complexity* p *and communication complexity* c *satisfies* $p \cdot c = \Omega(n)$.

3.A.3 Error Correcting Codes

We first introduce codes as objects of fixed length and then give asymptotic variants of the definitions. Let Σ be a finite alphabet. An error-correcting code (over Σ) is an injective function $C : \Sigma^k \to \Sigma^n$ where $k, n \in \mathbb{N}$ and $k < n$. Every element in the range of C is called a codeword. The stretch of the code is n (viewed as a function of k) and the relative distance is defined as d/n, where d is the minimal distance between two (distinct) codewords.

We say that the code C is a t-locally testable code (LTC), where $t : [0, 1] \to \mathbb{N}$, if there exists a probabilistic algorithm T that given oracle access to $w \in \Sigma^n$ and a proximity parameter $\varepsilon > 0$ makes at most $t(\varepsilon)$ queries. The algorithm accepts every codeword with probability 1, and rejects every string that is ε-far from the code with probability at least $1/2$. For further details on LTCs, see [GS06, Gol10b].

We say that the code C, with relative distance δ_0, is a t-locally decodable code (t-LDC), where $t \in \mathbb{N}$, if there exists a constant $\delta \in (0, \delta_0/2)$ called the decoding radius, and a probabilistic algorithm D that given $i \in [k]$ and oracle access to a string $w \in \{0, 1\}^n$ that is δ-close to a codeword $w' = C(m)$ for some $m \in \{0, 1\}^k$, makes at most t queries to the oracle and outputs m_i (i.e., the i^{th} bit of m) with probability at least $2/3$. Moreover, if w is a *codeword*, then the algorithm outputs m_i with probability 1. For further details on LDCs, see [KT00].

An important parameter of both LTCs and LDCs are their query complexities; that is, the number of queries t made to the string w. In both cases we are interested in codes for which the number of queries t is significantly smaller than n. While there are known LTCs with (almost) linear stretch and constant query complexity (i.e., t does not depend on n), obtaining an LDC with constant query complexity and polynomial stretch is a major open problem in coding theory.

We will also consider a relaxation of LDCs, introduced by Ben-Sasson *et al.* [BSGH+06], known as relaxed-LDC. In this variant, the decoder is allowed to abort on corrupted codewords. Indeed, the main advantage of relaxed-LDCs over standard LDCs is that there are known constructions (see [BSGH+06]) of relaxed-LDCs with constant query complexity and almost linear stretch.

Definition 3.27 (relaxed-LDC, adapted from [BSGH+06, Definition 4.5])**.** *We say that the code $C : \Sigma^k \to \Sigma^n$ with relative distance δ_0 is a t-relaxed-LDC if there exists a constant $\delta \in (0, \delta_0/2)$ and a probabilistic algorithm D that, given an integer $i \in [k]$ and oracle access to a string $w \in \Sigma^n$, makes at most t queries and satisfies the following two conditions:*

1. *If $w = C(m)$ is a codeword that encodes the message $m \in \{0,1\}^k$, then D outputs m_i with probability 1.*
2. *If w is δ-close to a codeword $w' = C(m)$, then, with probability at least $2/3$, the decoder D outputs a value $\sigma \in \{m_i, \perp\}$; that is, $\Pr[D^w(i) \in \{m_i, \perp\}] \geq 2/3$.*

We note that our definition differs from the original definition in [BSGH+06] in two ways. The first difference is that [BSGH+06] require an additional, third, condition that we do not need. (However, [BSGH+06] show that a code that satisfies conditions 1 and 2 above can be converted into an "equally good" code that satisfies also the additional third condition.) The second difference is that [BSGH+06] only require that the decoder succeed in decoding valid codewords with probability $2/3$ whereas we require successful decoding with probability 1. Fortunately, the constructions of [BSGH+06] actually satisfy the stronger requirement.

The asymptotic variants of the foregoing definitions are obtained in the natural way by considering families of codes, one for each input length. Let $k : \mathbb{N} \to \mathbb{N}$ be some (sublinear) function an let $\{\Sigma_n\}_{n\in\mathbb{N}}$ be an ensemble of alphabets. A family of codes is an ensemble $\{C_n\}_{n\in\mathbb{N}}$ such that $C_n : (\Sigma_n)^{k(n)} \to (\Sigma_n)^n$ is a code for every $n \in \mathbb{N}$.

We say that the family of codes is a t-LTC for a function $t : \mathbb{N} \times [0,1] \to \mathbb{N}$ if for every $n \in \mathbb{N}$, the code C_n is a $t(n,\cdot)$-LTC. Similarly we say that a family of codes is a t-LDC (resp., relaxed-LDC) for a function $t : \mathbb{N} \to \mathbb{N}$ if for every $n \in \mathbb{N}$, the code C_n is a $t(n)$-LDC (resp., $t(n)$-relaxed-LDC). We sometimes abuse notation and refer to a family of codes as a single code.

3.A.4 Multivariate Polynomials and Low Degree Testing

In this section we recall some important facts on multivariate polynomials (see [Sud95] for a far more detailed introduction). In the following we fix a finite field $\mathbb{F}$ and a dimension m and consider m-variate polynomials over $\mathbb{F}$.

Lemma 3.28 (Schwartz-Zippel Lemma)**.** *Let $P : \mathbb{F}^m \to \mathbb{F}$ be a non-zero polynomial of total degree d. Let $S \subset \mathbb{F}$ and let $r_1, \ldots, r_m$ be selected uniformly at random in S. Then,*

$$\Pr_{r_1,\ldots,r_m \in_R S}[P(r_1, \ldots, r_m) = 0] \leq \frac{d}{|S|}.$$

An immediate corollary of the Schwartz-Zippel Lemma is that two distinct polynomials $P, Q : \mathbb{F}^m \to \mathbb{F}$ of total degree d may agree on at most a $\frac{d}{|\mathbb{F}|}$-fraction of their domain (i.e., $\mathbb{F}^m$).

Theorem 3.29 (Self-Correction Procedure (cf. [GS92, Sud95])**.** *Let $\delta < 1/3$, and $d, m \in \mathbb{N}$. There exists an algorithm that, given $x \in \mathbb{F}^m$ and oracle access to an m-variate function $P : \mathbb{F}^m \to \mathbb{F}$ that is δ-close to a polynomial P' of individual degree d, makes $O(d \cdot m)$ oracle queries and outputs $P'(x)$ with probability $2/3$. Furthermore, if P has total degree t, then given $x \in \mathbb{F}^m$, the algorithm outputs $P(x)$ with probability 1.*

In Theorem 3.29, as well as in the two following theorems, the error probability can be decreased to be an arbitrarily small constant using standard error reduction (while increasing the number of queries by a constant factor).

Theorem 3.30 (Total Degree Test (a.k.a. Low Degree Test) (see [RS96, Sud95, AS03])**.** *Let $\varepsilon \in (0, 1/2)$, $t, m \in \mathbb{N}$. There exists an algorithm that, given oracle access to an m-variate function $P : \mathbb{F}^m \to \mathbb{F}$, makes $O(t \cdot \mathsf{poly}(1/\varepsilon))$ queries and:*

1. *Accepts every function that is a polynomial of* total *degree t with probability 1; and*
2. *Rejects functions that are ε-far from every polynomial of total degree t with probability at least $1/2$.*

We will also need a more refined version of the test that tests the individual degree of the polynomial. Such a test is implicit in [GS06, Section 5.4.2] but for sake of self-containment we provide a full proof via a reduction to the total degree test.

Theorem 3.31 (Individual Degree Test)**.** *Let $d, m \in \mathbb{N}$ such that $dm < |\mathbb{F}|/10$ and $\varepsilon \in (0, 1 - \frac{dm}{|\mathbb{F}|})$. There exists an algorithm that, given oracle access to an m-variate polynomial $P : \mathbb{F}^m \to \mathbb{F}$, makes $O(dm \cdot \mathsf{poly}(1/\varepsilon))$ queries, and:*

1. *Accepts every function that is a polynomial of* individual *degree d with probability 1; and*
2. *Rejects functions that are ε-far from every polynomial of individual degree d with probability at least $1/2$.*

Proof. Given oracle access to the function P, the verifier T first runs the total degree test on P with respect to proximity ε and total degree dm. If the total degree verifier rejects, then T rejects.

If the test succeeds, then for every axis $i \in [m]$, the verifier T chooses at random $r_1, \ldots, r_{i-1}, r_{i+1}, \ldots, r_m \in_R \mathbb{F}$, and runs a univariate degree d test on the polynomial $Q_i(z) \stackrel{\text{def}}{=} P(r_1, \ldots, r_{i-1}, z, r_{i+1}, \ldots, r_m)$ with soundness error $1/10$. If for some axis i the univariate test rejects, then T rejects, otherwise it accepts.

Completeness. Completeness follow from the completeness of the total degree test together with the fact that the restriction of an individual degree d polynomial to any of its axes is a degree d univariate polynomial.

Soundness. Suppose that P is ε-far from every polynomial of individual degree d. If P is ε-far from every *total* degree dm polynomial, then the total degree test rejects with probability $1/2$. Thus, we focus on the case that P is ε-close to a total degree dm polynomial P'. In this case the polynomials P and P' are polynomials of total degree dm and since $\varepsilon < 1 - \frac{dm}{\mathbb{F}}$, by the Schwartz-Zippel lemma, they must be identical. Thus, P is a polynomial of total degree dm.

By the hypothesis, P cannot have individual degree d and therefore, there exists $i \in [m]$ such that $P(x_1, \ldots, x_m)$, as a formal polynomial, has degree $d' > d$ in x_i. Thus, there exist polynomials $P_0, \ldots, P_{d'}$ each of total degree at most dm such that

$$P(x_1, \ldots, x_m) = \sum_{j \in \{0,\ldots,d'\}} P_j(x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_m) \cdot x_i^j$$

and $P_{d'} \not\equiv 0$.

Since $P_{d'}$ is a non-zero polynomial of total degree dm, by the Schwartz-Zippel lemma, it can vanish on only a $\frac{dm}{|\mathbb{F}|}$ fraction of its domain. Thus, when testing the i^{th} axis, with probability $1 - \frac{dm}{|\mathbb{F}|}$, the verifier selects $r_1, \ldots, r_{i-1}, r_{i+1}, \ldots, r_m \in \mathbb{F}$ such that it guaranties that $P_{d'}(r_1, \ldots, r_{i-1}, r_{i+1}, \ldots, r_m)$ does not vanish. In this case, the polynomial $Q(z) \stackrel{\text{def}}{=} P(r_1, \ldots, r_{i-1}, z, r_{i+1}, \ldots, r_m)$ is a degree d' univariate polynomial and the verifier rejects it with probability 0.9. Thus, the verifier rejects with probability at least $0.9^2 > 1/2$. □

3.A.5 The Sum-Check Protocol

In this appendix we provide some background on the sum-check protocol that was first introduced by Lund *et al.* [LFKN92]. Recall that the sum-check protocol is an interactive proof for a statement of the form

$$\sum_{x_1,\ldots,x_m \in H} P(x_1, \ldots, x_m) = 0.$$

where P is a (relatively) low-degree polynomial over a finite field $\mathbb{F}$.

In order to verify that the polynomial P sums to 0 over H^m it suffices to verify that for every $h \in H$, the sum of the sub-tensor $(h, *, \ldots, *)$ equals some value $a_h \in \mathbb{F}$ and that $\sum_{h \in H} a_h = 0$. However, the straightforward recursion (which computes the sum of *every* sub-tensor) will yield a total query complexity of $\Omega(H^m)$.

The sum-check protocol takes a different approach by having the prover convince the verifier of the sum of just a *single* randomly selected sub-tensor (thus, yielding the desired efficiency). More specifically, the verifier asks the prover to specify the sum of all sum-tensors of the form $(z, * \ldots, *)$ for every $z \in \mathbb{F}$ (rather than $z \in H$). A key point is that these sums can be specified by the *low-degree* polynomial:

$$P_1(z) \stackrel{\text{def}}{=} \sum_{x_2,\ldots,x_m \in H} P(z, x_2, \ldots, x_m).$$

Since P_1 has low-degree, if the prover provides a different (low-degree) polynomial $\tilde{P}_1$, then these two polynomials must differ on almost all points in $\mathbb{F}$. Thus, it suffices for the verifier to select at random a point $r \in_R \mathbb{F}$ and to have the prover recursively prove that $\sum_{x_2,\ldots,x_m \in H} P(r_1, x_2, \ldots, x_m) = \tilde{P}_1(r_1)$. Hence, we reduced the m-dimensional TensorSum problem to an $(m-1)$-dimensional TensorSum problem using 2 messages and *no queries*. The recursion terminates when $m = 1$ in which case the verifier can verify the claim directly.

We note that when extending the sum-check protocol to be an IPP, we need to take into account the possibility that P is not low degree but this is handled by using the low degree test (Theorem 3.30) and self-correction (Theorem 3.29).

3.B Proofs and Adaptations of Known Results

In this section we provide proofs and adaptations of known results, which are included here for completeness.

3.B.1 Proofs of Standard Claims from Section 3.5

In this section we provide the missing proofs of the standard claims used in Section 3.5.

Proof of Proposition 3.19. We show that every property $\Pi = \cup_{n\in\mathbb{N}}\Pi_n$ (where $\Pi_n \subseteq \{0,1\}^n$) can be tested by making $O(\log|\Pi_n|/\varepsilon)$ queries. Recall that the lemma can be proved via learning theory techniques, but we provide an alternative proof that makes use of the notion of MAPs.

Consider an MAP for Π in which the proof, of length $\log_2|\Pi_n|$, is an explicit and concise description of the object $x \in \Pi_n$ (e.g., its index with respect to the lexicographical ordering of the strings in Π_n). The verifier can verify the proof by querying the object x at $O(1/\varepsilon)$ locations uniformly at random (and compare the answers to the string reconstructed based on the proof). The lemma follows by noting that this MAP makes *proof-oblivious queries* and applying Theorem 3.19, which guarantees that if Π has an MAP verifier that makes q proof oblivious queries and uses a proof of length p, then Π has a tester that makes $O(p \cdot q)$ queries without using a proof. □

Proof of Proposition 3.20. We show that for every constant $\varepsilon \in (0, 1/4]$ and set $S \subseteq \{0,1\}^n$ it holds that $\Pr_{x\in_R\{0,1\}^n}[x \text{ is } \varepsilon\text{-close to } S] \leq |S| \cdot 2^{-n/8}$. Observe that

$$\begin{aligned}\Pr_{x\in_R\{0,1\}^n}[\exists s \in S \text{ such that } x \text{ is } \varepsilon\text{-close to } s] &\leq \sum_{s\in S} \Pr_{x\in_R\{0,1\}^n}[x \text{ is } \varepsilon\text{-close to } s] \\ &= |S| \cdot \Pr_{x\in_R\{0,1\}^n}[x \text{ has at most } \varepsilon n \text{ 1's}] \\ &\leq |S| \cdot \mathsf{exp}(-2 \cdot (1/4)^2 \cdot n).\end{aligned}$$

where the first inequality follows from the union bound, and the last inequality follows from the Chernoff bound and the fact that $\varepsilon < 1/4$. □

Proof of Proposition 3.22. Let $\mathcal{F}$ be a class of functions of size at most $2^{2^{n/4}}$. We show that 99% of sets of size $O(\log|\mathcal{F}|)$ are PRGs that fool $\mathcal{F}$.

For every set $S \subseteq \{0,1\}^n$ and function $f \in \mathcal{F}$, let $\delta_f(S) = |\Pr_{x\in_R S}[f(x)=1] - \mu_f|$ where $\mu_f \stackrel{\text{def}}{=} \Pr_{x\in_R\{0,1\}^n}[f(x)=1]$. Let $s \in [2^{n/4}]$ be an integer and let S be a random set of size s. Then, for every $f \in \mathcal{F}$ it holds that

$$\Pr_S[\delta_f(S) \geq 1/10] = \Pr_S\left[\left|\Pr_{x\in_R S}[f(x)=1] - \mu_f\right| \geq 1/10\right] \leq 2^{-\Omega(t)},$$

where the last inequality follows from the Chernoff bound.[21] Thus, by the union bound, the probability that for *every* $f \in \mathcal{F}$ it holds that $\delta_f(S) < 1/10$, is at least $|\mathcal{F}| \cdot 2^{-\Omega(s)}$ (where the probability is over the choice of S). The lemma follows by setting $s = \Theta(\log|\mathbb{F}|)$. □

3.B.2 Precision Sampling

Proof of Claim 3.23.1. We show that there exists $j \in [\lceil\log_2 2/\varepsilon\rceil]$ such that a $\frac{2^j\varepsilon}{4\cdot\lceil\log_2(2/\varepsilon)\rceil}$ fraction of $x_1,\ldots,x_k$ are 2^{-j}-far from their corresponding sub-properties $\Pi^{\alpha_1}_{n/k},\ldots,\Pi^{\alpha_k}_{n/k}$.

Let $d \stackrel{\text{def}}{=} \lceil\log_2(2/\varepsilon)\rceil$. Let $\Delta_{\mathsf{REL}}(z,W)$ be defined as the minimal *relative* Hamming distance of z from the set W. For every $j \in [d]$, let

$$S_j \stackrel{\text{def}}{=} \left\{i \in [k] : \Delta_{\mathsf{REL}}\left(x_i, \Pi^{\alpha_i}_{n/k}\right) \in \left(2^{-j}, 2^{-(j-1)}\right]\right\},$$

and let $T = [k]\backslash(\cup_{i\in[d]}S_j)$. Notice that the sets $T, S_1, S_2, \ldots, S_d$ form a partition of the k inputs. Also note that, by our setting of d, for every $i \in T$, it holds that x_i is $\varepsilon/2$-close to $\Pi^{\alpha_i}_{n/k}$.

Suppose towards a contradiction that for every $j \in [d]$ it holds that $|S_j| < \frac{2^j\varepsilon}{4d}\cdot k$. Using the fact that for every $i \in S_j$ it holds that x_i is $2^{-(j-1)}$-close to Π^{α_i}, we get

$$\begin{aligned}
\Delta_{\mathsf{REL}}\left(x, \Pi^{\alpha_1}_{n/k}\times\ldots\times\Pi^{\alpha_k}_{n/k}\right) &\leq \frac{1}{k}\sum_{i=1}^{k}\Delta_{\mathsf{REL}}\left(x_i,\Pi^{\alpha_i}\right)\\
&= \frac{1}{k}\sum_{i\in T}\Delta_{\mathsf{REL}}\left(x_i,\Pi^{\alpha_i}\right) + \frac{1}{k}\sum_{j\in[d]}\sum_{i\in S_j}\Delta_{\mathsf{REL}}\left(x_i,\Pi^{\alpha_i}_{n/k}\right)\\
&\leq \frac{|T|}{k}\cdot\frac{\varepsilon}{2} + \frac{1}{k}\sum_{j\in[d]}2^{-(j-1)}\cdot|S_j|\\
&< \frac{\varepsilon}{2} + \sum_{j\in[d]}\frac{\varepsilon}{2d}\\
&= \varepsilon,
\end{aligned}$$

[21] We note that since the set S is chosen *with repetitions* one cannot directly apply the Chernoff bound. Still, since $s \leq 2^{n/4}$ the probability for a repetition is at most $s^2/2^n \leq 2^{-\Omega(n)}$. Conditioning on an event (i.e., that there are no repetitions) that occurs with probability $1-\delta$ can increase the probability by at most a $1/(1-\delta)$ factor.

contradicting our assumption that x is ε-far from $\Pi^{\bar{A}}$. □

3.B.3 Lower Bound on the MA Communication Complexity of GHD

Let $n \in \mathbb{N}$, and let $t, g > 0$. The Gap Hamming Distance problem is the promise problem wherein Alice gets as input an n-bit string x, Bob gets as input an n-bit string y, and the players need to decide whether the Hamming distance of their strings is greater than $t+g$ (a YES instance), or smaller than $t-g$ (a NO instance). Formally,

Definition 3.29. *The* Gap Hamming Distance *problem is the communication complexity problem of computing the (partial) Boolean function* $\mathsf{GHD}_{n,t,g} : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ *given by*

$$\mathsf{GHD}_{n,t,g}(x,y) = \begin{cases} 1 & if \quad \Delta(x,y) \geq t+g \\ 0 & if \quad \Delta(x,y) \leq t-g \end{cases}.$$

We denote $\mathsf{GHD} \stackrel{\text{def}}{=} \mathsf{GHD}_{n,\frac{n}{2},\sqrt{n}}$.

The (standard) communication complexity of GHD has been studied extensively, and after a long line of work, Chakrabarti and Regev [CR11] have shown the seminal linear lower bound on the communication complexity of GHD (later, the proof was significantly simplified by [Vid11, She11]).

In a subsequent work, Gur and Raz [GR13b] showed the following tight lower bound on the MA communication complexity of GHD.

Theorem 3.32 ([GR13b])**.** *Every* MA *communication complexity protocol for* GHD*, with proof complexity* $p \geq 1$*, has communication complexity at least* $\Omega(n/p)$*.*

We note that the aforementioned lower bound can be extended for general settings of the parameters of the *Gap Hamming Distance* problem. Specifically, we use the fact that the simple reductions in [CR11, Section 4]) are based solely on padding arguments (and thus are robust to MA) to obtain the following corollary.

Corollary 3.33. *Let* $g, n \in \mathbb{N}$ *such that* $g \leq n$*, let* $\alpha \in (0,1)$ *and* $t = \alpha n$*. Then, every* MA *communication complexity protocol for* $\mathsf{GHD}_{n,t,g}$*, with proof complexity* $p \geq 1$*, has communication complexity at least* $\Omega\left(\frac{\min\left(n,(n/g)^2\right)}{p}\right)$.

Chapter 4

Proofs of Proximity for Context-Free Languages and Read-Once Branching Programs

4.1 Introduction

The field of property testing, initiated by Rubinfeld and Sudan [RS96] and Goldreich, Goldwasser and Ron [GGR98], studies a computational model that consists of probabilistic algorithms, called *testers*, that need to decide whether a given object has a certain global property or is far (say, in Hamming distance) from all objects that have the property, based only on a local view of the object.

A line of work [EKR04, BSGH+06, DR06, RVW13, GR13b, FGL14, KR14] has considered the question of designing *proof systems* within the property testing model. The minimal type of such a proof system, which was recently studied by Gur and Rothblum [GR13b], augments the property testing framework by replacing the tester with a *verifier* that receives, in addition to oracle access to the input, also free access to an explicitly given short (i.e., sub-linear length) proof. The guarantee is that for inputs that have the property there exists a proof that makes the verifier accept with high probability, whereas, for inputs that are far from the property, the verifier will reject *every* alleged proof with high probability. These proof systems can be thought of as the NP (or more accurately MA) analogue of *property testing*, and are called *Merlin-Arthur proofs of proximity (*MAP*)*.[1]

A more general notion was considered by Rothblum, Vadhan and Wigderson [RVW13] (prior to [GR13b]). Their proof system, which can be thought of as the IP analogue of property testing, consists of an all powerful (but untrusted) prover who interacts with a verifier that only has oracle access to the input x. The prover tries to convince the verifier

[1] A related notion is that of a *probabilistically checkable proof of proximity (*PCPP*)* [BSGH+06, DR06]. PCPPs differ from MAPs in that the verifier is only given *query* (i.e., oracle) access to the proof, whereas in MAPs, the verifier has free (*explicit*) access to the proof. Hence, PCPPs are a PCP analogue of property testing.

that x has a particular property Π. Here, the guarantee is that for inputs in Π, there exists a prover strategy that will make the verifier accept with high probability, whereas for inputs that are far from Π, the verifier will reject with high probability no matter what prover strategy is employed. The latter proof systems are known as *interactive proofs of proximity (*IPP*s)*.[2]

The focus of this paper is identifying natural classes of properties that are known to be hard to test, but become easy to *verify* using the power of a proof (MAP) or interaction with a prover (IPP).

4.1.1 Our Results

One well-known class of properties that is hard to test is the class of *context-free languages*. Alon *et al.* [AKNS00] showed that there exists a context-free language that requires $\Omega(\sqrt{n})$ queries to test (where here and throughout this work, n denotes the size of the input) and a context-free language that requires $\Omega(n)$ queries to test with *one-sided error*. Furthermore, there are no known (non-trivial) testers for general context-free languages.

Another interesting class is the class of languages that are accepted by small *read-once branching programs (ROBPs)*. Newman [New02] showed that the set of strings accepted by any small width ROBP can be efficiently tested.[3] More specifically, Newman showed that width w ROBPs can be tested using $(2^w/\varepsilon)^{O(w)}$ queries, where ε is the proximity parameter. Bollig [Bol05] showed that Newman's result cannot be extended to polynomial-sized ROBPs, by exhibiting an $O(n^2)$-sized ROBP that requires $\Omega(\sqrt{n})$ queries to test. No (non-trivial) testers for general ROBPs are known for width $\Omega(\sqrt{\log n})$.

In this work we consider the question of constructing *efficient* MAPs and IPPs for these two classes.[4] Here, by "efficient", we mean that *both* the *query complexity* (i.e., the number of queries performed by the verifier to the input) and the *proof complexity* (i.e., the length of the MAP proof) or *communication complexity* (i.e., the amount of communication with the IPP prover) are small and, in particular, sub-linear[5].

Our first pair of results are efficient MAPs for context-free languages and for ROBPs. These MAPs offer a multiplicative trade-off between the query and proof complexities. Here and throughout this work, $n \in \mathbb{N}$ specifies the length of the main input and $\varepsilon \in (0,1)$ denotes the proximity parameter.

[2] Indeed, MAPs can be thought of as a restricted case of IPPs, in which the interaction is limited to a single message sent from the prover to the verifier.

[3] The result in [New02] is stated only for *oblivious* ROBPs but in [Bol05, Section 1.3] it is stated that Newman's result holds also for general *non-oblivious* ROBPs.

[4] To see that these two classes do not contain each other, observe that the language $\{0^i1^j2^i3^j : i,j \geq 1\}$, which is *not* a context-free language [HMU06, Example 7.20], has a $\mathsf{poly}(n)$-width ROBP (which simply counts the number of repeated occurrences of 0, 1, 2 and 3). On the other hand, Kriegal and Waack [KW88] showed that every ROBP for the Dyck_2 language, which is a context-free language, has size $2^{\Omega(n)}$.

[5] As pointed out in [GR13b], if we do not restrict the length of the proof, then *every* property Π can be verified trivially using only a constant amount of queries, by considering an MAP proof that contains a full description of the input.

Theorem 4.1. *For every context-free language $\mathcal{L}$ and every $k = k(n)$ such that $2 \leq k \leq n$, there exists an* MAP *for $\mathcal{L}$ that uses a proof of length $O(k \cdot \log n)$ and has query complexity $O\left(\frac{n}{k} \cdot \varepsilon^{-1}\right)$. Furthermore, the* MAP *has one-sided error.*

Theorem 4.2. *If a language $\mathcal{L}$ is recognized by a size $s = s(n)$ ROBP, then for every $k = k(n)$ such that $2 \leq k \leq n$, there exists an* MAP *for $\mathcal{L}$ that uses a proof of length $O(k \cdot \log s)$ and has query complexity $O\left(\frac{n}{k} \cdot \varepsilon^{-1}\right)$. Furthermore, the* MAP *has one-sided error.*

Hence, by setting $k = \sqrt{n}$, every context-free language and every language accepted by an ROBP of size at most $2^{\mathsf{polylog}(n)}$, has an MAP in which both the proof and query complexity are $\tilde{O}\left(\sqrt{n}\right)$ (w.r.t. constant proximity parameter).

Next, we ask whether the query and proof complexity in Theorems 4.1 and 4.2 can be significantly reduced by allowing more extensive *interaction* between the verifier and the prover (i.e., arbitrary interactive communication rather than just a fixed non-interactive proof). Very relevant to this question is a recent result of [RVW13] by which, loosely speaking, every language in NC (which contains all context-free languages [Ruz81] and languages accepted by small ROBPs[6]) has an IPP with $\tilde{O}(\sqrt{n})$ query and communication complexities. While the [RVW13] result is more general, for context-free languages and ROBPs it achieves roughly the same query and communication complexities as the MAPs in Theorems 4.1 and 4.2, but uses much more interaction (i.e., at least logarithmically many rounds of interaction compared to just a single message in our MAPs).

Using cryptographic assumptions[7], Kalai and Rothblum [KR14] recently showed that there exists a language in NC_1 for which every IPP requires that either the query or communication complexity be $\Omega(\sqrt{n})$. Hence, we cannot hope to improve the [RVW13] result in general. Still, for the special case of context-free languages and ROBPs, we show that we can actually extend the MAP protocols in Theorems 4.1 and 4.2 into highly efficient IPPs with only *poly-logarithmic* complexity (using a sub-logarithmic number of rounds). More generally, our IPPs offer a trade-off between the number of rounds of interaction and the query and communication complexities.

Theorem 4.3. *For every context-free language $\mathcal{L}$, every $k = k(n) \geq 2$ and $r = r(n) \geq 1$ such that $k^r \leq n$, there exists an r-round* IPP *for $\mathcal{L}$ with communication complexity $O\left((rk \log n) \cdot \varepsilon^{-1}\right)$ and query complexity $O\left(\frac{n}{k^r} \cdot \varepsilon^{-1}\right)$. Furthermore, the* IPP *is public-coin and has one-sided error.*

Theorem 4.4. *If a language $\mathcal{L}$ is recognized by a size $s = s(n)$ ROBP, then for every $k = k(n) \geq 2$ and $r = r(n) \geq 1$ such that $k^r \leq n$, there exists an r-round* IPP *for $\mathcal{L}$ with communication complexity $O\left((rk \log s) \cdot \varepsilon^{-1}\right)$ and query complexity $O\left(\frac{n}{k^r} \cdot \varepsilon^{-1}\right)$. Furthermore, the* IPP *is public-coin and has one-sided error.*

[6] See Section 4.B for a discussion on why languages accepted by ROBPs can be computed in small depth.

[7] A sufficient assumption for [KR14] is the existence of (length-doubling) PRGs that can be computed in NC_1 and whose output cannot be distinguished from random by circuits of size $2^{o(n)}$.

(Interestingly, and in contrast to Theorems 4.1 and 4.2, here the communication complexity also depends on the proximity parameter ε.) In particular, by setting $k = \log n$ and $r = \frac{\log n}{\log \log n}$, we obtain IPPs for context-free languages and size $2^{\mathsf{polylog}(n)}$ ROBPs, with a sub-logarithmic number of rounds, constant query complexity, and poly-logarithmic communication complexity (w.r.t. constant proximity parameter).

A Remark on Computational Complexity. Following the property testing literature, we view the query complexity and the proof complexity (resp., communication complexity) as the primary resources of an MAP (resp., IPP). Still, the running time of the verifier and of the prover are also important resources. The proofs/provers in our MAPs and IPPs are indeed efficient; that is, polynomial in the main input x (and in the case of ROBPs also in the size of the ROBP).

As for our verifiers, those in Theorems 4.1 and 4.3 run in polynomial time (i.e., $\mathsf{poly}(|x|)$ time) rather than in *sub-linear* time as one might hope. However, by increasing the round complexity in Theorem 4.3 by a poly-logarithmic factor, we can obtain an IPP with sub-linear time verification. Constructing an MAP for context-free languages with sub-linear time verification remains an interesting open question. The verifiers in Theorems 4.2 and 4.4 run in *sub-linear time* if they are given a *suitable* (natural) representation of the ROBP.[8] See the technical sections (specifically Remark 4.7 and Remark 4.16) for further details.

Improved Results for Specific Languages. The paradigm used for the general results in Theorems 4.1-4.4 can be extended to yield better results for specific languages. A notable class of languages for which we obtain such an improvement is the class of languages of balanced parentheses expressions (a.k.a the Dyck languages), which are context-free languages, for which Parnas *et al.* [PRR01] showed a lower bound of $\widetilde{\Omega}(n^{1/11})$ for ordinary testers. Using special properties of the Dyck languages, we can improve on the general result in Theorem 4.1 in this special case and obtain a somewhat more efficient MAP for the Dyck languages. See details in Section 4.4.3.

4.1.2 Proof Overview

The proofs of Theorems 4.1 and 4.2 (i.e., the MAP results) will follow (roughly) as special cases of the proofs of Theorems 4.3 and 4.4 (i.e., the IPP results), respectively. Hence, in this overview we focus on the proofs of Theorems 4.3 and 4.4, while explaining how to derive Theorems 4.1 and 4.2 as special cases.

The proofs of Theorems 4.3 and 4.4 share a common theme: For $\mathcal{L}$ that is either a context-free language or is accepted by a ROBP, we show that every input $x \in \mathcal{L}$ can be broken-down into k sub-problems (related to $\mathcal{L}$) such that the following holds:

[8] Indeed, the running time of the verifier crucially relies on the specific representation of the ROBP. We remark that there are other natural representations of ROBPs than the one we use, and for some of these representations obtaining sub-linear running time may not be feasible.

1. On the one hand, if $x \in \mathcal{L}$, then there exists (1) a partition of $[n]$ into sets $S_1, \ldots, S_k$ (each of size roughly n/k); and (2) languages $\mathcal{L}_1, \ldots, \mathcal{L}_k$ such that both (1) and (2) have a concise representation, and, for every $i \in [k]$, the projection of x on S_i, denoted $x[S_i]$, is in the language $\mathcal{L}_i$. Furthermore, if $\mathcal{L}$ is a context-free language (resp., accepted by an ROBP), then the languages $\mathcal{L}_1, \ldots, \mathcal{L}_k$ are all "variants" of context-free languages[9] (resp., accepted by ROBPs).

2. On the other hand, if x is "far" from $\mathcal{L}$, then for every concise representation of a partition $S_1, \ldots, S_k$ of $[n]$ and languages $\mathcal{L}_1, \ldots, \mathcal{L}_k$ (of the type used in 1), for an average $i \in [k]$, it holds that $x[S_i]$ is proportionally "far" from $\mathcal{L}_i$.

By design, the partition $S_1, \ldots, S_k$ as well as the corresponding languages $\mathcal{L}_1, \ldots, \mathcal{L}_k$ depend on the entire input x, and so the verifier (who only has query access to x) cannot generate them by itself. Instead, the concise representation of $S_1, \ldots, S_k$ and $\mathcal{L}_1, \ldots, \mathcal{L}_k$ will be specified by the prover (as a single message in the case of an IPP, or as the entire proof string in the case of an MAP).

Given the latter, we construct an MAP as follows. The MAP verifier selects at random a small subset $I \subseteq [k]$ and, for every $i \in I$, reads *all* of $x[S_i]$ (which is of length roughly n/k) and checks that $x[S_i] \in \mathcal{L}_i$. Indeed, by the two foregoing conditions, if $x \in \mathcal{L}$, then $x[S_i] \in \mathcal{L}_i$ for every $i \in [k]$, whereas if x is "far" from $\mathcal{L}$, then, by an averaging argument, for many $i \in [k]$, it holds that $x[S_i]$ is proportionally "far" from $\mathcal{L}_i$ (and in particular $x[S_i] \notin \mathcal{L}_i$), and the verifier will reject.

A natural approach for extending the foregoing MAP to an IPP is to have the verifier send the set I (where I is chosen at random as in the MAP) to the prover, and then *recursively* run $|I|$ IPP protocols to check that $x[S_i]$ is close to $\mathcal{L}_i$, for every $i \in I$. In each recursive call the input shrinks by (roughly) a factor of k. After the recursion reaches depth r, where r is a predetermined bound on the number of rounds, the verifier can simply read its entire current input (of length $O(n/k^r)$) and decide whether to accept or reject.

The foregoing approach indeed works, but because there is more than one recursive call in each round, the complexity of the resulting IPP depends *exponentially* on the number of rounds r. Instead, we use a more economical approach, which avoids the exponential dependence on r, based on the notion of a *proximity oblivious tester* [GR11]. Recall that a proximity oblivious tester for a property Π is a tester that does not receive the proximity parameter ε as input and is only required to reject inputs that are ε-far from Π with probability proportional to ε (rather than probability 2/3). To present a more economical recursion, the IPP that we design is similarly "proximity oblivious". The idea is to have the verifier select at random only a single index $i \in [k]$, send i to the prover, and then have the two parties recursively run an IPP protocol for verifying that $x[S_i]$ is close to $\mathcal{L}_i$. Indeed, if $x \in \mathcal{L}$ then $x[S_i] \in \mathcal{L}_i$, whereas if x is ε-far from $\mathcal{L}$, then, since i was chosen at random, on the average $x[S_i]$ is ε-far from $\mathcal{L}_i$, and therefore, by inductive

[9] If $\mathcal{L}$ is a context-free language, then the languages $\mathcal{L}_1, \ldots, \mathcal{L}_k$ will be variants of context-free languages, which we call "partial derivation languages". However, if $\mathcal{L}$ is accepted by an ROBP, then the languages $\mathcal{L}_1, \ldots, \mathcal{L}_k$ are also accepted by (different) ROBPs.

reasoning, the verifier will reject with probability ε. To obtain constant soundness we can just repeat[10] the entire proximity oblivious protocol $O(1/\varepsilon)$ times in parallel.

This concludes the high-level description of our MAPs and IPPs. Of course, the way in which the partition is generated is quite different in the case of context-free languages and in the case of ROBP, and different technical problems arise in each case. In the following subsections we discuss the specific details. In Section 4.1.2.1 we give an overview of how to partition read-once branching programs. Partitioning context-free languages is more involved, and so, in Section 4.1.2.2, as a warm-up, we first consider partitioning into *two parts* (i.e., $k = 2$). Then, in Section 4.1.2.3 we show how to extend the technique to *multiple parts* (i.e., general $k \geq 2$).

4.1.2.1 Partitioning ROBPs

Recall that a branching program on n variables is a directed acyclic graph with a unique source vertex with in-degree 0 and (possibly) multiple sink vertices with out-degree 0. Each sink vertex is labeled with either 0 (i.e., *reject*) or 1 (i.e., *accept*). Each non-sink vertex is labeled by an index $i \in [n]$ and has exactly 2 outgoing edges, which are labeled by 0 and 1. The output of the branching program B on input $x \in \{0,1\}^n$, denoted $B(x)$, is computed in a natural way by starting at the source vertex and taking a walk such that at a vertex labeled by $i \in [n]$, we traverse the outgoing edge labeled by x_i. Once a sink is reached, we output its label. The branching program is *read-once* (ROBP for short) if along every path from source to sink, every index ($i \in [n]$) appears at most once. The *size* of a branching program B, denoted $|B|$, is the number of vertices in it.

For any fixed ROBP B, we construct an IPP (and an MAP, which is a special case of the IPP) for the language accepted by B, denoted $\mathcal{L}_B \stackrel{\text{def}}{=} \{x \in \{0,1\}^n : \ B(x) = 1\}$. In this overview, we make a simplifying assumption that B is both *layered* and *ordered* (a.k.a., an *ordered binary decision diagram* or OBDD). That is, we assume that the vertices of B are partitioned into $n+1$ layers such that, for every $i \in [n]$, edges only go from layer i to layer $i+1$; and vertices in layer i are labeled by the index i (i.e., the ROBP reads its input "in order").

The key idea, which enables the IPP verifier to generate the aforementioned partition $S_1, \ldots, S_k$ (together with the corresponding languages), is to have the prover specify k evenly-spaced vertices along the accepting path corresponding to the input $x \in \mathcal{L}_B$. More specifically, observe that x induces a path $\varphi_0 \to \varphi_1 \to \cdots \to \varphi_n$ from the start vertex φ_0 to some accepting sink φ_n. The prover sends to the verifier a subsequence of this walk, specifically the subsequence $\varphi_{n/k}, \ldots, \varphi_{i \cdot n/k}, \ldots, \varphi_n$.

Given the subsequence, we can reduce the problem of verifying that there exists a path of length n from φ_0 to φ_n to verifying that there exists a path of length n/k between each pair of consecutive vertices in the sequence $\varphi_0, \varphi_{n/k}, \ldots, \varphi_{i \cdot n/k}, \ldots, \varphi_n$. In other words, for every $i \in [k]$ we consider the ROBP B_i that consists only of layers $(i-1) \cdot n/k$ up to $i \cdot n/k$ of B, with the starting state $\varphi_{(i-1) \cdot n/k}$ and the (only) accepting state $\varphi_{i \cdot n/k}$. Verifying

[10] As expected, parallel repetition reduces the soundness error of IPPs at an exponential rate. See Section 4.A for details.

that $x \in \mathcal{L}_B$ can be reduced to verifying that $x[S_i] \in \mathcal{L}_{B_i}$, for every $i \in [k]$, where $S_i \subseteq [n]$ is the set of coordinates of x that are read by B_i and $\mathcal{L}_{B_i} \stackrel{\text{def}}{=} \{z \in \{0,1\}^{n/k} : B_i(z) = 1\}$. Moreover, since $S_1, \ldots, S_k$ is a partition of $[n]$, if x is ε-far from $\mathcal{L}_B$, then $x[S_i]$ is ε-far from $\mathcal{L}_{B_i}$, for an average $i \in [k]$. Hence, we can follow the high-level outline that was suggested in Section 4.1.2; that is, the IPP verifier selects $i \in [k]$ at random, sends i to the prover, and then the two parties recursively run an IPP protocol to verify that $x[S_i]$ is close to the $\mathcal{L}_{B_i}$.

The foregoing intuition almost works but there is a subtle problem: What if the message sent by a *cheating* prover is such that $\mathcal{L}_{B_{i^*}}$ is empty, for some $i^* \in [k]$. This corresponds to a situation in which the branching program B contains no path from $\varphi_{(i^*-1)\cdot n/k}$ to $\varphi_{i^* \cdot n/k}$. In such case, with high probability (i.e., if the verifier chooses i such that $i \neq i^*$) the verifier, as described so far, will not notice this fact and may accept inputs that are far from $\mathcal{L}_B$.

We overcome this difficulty by observing that when the verifier interacts with the honest prover, it holds that $x[S_i] \in \mathcal{L}_{B_i}$ for every $i \in [k]$, and therefore $\mathcal{L}_{B_i} \neq \emptyset$. Hence, we can have the verifier explicitly check that $\mathcal{L}_{B_i} \neq \emptyset$ for *every* $i \in [k]$ (i.e., that there exists *some* input that leads from $\varphi_{(i-1)\cdot n/k}$ to $\varphi_{i \cdot n/k}$ in B). This check requires direct and full access to the branching program B (which is fixed) but does *not* require any queries to the input x, and so we can perform it for *every*[11] $i \in [k]$.

Given this additional check, we can show that the foregoing IPP works. To do so, we argue by induction on the number of rounds that if the input x is ε-far from $\mathcal{L}$ then the verifier rejects with probability at least ε. Indeed, if x is ε-far from $\mathcal{L}_B$, then in the first round we have that:

$$\begin{aligned}
\Pr\left[\text{Verifier for } \mathcal{L}_B \text{ rejects } x\right] &= \mathop{\mathbf{E}}_{i}\left[\Pr\left[\text{Verifier for } \mathcal{L}_{B_i} \text{ rejects } x[S_i]\right]\right] \\
&\geq \mathop{\mathbf{E}}_{i}\left[\varepsilon_i\right] \\
&\geq \varepsilon,
\end{aligned}$$

where ε_i denotes the relative distance of $x[S_i]$ from $\mathcal{L}_{B_i}$, for every $i \in [k]$, and the first inequality follows from the induction hypothesis.

We remark that when dealing with general ROBPs, rather than OBDDs, there are several additional technical difficulties. In particular, since B is not layered, we have to modify our definition of B_i (which previously consisted of layers $(i-1) \cdot n/k$ to $i \cdot n/k$ of B). A natural approach is to define B_i to consist of all paths (in B) of length n/k starting at $\varphi_{(i-1)\cdot n/k}$.[12] The difficulty is that B_i may depend on many, possibly even all, of the bits of x (since different paths may look at different bits), rather than just n/k bits (as was the case for OBDDs). Hence, the input does not necessarily shrink in the recursive step. Nevertheless, we resolve this issue by showing that the *effective length* of

[11] However, this check does increase the running time of the verifier (which we view as a secondary resource) to $\mathsf{poly}(|B|)$. This computation can be minimized by using a pre-processing step in which we compute a $|B| \times |B|$-sized table whose $(v,u)^{\text{th}}$ entry says whether the vertices v and u are connected in B.

[12] The actual definition of B_i that we use is different. See Section 4.3 (in particular Footnote 18).

the input, which is the number of bits that need to be read in order to determine whether the ROBP accepts, does shrink, and this suffices to make progress in the recursion. For further details, see Section 4.3.

4.1.2.2 Partitioning Context-Free Languages into Two Parts

Recall that a *context-free grammar* is a tuple $G = (V, \Sigma, R, A_{\mathsf{start}})$, where $V = \{A_1, A_2, \dots\}$ denotes a (finite) set of variables, $\Sigma = \{\sigma_1, \sigma_2, \dots\}$ denotes a (finite) set of terminal symbols (i.e., the alphabet), R is a set of production rules (e.g., rules of the form $A_7 \to \sigma_5 A_3 A_9 \sigma_8 A_2$) and $A_{\mathsf{start}} \in V$ denotes a special "start" variable. We say that a string $\alpha \in (\Sigma \cup V)^*$ is *derived* from a variable A_j, denoted by $A_j \stackrel{*}{\Rightarrow} \alpha$, if α can be obtained from A_j by iteratively applying production rules in R. Each such derivation can be described by a *derivation tree*, which is a rooted, directed, ordered, and labeled tree (with edges oriented away from the root), where the root is labeled by A_j, the leaves are labeled by the sybmols of α (in order), and the children of each vertex in the tree correspond to an application of a production rule in G. The language $\mathcal{L} \subseteq \Sigma^*$ *generated* by G consists of all strings that can be derived from A_{start} using the production rules in R.

Let $\mathcal{L}$ be a context-free language and let $G = (V, \Sigma, R, A_{\mathsf{start}})$ be the context-free grammar that generates $\mathcal{L}$. In this section we show how to partition $x \in \mathcal{L}$ into two parts. Next, in Section 4.1.2.3, we show how to extend this technique to multiple parts.

For $x \in \mathcal{L}$ (i.e., $A_{\mathsf{start}} \stackrel{*}{\Rightarrow} x$), there exists a derivation tree T corresponding to the derivation $A_{\mathsf{start}} \stackrel{*}{\Rightarrow} x$. For simplicity, let us assume that T is a *binary* tree. The root of T is labeled by A_{start} and the leaves are labeled, in order, by $x_1, \dots, x_n$, where $n \stackrel{\text{def}}{=} |x|$. Recall that the Lewis-Stearns-Harmanis Lemma [LSH65] states that every binary tree on n leaves has a subtree[13] with a number of leaves between $n/3$ and $2n/3$. Applying this lemma to T, we can find such a subtree T' of T. Observe that T' induces a partition of $[n]$ into two parts $S_1, S_2 \subseteq [n]$, where S_1 (which is actually an interval) contains all the leaves of T that belong to T' and $S_2 \stackrel{\text{def}}{=} [n] \setminus S_1$ contains all other leaves. The IPP prover finds T' and sends S_1 and A_1 to the verifier, where A_1 is the label of the root of T'. Since S_1 is an interval, the latter requires only $O(\log n)$ communication.

Given (S_1, A_1), the verifier can construct the partition and the corresponding languages, where the partition is simply (S_1, S_2) and the languages are

$$\mathcal{L}_1 \stackrel{\text{def}}{=} \left\{ w \in \Sigma^{|S_1|} \,:\, A_1 \stackrel{*}{\Rightarrow} w \right\}$$

and

$$\mathcal{L}_2 \stackrel{\text{def}}{=} \left\{ w \in \Sigma^{|S_2|} \,:\, A_2 \stackrel{*}{\Rightarrow} w[1, \dots, s-1] \circ A_1 \circ w[s, \dots, |S_2|] \right\},$$

where $A_2 \stackrel{\text{def}}{=} A_{\mathsf{start}}$ and $s \in [n]$ is the starting position of the interval S_1 in $[n]$.

[13] Here and throughout this work, by a subtree, we mean a node of the tree together with *all* of its descendants, see also Section 4.2.3.

Note that $\mathcal{L}_2$ is not quite a context-free language (although $\mathcal{L}_1$ is). Rather, $\mathcal{L}_2$ consists of strings that correspond to *partial derivations* (i.e., derivation processes that end before all symbols are terminals) starting from A_{start} that produce strings that have the variable A_1 in their s^{th} coordinate. We refer to such languages, which we view as generalization of context-free languages, as *partial derivation languages*, and for the recursion to go through, we actually design the original protocol to handle not only context-free languages but also partial derivation languages.

Observe that if $x \in \mathcal{L}$, then clearly $x[S_1] \in \mathcal{L}_1$ and $x[S_2] \in \mathcal{L}_2$. On the other hand, suppose that $x[S_1]$ is ε_1-close to a string $z_1 \in \mathcal{L}_1$ and $x[S_2]$ is ε_2-close to a string $z_2 \in \mathcal{L}_2$. If we choose $i \in \{1,2\}$ at random, such that $\Pr[i=1] = |S_1|/n$ and $\Pr[i=2] = |S_2|/n$, then x is $\mathbf{E}_i[\varepsilon_i]$-close to the string $z = z_2[1,\ldots,s-1] \circ z_1 \circ z_2[s,|S_2|]$. Since $A_1 \overset{*}{\Rightarrow} z_1$ and $A_{\mathsf{start}} \overset{*}{\Rightarrow} z_2[1,\ldots,s-1] \circ A_1 \circ z_2[s,\ldots,|S_2|]$ (because $z_1 \in \mathcal{L}_1$ and $z_2 \in \mathcal{L}_2$), we deduce that $A_{\mathsf{start}} \overset{*}{\Rightarrow} z$, and therefore $z \in \mathcal{L}$. Hence, x is $\mathbf{E}_i[\varepsilon_i]$-close to $\mathcal{L}$.

Given the above, we can design an IPP for $\mathcal{L}$ similarly to the IPP for ROBP that was described in Section 4.1.2.1. Specifically, given (S_1, A_1), the verifier chooses at random $i \in \{1,2\}$ according to the distribution above, sends i to the prover, and both parties run the protocol recursively, with respect to the language $\mathcal{L}_i$ and the input $x[S_i]$.

4.1.2.3 Partitioning Context-Free Languages into Multiple Parts

The first step in partitioning context-free languages into *multiple* parts is a generalization of the Lewis-Stearns-Hartmanis lemma that shows that, for every desired parameter $t \in [n]$, every (constant degree) tree T with n leaves has a subtree with roughly t leaves. The precise statement of the lemma and its proof are given in Lemma 4.5 below.

Using Lemma 4.5, we can partition an input $x \in \mathcal{L}$ into k parts of (roughly) the same size in the following way. As before, we construct a derivation tree T corresponding to the derivation $A_{\mathsf{start}} \overset{*}{\Rightarrow} x$. However, this time we use Lemma 4.5 to find a subtree T_1 with roughly n/k leaves. The coordinates of the leaves of T_1 constitute the first part of the partition (denoted by S_1). To find the second subtree, we remove the entire subtree T_1 from T, *except for its root*. We obtain a new tree T' with (roughly) $n - \frac{n}{k}$ leaves, where one of the leaves of T' is labeled by a variable rather than a terminal. By applying Lemma 4.5 again on the new tree T', we can find a subtree T_2 of T' with roughly n/k leaves. The second part (denoted by S_2) of our partition will consist of the coordinates of all the leaves of T_2 that are labeled by terminals (i.e., are also leaves of the original tree T). We stress that S_2 may not be an interval (but rather two intervals separated by S_1).

We proceed similarly, where in each iteration we remove the subtree that was found in the previous iteration (except for its root) and find a new subtree T_i of T with roughly n/k leaves. The subtrees $T_1, T_2, \ldots, T_k$ induce a partition of $[n]$ where the i^{th} part, denoted S_i (of size roughly n/k), consists of all leaves of T_i that are labeled by terminals (i.e., are leaves of the original tree T) but do not belong to $S_1 \cup \cdots \cup S_{i-1}$.

While the representation of a general partition of $[n]$ into k parts requires $n \cdot \log_2(k)$ bits, we show that the partition $S_1, \ldots, S_\ell$ actually has a concise representation. Indeed,

each subtree T_i induces an interval $I_i \subseteq [n]$, which contains all of its leaves (but potentially also coordinates of other parts in the partition). Given $I_1, \ldots, I_\ell$, the partition $S_1, \ldots, S_\ell$ is uniquely determined (by setting $S_i = I_i \backslash (I_1 \cup \cdots \cup I_{i-1})$). We remark that each pair of *intervals* can be either disjoint or nested (i.e., either $I_i \cap I_j = \emptyset$ or $I_i \subsetneq I_j$).

In light of the foregoing discussion, the prover can send to the verifier the intervals $I_1, \ldots, I_k$ and the variables $A_1, \ldots, A_\ell$ of the roots of the subtrees $T_1, \ldots, T_k$ (respectively). Note that the root of the last subtree T_k is in fact the root of the original derivation tree T (and thus $A_k = A_{\mathsf{start}}$) and that its corresponding interval I_k is $[n]$.

Let $I_{i_1}, \ldots, I_{i_k}$ be the ordered (from left to right) maximal intervals of $I_k = [n]$. That is, the (disjoint) intervals that are contained in I_k but are not contained in any of the other intervals. Observe that if the intervals were generated as prescribed, then A_{start} yields a string x' (composed of terminals and variables) that results from x by replacing the substring $x[I_{i_j}]$ with the variable A_{i_j}, for every $j \in [k]$. Denote the language that contains all such strings by $\mathcal{L}_k$. Similarly, for any interval $I_{i_j} \in \{I_{i_1}, \ldots, I_{i_k}\}$, observe that A_{i_j} yields the string that results from $x[I_{i_j}]$ by replacing coordinates in the maximal intervals that I_{i_j} contains with the corresponding variables. Denote the language of all such strings by $\mathcal{L}_{i_j}$. We show that by applying this idea iteratively we obtain languages $\mathcal{L}_1, \ldots, \mathcal{L}_k$ such that (1) if $x \in \mathcal{L}$, then $x[S_i] \in \mathcal{L}_i$ for every $i \in [k]$; and (2) if x is ε-far from $\mathcal{L}$, then $x[S_i]$ is ε-far from $\mathcal{L}_i$, for an average $i \in [k]$, where the average is weighted proportionally to the sizes of $S_1, \ldots, S_k$.

Given the partition above, verifying that $x \in \mathcal{L}$ is reduced to testing that the sub-input $x[S_i]$ is close to $\mathcal{L}_i$, for $i \in [k]$ distributed as above. Hence, as before, the verifier chooses i at random, sends i to the prover and the two parties recursively run an IPP for verifying that $x[S_i]$ is ε-close to $\mathcal{L}_i$.

We emphasize that, as was the case for $k = 2$, the languages $\mathcal{L}_1, \ldots, \mathcal{L}_k$ are not necessarily *context-free languages* but are rather "partial derivation languages". Indeed, for the recursion to go through, we design the IPP to work for such languages (rather than just context-free languages).

4.1.2.4 Digest and Relation to Concatenation Problems

The proofs of Theorems 4.1-4.4 are based on a natural paradigm for designing proofs of proximity. This paradigm consists of two steps: (1) partition the problem into smaller related sub-problems, and (2) verifying a small random sample of the sub-problems. This basic approach was taken by [RVW13] in their construction of an IPP for the Hamming weight problem (i.e., approximating whether a given string has Hamming weight $n/2$). The partitioning in this case is into several intervals of equal length and the IPP prover specifies the Hamming weight of each substring. A more general instantiation of this paradigm was used in [GR13b] to construct MAPs for *parameterized concatenation problems*. Loosely speaking, a language $\mathcal{L}$ is a parameterized concatenation problem if $\mathcal{L} = \mathcal{L}_{\alpha_1} \times \cdots \times \mathcal{L}_{\alpha_k}$, for some integer k, where each language $\mathcal{L}_{\alpha_i}$ is a language parameterized by α_i; thus, the partitioning is done by providing the parameters $\alpha_1, \ldots, \alpha_k$.

In this work we significantly extend the foregoing framework in several aspects: The

partition is not restricted to contiguous intervals, but is rather more involved and depends more dramatically on the structure of the specific language and, moreover, also on the specific input. Furthermore, whereas for concatenation problems the parameterization of each problem is "light" (typically having a logarithmic description length), in our settings the parameterization can be quite extensive, as in *massively parameterized problems* (see survey by Newman [New10]).

4.1.3 Organization

In Section 4.2 we provide the necessarily preliminaries regarding proofs of proximity, context-free languages, and branching programs. In Section 4.3 we construct MAPs and IPPs for languages accepted by ROBPs (with additional discussion on testing affine subspaces in Section 4.3.3). In Section 4.4 we construct MAPs and IPPs for context-free languages (with additional discussion on the Dyck languages in Section 4.4.3). Sections 4.3 and 4.4 can be read independently of each other. We note that the implementation of the outline provided in Section 4.1.2 is far more involved in the case of context-free languages.

4.2 Preliminaries

We begin with some standard notations:

- We denote the concatenation of two strings $x \in \Sigma^n$ and $y \in \Sigma^m$ (over a common alphabet Σ) by $x \circ y \in \Sigma^{n+m}$.

- We denote the *absolute distance* between two (equal length) strings $x \in \Sigma^n$ and $y \in \Sigma^n$ by $\overline{\Delta}(x,y) \stackrel{\text{def}}{=} |\{x_i \neq y_i : i \in [n]\}|$, and their *relative distance* by $\Delta_{\mathsf{REL}}(x,y) \stackrel{\text{def}}{=} \frac{\overline{\Delta}(x,y)}{n}$. If $\Delta_{\mathsf{REL}}(x,y) \leq \varepsilon$, we say that x is ε-close to y, and otherwise we say that x is ε-far from y. Similarly, we denote the *absolute distance* of x from a non-empty set $S \subseteq \Sigma^n$ by $\overline{\Delta}(x,S) \stackrel{\text{def}}{=} \min_{y \in S} \overline{\Delta}(x,y)$ and the *relative distance* of x from S by $\Delta_{\mathsf{REL}}(x,S) \stackrel{\text{def}}{=} \min_{y \in S} \Delta_{\mathsf{REL}}(x,y)$. If $\Delta_{\mathsf{REL}}(x,S) \leq \varepsilon$, we say that x is ε-close to S, and otherwise we say that x is ε-far from S.

- We denote the projection of a string $x \in \Sigma^n$ to a subset of coordinates $S \subseteq [n]$ by $x[S]$. For every $i,j \in [n]$, we denote by $x[i,j]$ the projection of x to the interval $[i,j]$ (if $i > j$ then the interval is empty).

- We denote by $A^x(y)$ the output of algorithm A, given direct access to input y and query (i.e., oracle) access to the string x. Given two interactive machines A and B, we denote by $(A^x, B(y))(z)$ the output of A when interacting with B, where A (resp., B) is given oracle access to x (resp., direct access to y) and both parties have direct access to z.

Integrality. Throughout this work, for simplicity of notation, we use the convention that all (relevant) integer parameters that are stated as real numbers are implicitly rounded to the closest integer.

4.2.1 Property Testing, MAPs and IPPs

In this section we define testers, MAPs and IPPs. Actually, testers and MAPs will be defined as restrictions of IPPs.

4.2.1.1 IPP

We define a language, over an alphabet Σ, as an ensemble $\mathcal{L} \stackrel{\text{def}}{=} \cup_{n\in\mathbb{N}}\mathcal{L}_n$, where $\mathcal{L}_n \subseteq \Sigma^n$ for every $n \in \mathbb{N}$. The definition of an IPP is a natural extension of the standard definition of IP (interactive proof) where the main distinction is that the verifier only has oracle access to the input. Also, since our focus is on the query and communication complexities, we do not restrict the computational complexity of the verifier (see discussion at the end of Section 4.1).

Definition 4.1 (Interactive Proof of Proximity (IPP) [EKR04, RVW13])**.** *An* interactive proof of proximity *(*IPP*) for the language $\mathcal{L} = \cup_{n\in\mathbb{N}}\mathcal{L}_n$ is an interactive protocol with two parties: a (computationally unbounded) prover $\mathcal{P}$, which has* free *access to input x, and a verifier $\mathcal{V}$, which is a probabilistic computationally unbounded algorithm which has oracle access to x. The parties send messages to each other, and at the end of the communication, the following two conditions are satisfied:*

1. **Completeness:** *For every $n \in \mathbb{N}$, proximity parameter $\varepsilon > 0$ and $x \in \mathcal{L}_n$ it holds that*
$$\Pr\left[(\mathcal{V}^x, \mathcal{P}(x))(n, \varepsilon) = 1\right] \geq 2/3.$$
where the probability is over the coin tosses of $\mathcal{V}$.

2. **Soundness:** *For every $n \in \mathbb{N}$, $\varepsilon > 0$, and $x \in \{0,1\}^n$ that is ε-far from $\mathcal{L}_n$ and for every computationally unbounded (cheating) prover $\mathcal{P}^*$ it holds that*
$$\Pr\left[(\mathcal{V}^x, \mathcal{P}^*)(n, \varepsilon) = 0\right] \geq 2/3.$$
where the probability is over the coin tosses of $\mathcal{V}$.

If the completeness condition holds with probability 1, then we say that the IPP *has a* one-sided error *and otherwise the* IPP *is said to have a* two-sided error*. If all of the verifier's messages are uniformly distributed and independent random strings then the* IPP *is said to be* public-coin*.*

An IPP for $\mathcal{L} = \cup_{n\in\mathbb{N}}\mathcal{L}_n$ is said to have query complexity $q : \mathbb{N} \times \mathbb{R}^+ \to \mathbb{N}$ if, for every[14] $n \in \mathbb{N}$, $\varepsilon > 0$ and $x \in \mathcal{L}_n$, the verifier $\mathcal{V}$ makes at most $q(n, \varepsilon)$ queries to x when

[14]We measure the resources used in the protocol only when the verifier interacts with the *honest* prover. However, in the general case, the verifier can simply halt once one of its resources exceeds the corresponding bound (since it knows that it must be interacting with a *cheating* prover).

interacting with $\mathcal{P}$. The IPP is said to have communication complexity $c : \mathbb{N} \times \mathbb{R}^+ \to \mathbb{N}$ if, for every $n \in \mathbb{N}$, $\varepsilon > 0$ and $x \in \mathcal{L}_n$, the communication between $\mathcal{V}$ and $\mathcal{P}$ consists of at most $c(|x|, \varepsilon)$ bits. A round of communication consists of a single message sent from the verifier to the prover followed by a single message sent from the prover to the verifier. The IPP is said to have r rounds (sometimes called an r-round IPP), for $r : \mathbb{N} \times \mathbb{R}^+ \to \mathbb{N}$ if, for every $n \in \mathbb{N}$, $\varepsilon > 0$ and $x \in \mathcal{L}_n$, if the number of rounds in the interaction between $\mathcal{V}$ and $\mathcal{P}$ on input x is at most $r(|x|, \varepsilon)$.

The standard definition of a property tester may be derived from Definition 4.1 by restricting the communication complexity to 0. The definition of an MAP can be derived by restricting the communication to be only from the prover to the verifier (see [GR13b] for further details on MAPs).

Non-uniform IPPs. While Definition 4.1 refers to a *uniform* definition of IPP, throughout this work it will be convenient for us to use a *non-uniform* definition. That is, we fix an integer $n \in \mathbb{N}$, which we think of as a variable parameter, and restrict Definition 4.1 to inputs of length n. Hence, unless stated otherwise, by an IPP we actually mean the *non-uniform* variant. Despite the fact that the integer n is fixed, we view it as a generic parameter and allow ourselves to write asymptotic expressions such as $O(n)$. We also note that while our results are proved in terms of non-uniform IPP, they can be extended to the uniform setting in a straightforward manner.

4.2.1.2 Proximity Oblivious IPP

Extending the notion of proximity oblivious testers [GR11], we define a *proximity oblivious* IPP, as a variant of an IPP in which neither party receives the proximity parameter as input and for every $\varepsilon > 0$, the verifier is required to reject inputs that are ε-far from the language with some probability $\rho(\varepsilon)$.

Definition 4.2 (Proximity Oblivious IPP)**.** *Let $\rho : (0, 1] \to (0, 1]$ be a monotone function. A* proximity oblivious IPP with detection probability ρ*, for the language $\mathcal{L} = \cup_{n \in \mathbb{N}} \mathcal{L}_n$ is similar to Definition 4.1, except that the neither the verifier nor the prover*[15] *receive the proximity parameter as input, and the completeness and soundness conditions are modified as follows:*

1. **Completeness:** *For every $n \in \mathbb{N}$, and $x \in \mathcal{L}_n$ it holds that*[16]

$$\Pr\left[(\mathcal{V}^x, \mathcal{P}(x))(n) = 1\right] = 1.$$

[15] Since we do not bound the computational resources of the prover, it can simply deduce the proximity parameter from the input.

[16] Note that we require the verifier to accept inputs $x \in \mathcal{L}$ with probability 1. A more general definition could allow this probability to be some smaller constant or even a function of ε (see [GS12]). For simplicity (and since it suffices for our purposes), in this work we only consider proximity oblivious IPPs with perfect completeness.

2. **Soundness:** *For every $n \in \mathbb{N}$, every $x \in \Sigma^n$, and for every computationally unbounded (cheating) prover $\mathcal{P}^*$ it holds that*

$$\Pr\left[(\mathcal{V}^x, \mathcal{P}^*)(n) = 0\right] \geq \rho\big(\Delta(x, \mathcal{L}_n)\big)$$

In both condiditons the probability is over the coin tosses of $\mathcal{V}$.

Note that any proximity oblivious IPP with detection probability $\rho(\cdot)$, can be transformed into a standard IPP (as in Definition 4.1) by repeating the proximity oblivious IPP $O(1/\rho(\varepsilon))$ times in parallel (see Section 4.A for details on parallel repetition for IPPs).

4.2.2 Read-Once Branching Programs (ROBPs)

In this section we provide the necessary background on ROBPs (needed only for Section 4.3). An ROBP is defined as follows

Definition 4.3 (ROBP)**.** *A* branching program *on n variables is a directed acyclic graph that has a unique source vertex with in-degree 0 and (possibly) multiple sink vertices with out-degree 0. Each sink vertex is labeled either with 0 (i.e.,* reject*) or 1 (i.e.,* accept*). Each non-sink vertex is labeled by an index $i \in [n]$ and has exactly 2 outgoing edges, which are labeled by 0 and 1.*

The output of the branching program B on input $x \in \{0,1\}^n$, denoted $B(x)$, is the label of the sink vertex reached by taking a walk, starting at the source vertex such that at every vertex labeled by $i \in [n]$, the step taken is on the edge labeled by x_i.

The branching program is said to be read-once *(or ROBP for short) if along every path from source to sink, every index $i \in [n]$ appears at most once. The* size *of a branching program B, denoted $|B|$, is the number of vertices in its graph.*

Let φ and ψ be vertices in the branching program B on n variables and let $x \in \{0,1\}^n$. Loosely speaking, we write $\varphi \overset{x,k}{\rightsquigarrow} \psi$ if the walk of length k corresponding to x that starts at φ ends at ψ. Note that only k coordinates of x are read (adaptively) in this walk and that φ itself only determines the first variable read. Formally, we write $\varphi \overset{x,1}{\rightsquigarrow} \psi$ if the edge (φ, ψ) appears in B and is labeled by x_i, where i is the label of φ, and we (inductively) write $\varphi \overset{x,k}{\rightsquigarrow} \psi$ if there exists a vertex ζ in B such that $\varphi \overset{x,1}{\rightsquigarrow} \zeta$ and $\zeta \overset{x,k-1}{\rightsquigarrow} \psi$.

4.2.3 Context-Free Languages

In this section we provide the necessary background on context-free languages (needed only for Section 4.4). To define *context-free languages*, we first define context-free grammars (see [HMU06] for more details).

Definition 4.4 (Context-free grammar)**.** *A* context-free grammar *is a tuple $G = (V, \Sigma, R, A_{\mathsf{start}})$ such that V is a (finite) set of symbols, referred to as* variables*; Σ is a (finite) set of symbols, referred to as* terminals*; $R \subseteq V \times (V \cup \Sigma)^*$ is a (finite) relation, where each $(A, \alpha) \in R$ is referred to as a* production rule *and is denoted by $A \to \alpha$; $A_{\mathsf{start}} \in V$ is a variable that is referred to as the* start variable*.*

Let $G = (V, \Sigma, R, A_{\mathsf{start}})$ be a context-free grammar, and let $\alpha, \beta \in (V \cup \Sigma)^*$ be strings of terminals and variables. We say that α directly yields β, denoted by $\alpha \Rightarrow \beta$, if there exists a production rule $A \to \gamma$ in R such that β is obtained from α by replacing exactly one occurrence of the variable A in α with the string $\gamma \in (V \cup \Sigma)^*$. We say that α yields β, denoted $\alpha \overset{*}{\Rightarrow} \beta$ if there exists a finite sequence of strings $\alpha_0, \ldots, \alpha_k \in (V \cup \Sigma)^*$ such that $\alpha_0 = \alpha$, $\alpha_k = \beta$, and $\alpha_0 \Rightarrow \ldots \Rightarrow \alpha_k$. The language $\mathcal{L}_n \subseteq \Sigma^n$ is a context-free language if there exists a grammar $G = (V, \Sigma, R, A_{\mathsf{start}})$ such that $\mathcal{L}_n = \{x \in \Sigma^n : A_{\mathsf{start}} \overset{*}{\Rightarrow} x\}$, where the derivation is with respect to the rules in R.

Derivation Tree. Let $G = (V, \Sigma, R, A_{\mathsf{start}})$ be a context-free grammar. For $A \in V$ and $x \in \Sigma^*$, a derivation tree, corresponding to the derivation $A \overset{*}{\Rightarrow} x$, is a rooted, directed, ordered, and labeled tree T (with edges oriented away from the root) that satisfies the following properties:

- Each internal vertex is labeled by some variable, and the root is labeled by the variable A.
- Each leaf is labeled by a terminal symbol, where the i^{th} leaf is labeled by the i^{th} symbol of x.
- For every internal vertex v, if v is labeled by the variable $A' \in V$ and for every $i \in [k]$ (where k is the number of children of v) the i^{th} child of v is labeled by $\alpha_i \in V \cup \Sigma$, then the production rule $A' \to \alpha_0 \circ \cdots \circ \alpha_k$ must belong to R.

For every derivation $A \overset{*}{\Rightarrow} x$ there exists a corresponding derivation tree.

Trees and the Lewis-Stearns-Hartmanis Lemma. In this work we only consider trees that are rooted, directed, and ordered (e.g., derivation trees as above). Thus, throughout this work, whenever we say tree we mean a rooted, directed, and ordered tree (with edges oriented away from the root). Note that the fact that the tree is ordered induces an ordering of its leaves. We define the arity of a tree to be the maximal number of children of any vertex in the tree. We follow the data-structure literature and define a subtree of a tree T as a tree consisting of a node in T and all of its descendants in T.[17]

For a tree T, we denote by $L(T)$ the number of leaves of T. We will use the following straightforward generalization of the Lewis-Stearns-Hartmanis Lemma [LSH65]:

Lemma 4.5. *Let T be a tree with arity d and let $t \in [L(T)]$. Then, there exists a subtree T' of T with $L(T') \in [t/d, t]$ leaves.*

The Lewis-Stearns-Hartmanis lemma corresponds to the special case of Lemma 4.5 in which $d = 2$ (i.e., a binary tree) and $t = 2n/3$.

[17] This definition differs from the graph-theoretic definition that defines a subtree as any connected subgraph of a tree. For example, the root of a tree is a subtree in the graph theoretic sense but not according to our definition (unless the tree has exactly one vertex).

Proof of Lemma 4.5. We prove by induction on the *size* of the tree (not on the number of leaves), noting that the base case holds trivially. Suppose that the lemma holds for all trees of size less than n. Let T be a tree of size n and let $t \in [L(T)]$.

If $t = L(T)$, then we are done (since $L(T) \in [t/d, t]$ and so T itself has the desired property). Otherwise, if $t < L(T)$, then T has a subtree T' (rooted at one of its children) such that $L(T') \geq t/d$ (since otherwise T has a total of less than $d \cdot t/d = t < L(T)$ leaves). If $L(T') \leq t$, then we are also done (since $L(T') \in [t/d, t]$ and so T' satisfies the desired property). Otherwise, $t \in [L(T')]$ and the lemma follows by applying the induction hypothesis on T'. □

4.3 MAPs and IPPs for Read-Once Branching Programs

In this section we prove Theorem 4.4 (and Theorem 4.2 will follow as a special case of the proof) by constructing an IPP for every language that is accepted by an ROBP.

4.3.1 IPPs for ROBPs

Before presenting the IPP formally, we provide a short overview of the protocol (for a more detailed overview, see Section 4.1.2.1). Let B be an ROBP on n variables. We construct an r-round public-coin IPP for the language that is accepted by B. The IPP runs recursively, where each round of communication is as follows. Given an input x that is accepted by B within $n' \leq n$ steps, the prover finds the accepting path $\varphi_0 \to \varphi_1 \to \cdots \to \varphi_{n'}$ and sends to the verifier the subsequence $\varphi_{n'/k}, \ldots, \varphi_{i \cdot n'/k}, \ldots, \varphi_{n'}$ that contains every $(n'/k)^{\text{th}}$ vertex along this path. The verifier checks that every two consecutive vertices in the prover's message are connected (this can be done *without* making queries to the input x), then selects uniformly at random $i \in [k]$, sends i to the prover, and defines a new ROBP B_i that has the same graph as B, but its source vertex is $\varphi_{(i-1) \cdot n'/k}$ and its *unique* accepting sink is $\varphi_{i \cdot n'/k}$.[18] Subsequently, both parties (recursively) invoke the IPP protocol on input x and the ROBP B_i.

A crucial point is that although the input x does not not shrink in each recursive call, the *effective length of the input*, n', which is the alleged number of bits of x that need to be read in order to get from the source to the accepting sink, does shrink (by a factor of k). Hence, after r such rounds, the verifier can read all bits of x that are required to verify the current statement (which refers to a path of length n/k^r). Interestingly, and in contrast to the simpler case of OBDDs, in this last step the verifier reads the n/k^r bits of the input *adaptively*, based on the steps taken by the ROBP.

[18] One could alternatively define B_i to consist only of vertices at distance at most n'/k from $\varphi_{(i-1) \cdot n'/k}$. We refrain from taking this approach due to technical reasons. Note that also when using this alternate definition, when considering general ROBPs (rather than OBDDs), B_i could potentially look at all of the n bits of the input (see discussion at the end of Section 4.1.2.1).

In the IPP that we construct, we assume for simplicity that the verifier is given an integer $n' \leq n$, and the claim (which the verifier is trying to validate) is that $B(x) = 1$ after reading *exactly* n' bits of the input x. Furthermore, we assume that the ROBP B is such that there exists *some* accepting path (i.e., from the source to some accepting sink) of length n'. We can reduce the general case to this restricted setting by having the prover send n' as part of its first message and having the verifier explicitly check that there is some accepting path of length n' (this check requires no queries to the main input x).

Recall that, as noted in Section 4.1.2, to facilitate the recursion we use the notion of a *proximity oblivious* IPP (see Section 4.2.1.2). When handling general ROBPs (rather than OBDDs), we follow this approach in *spirit* but, due to technical reasons, the construction will not exactly fit Definition 4.2. More specifically, since in each step of the recursion only the *effective* length of the input shrinks (but the actual length of the input stays the same), throughout the proof it will be more convenient for us to use *absolute* distances, denoted by $\overline{\Delta}$ (see Section 4.2) rather than with distances that are relative to n. Hence, the detection probability $\overline{\rho}$ of the verifier will be a function of the *absolute distance* (rather than of the relative distance) of the input from the language. That is, we will construct an absolute proximity oblivious IPP with detection probability $\left\{\overline{\rho}_n : \{0,\ldots,n\} \to (0,1]\right\}_{n\in\mathbb{N}}$, which is the same as Definition 4.2 except that we modify the soundness condition as follows:

- ***Soundness:*** For every $n \in \mathbb{N}$, $x \in \Sigma^n$, and for every computationally unbounded (cheating) prover $\mathcal{P}^*$ it holds that

$$\Pr\left[(\mathcal{V}^x, \mathcal{P}^*)(n) = 0\right] \geq \overline{\rho}_n\big(\overline{\Delta}\left(x, \mathcal{L}_n\right)\big) \tag{4.1}$$

Again, we can transform an *absolute* proximity oblivious IPP with detection probability $\left\{\overline{\rho}_n : \{0,\ldots,n\} \to (0,1]\right\}_{n\in\mathbb{N}}$ into a standard IPP (as in Definition 4.1) by repeating the base protocol $O(1/\overline{\rho}_n(\varepsilon \cdot n))$ times in parallel.

The absolute proximity oblivious IPP for ROBPs, denoted ROBP-IPP, is presented in Fig. 4.1 (recall that the notation $\varphi \overset{x,m}{\rightsquigarrow} \psi$, which is used in Fig. 4.1 means that, given input x, the ROBP walks from φ to ψ in m steps, see Section 4.2.2 for details).

It can be easily verified that the round complexity is r, the communication complexity is $O(rk \cdot \log(|B|))$ and the query complexity is $O(n/k^r)$. We proceed to show that completeness and soundness hold.

Completeness. Let B be a ROBP on n variables, let $r \geq 0$, $n' \in [n]$, and let $x \in \mathcal{B}^n$ such that $B(x) = 1$ after reading exactly n' bits of the input. (Perfect) completeness follows by induction on r as follows.

For $r = 0$, the verifier just reads the appropriate n' bits of the input and accepts with probability 1. For $r \geq 1$, let $(\varphi_0, \varphi_1, \ldots, \varphi_{n'})$ be the accepting path corresponding to x. The checks that the verifier performs in the current round pass, since $\varphi_{n'}$ is indeed an accepting sink and $\varphi_{(i-1)\cdot n'/k^r} \overset{x,n'/k^r}{\rightsquigarrow} \varphi_{i\cdot n'/k^r}$, for every $i \in [p^r]$. Furthermore, since for

The Protocol $\mathsf{ROBP\text{-}IPP}^{B}_{n,n',r}$**:**

Common Input: Integers $n, n' \in \mathbb{N}$, a ROBP B on n variables such that there exists some accepting path of length n' in B, and a parameter $r \in \mathbb{N}$.

Prover's Input: Direct access to $x \in \{0,1\}^n$ such that $B(x) = 1$ in exactly n' steps.

Verifier's Input: Oracle access to the same x.

1. If $r = 0$, the verifier $\mathcal{V}$ checks whether $B(x) = 1$ after exactly n' steps by (adaptively) reading the appropriate n' bits of x. If $B(x) = 1$, then $\mathcal{V}$ accepts, otherwise it rejects, and in either case both parties terminate the protocol.
2. The Prover $\mathcal{P}$:
   (a) Let $\varphi = (\varphi_0, \ldots, \varphi_{n'}) \in [|B|]^{n'}$ be the sequence of vertices in the accepting path (of length $n' \leq n$) in B that corresponds to the evaluation of B on input x.
   (b) Send $(\varphi_{n'/k}, \varphi_{2n'/k}, \ldots, \varphi_{n'})$ to $\mathcal{V}$.[a]
3. The Verifier $\mathcal{V}$:
   (a) Check that $\varphi_{n'}$ is an accepting sink of B.
   (b) Let φ_0 be the source of B.
   (c) For every $i \in [k]$, check that there exists some input $x^{(i)} \in \{0,1\}^n$ such that $\varphi_{(i-1)\cdot n'/k} \overset{x^{(i)}, n'/k}{\rightsquigarrow} \varphi_{i\cdot n'/k}$.[b]
   (d) Select uniformly at random an index i in $[k]$, and send i to $\mathcal{P}$.
4. Denote by B_i the ROBP that has the same graph as B, except that its source is $\varphi_{(i-1)\cdot n'/k}$, and its *unique* accepting vertex is $\varphi_{i\cdot n'/k}$.
5. Both parties (recursively) invoke $\mathsf{ROBP\text{-}IPP}^{B_i}_{n,n'',r-1}$, where $n'' = n'/k$, on input x.

[a] The accepting sink $\varphi_{n'}$ is also sent since (in the first step of the recursion) there could be multiple accepting sinks.

[b] This check is performed without making any queries to the main input x. Although our focus is not on *computational* complexity, we note that this can be done in $\mathsf{poly}(s)$ time.

Figure 4.1: IPP for ROBPs

every choice of $i \in [k]$ (made by the verifier) it holds that $\varphi_{(i-1)\cdot n'/k} \overset{x,n'/k}{\rightsquigarrow} \varphi_{i\cdot n'/k}$, the two parties recursively run the $r-1$ round protocol on a branching program B_i such that $B_i(x) = 1$ after reading exactly n'/k bits of x. Hence, by the inductive hypothesis the verifier accepts with probability 1.

Soundness. Soundness follows directly from the following lemma, which is proved by induction on the number of rounds r. We suggest to the reader to first consider the case that $n' = n$ in both the lemma statement and its proof. Nevertheless, we stress that in lower levels of the recursion, the parameter n' becomes much smaller than n.

Lemma 4.6. *Let $n \in \mathbb{N}$, $n' \in [n]$ and $r \geq 0$. For every ROBP B of size s on n variables that has an accepting path of length n', every x that is in* absolute *distance $\varepsilon \cdot n'$ from $\{z \in \{0,1\}^n : B(z) = 1\}$, and for every cheating prover strategy $\mathcal{P}^*$ it holds that:*

$$\Pr[(\mathcal{V}^x, \mathcal{P}^*)(n, n', B, r) = 0] \geq \varepsilon,$$

where $\mathcal{V}$ is the r-round verifier of ROBP-IPP *(of Fig. 4.1).*

Proof. We prove Lemma 4.6 by induction on the number of rounds $r \geq 0$. In the base case, corresponding to $r = 0$, the verifier simply ignores the prover and reads the appropriate n' bits of x. Hence, if $B(x) \neq 1$, then the verifier rejects with probability 1.[19]

In the inductive step, for $r \geq 1$, let $x \in \{0,1\}^n$ that is at absolute distance $\varepsilon \cdot n'$ from $\{z \in \{0,1\}^n : B(z) = 1\}$ and let P^* be a (deterministic) cheating prover strategy for the protocol ROBP-IPP of Fig. 4.1 (with r rounds). Let $(\varphi_{n'/k}, \varphi_{2n'/k}, \ldots, \varphi_{n'})$ be the first message sent by P^* to $\mathcal{V}$ and let φ_0 be the source. Since the verifier explicitly checks these conditions, it must be the case that $\varphi_{n'}$ is an accepting sink, and that for every $i \in [k]$, there exists some $x^{(i)} \in \{0,1\}^n$ such that $\varphi_{(i-1)\cdot n'/k} \overset{x^{(i)},n'/k}{\rightsquigarrow} \varphi_{i\cdot n'/k}$. Furthermore, for every $i \in [k]$, we assume without loss of generality that $x^{(i)}$ is the string z that minimizes the distance of x to the set $\left\{z \in \{0,1\}^n : \varphi_{(i-1)\cdot n'/k} \overset{z,n'/k}{\rightsquigarrow} \varphi_{i\cdot n'/k}\right\}$.

Recall that $\overline{\Delta}$ denotes absolute distance (see Section 4.2) and let $\varepsilon_i \overset{\text{def}}{=} \frac{\overline{\Delta}(x,x^{(i)})}{n'/k}$. The following claim, which crucially uses the fact that B is *read-once*, shows that the average of the ε_i's (which will later be shown to lower bound the rejection probability of $\mathcal{V}$) is at least ε.

Claim 4.6.1. $\mathbf{E}_i[\varepsilon_i] \geq \varepsilon$.

Proof. For every $i \in [k]$, let $J_i \subseteq [n]$ be the set of n'/k variables that are read when going from $\varphi_{(i-1)\cdot n'/k}$ to $\varphi_{i\cdot n'/k}$ on input $x^{(i)}$. We first prove that the sets J_i's are disjoint. Assume otherwise; that is, that there exists $j \in J_{i_1} \cap J_{i_2}$ for some $i_1, i_2 \in [k]$. For every $i \in [k]$, denote the path from $\varphi_{(i-1)\cdot n'/k}$ to $\varphi_{i\cdot n'/k}$ (in B) on input $x^{(i)}$ by P_i. Consider

[19] In the trivial case that $\varepsilon = 0$ (i.e., $B(x) = 1$), the verifier also satisfies the requirement, since it rejects with probability at least $0 = \varepsilon$. It may also be worth mentioning that it always holds that $\varepsilon \leq 1$, since B has an accepting path of length n'.

the concatenated path $P_1 \circ \cdots \circ P_k$. This is a path in B in which the label j appears twice (both in P_{i_1} and in P_{i_2}) in contradiction to our assumption that B is a *read-once* branching program.

Define $x' \in \{0,1\}^n$ as follows. For every $j \in [n]$, if $j \in J_i$ for some $i \in [k]$ (which must be unique as just shown), then $x'[j] \stackrel{\text{def}}{=} x^{(i)}[j]$, and otherwise (i.e., if $j \notin J_1 \cup \cdots \cup J_p$) we set $x'[j] \stackrel{\text{def}}{=} x[j]$. Note that

$$\varphi_0 \overset{x',n'/k}{\rightsquigarrow} \varphi_{n'/k} \overset{x',n'/k}{\rightsquigarrow} \cdots \overset{x',n'/k}{\rightsquigarrow} \varphi_{n'}$$

and therefore $B(x') = 1$. The claim follows by noting that

$$\varepsilon \cdot n' \leq \overline{\Delta}(x, x') = \sum_{i \in [k]} \overline{\Delta}(x[J_i], x'[J_i]) = \sum_{i \in [k]} \overline{\Delta}\left(x[J_i], x^{(i)}[J_i]\right) \leq \sum_{i \in [k]} \overline{\Delta}\left(x, x^{(i)}\right) = \sum_{i \in [k]} \varepsilon_i \cdot n'/k,$$

where the first inequality follows from the fact that x is in absolute distance $\varepsilon \cdot n'$ from $\{z \in \{0,1\}^n : B(z) = 1\}$ combined with the fact that $B(x') = 1$, and the first and second equality follow from the definition of x' (the first equality also uses the fact that the J_i's are disjoint). The claim follows. □

For every $i \in [k]$, let B_i be the ROBP that has the same graph as B, but its source is $\varphi_{(i-1)\cdot n'/k}$, and its *unique* accepting vertex is $\varphi_{i \cdot n'/k}$. Let P_i^* be the residual $r-1$ round strategy of P^* after receiving the message i from $\mathcal{V}$ in the first round, and let $\mathcal{V}_i$ be the residual strategy of $\mathcal{V}$ after fixing its first message to i. Note that $\mathcal{V}_i$ is exactly the strategy of the verifier in $\mathsf{ROBP\text{-}IPP}^{B_i}_{n,n',r-1}$.

Claim 4.6.2. *For every $i \in [k]$, it holds that*

$$\Pr[(\mathcal{V}_i^x, P_i^*)(n, n'', B_i, r-1) = 0] \geq \varepsilon_i,$$

where $n'' = n'/k$.

Proof. Let $i \in [k]$. Recall that $x^{(i)}$ was chosen as $z \in \{0,1\}^n$ that minimizes the distance of x to the set $S_i \stackrel{\text{def}}{=} \left\{z \in \{0,1\}^n : B_i(z) = 1 \text{ using exactly } n'/k \text{ steps}\right\}$. Hence,

$$\overline{\Delta}(x, S_i) = \overline{\Delta}\left(x, x^{(i)}\right) = \varepsilon_i \cdot n'/k.$$

Hence, $(\mathcal{V}_i^x, P_i^*)(n, n'', B_i, r-1)$ corresponds to an invocation of the $r-1$ round version of the protocol on an input x that is in absolute distance $\varepsilon_i \cdot n'/k$ from S_i. Therefore, by the inductive hypothesis, the verifier $\mathcal{V}_i$ rejects with probability at least ε_i. □

Using Claim 4.6.1 and Claim 4.6.2 we obtain that

$$\Pr[(\mathcal{V}^x, P^*)(n, n', B, r) = 0] = \mathop{\mathbf{E}}_{i \in [k]}\left[\Pr[(\mathcal{V}_i^x, P_i^*)(n, n'/k, B_i, r-1) = 0]\right] \geq \mathop{\mathbf{E}}_{i \in [k]}[\varepsilon_i] \geq \varepsilon, \tag{4.2}$$

and the lemma follows. □

This concludes the proof of Theorem 4.4.

Remark 4.7 (Computational Complexity)**.** *The running time of the* IPP *prover in Fig. 4.1 is polynomial in its input (i.e.,* $\mathsf{poly}(|B|, n, k, r, 1/\varepsilon)$*). As for the* IPP *verifier, if the representation of the ROBP* B *allows one to check if two vertices in the graph of* B *are connected in* $\mathsf{polylog}(|B|)$ *time, then the verifier runs in time* $\mathsf{poly}(\log n, k, r, \log(|B|), 1/\varepsilon)$*. If such a representation is not available, then it can be generated in a relatively expensive (i.e.,* $\mathsf{poly}(|B|)$ *time) pre-processing step, which does not depend on the input* x *and can be re-used for multiple inputs.*

Alternatively, for some other natural representations, the verifier can employ the prover to efficiently check if two vertices in B *are connected. Consider for example a natural representation in which there exists a polynomial (i.e.,* $\mathsf{poly}(\log(|B|))$*) size circuit* C *that on input a vertex* v *(in the graph of* B*) and a bit* $\sigma \in \{0,1\}$ *outputs the neighbor* u *of* v *that* σ *leads to (i.e., the edge* (v,u) *is labeled by* σ*). Suppose further that* C *is* $O(\log(|B|))$*-space uniform (i.e., can be generated by an* $O(\log(|B|))$*-space Turing machine). In such case we can use the prover to check connectivity, as described next, and so we obtain sub-linear verification.*

In order to verify connectivity efficiently, we first observe that there exists an ($O(\log(|B|))$*-space uniform) circuit, of* $\mathsf{polylog}(|B|)$*-depth and* $\mathsf{poly}(|B|)$*-size, that on input two vertices* v *and* u *outputs* 1 *if and only if they are connected (possibly via a long path).*[20] *Now we can apply the efficient interactive proof-system for low-depth computation*[21] *of Goldwasser et al. [GKR08, Theorem 1] to obtain an interactive proof-system that verifies that two given vertices are connected, where the verifier runs in time* $\mathsf{polylog}(|B|)$ *and the prover runs in time* $\mathsf{poly}(|B|)$*.*[22] *We note that employing this proof-system inside our* IPP *increases the round complexity of the* IPP *by a* $\mathsf{polylog}(|B|)$ *factor.*

Remark 4.8 (IPPs for Ordered Binary Decision Diagrams)**.** *Recall that an ordered binary decision diagram (OBDD) is an ROBP that is both* layered *and* ordered *(see Section 4.1.2.1). We observe that the communication complexity in Theorem 4.4 can be slightly improved for OBDDs of width* w *and size* $s = O(nw)$ *from* $O\big((pr \log s) \cdot \varepsilon^{-1}\big)$ *to* $O\big((pr \log w) \cdot \varepsilon^{-1}\big)$*, by noting that the* i^{th} *vertex specified by the prover (say, in the first round)* must *be in layer* $i \cdot n/p$ *and therefore it can be specified using only* $\log_2 w$ *bits.*

[20] The circuit first uses C to generate the entire adjacency matrix of B and then checks whether v and u are connected by repeated squaring of the adjacency matrix. Note that all actions can be implemented in $\mathsf{polylog}(|B|)$-depth and $\mathsf{poly}(|B|)$-size.

[21] Goldwasser *et al.* show that any language that is accepted by a ($O(\log(S(n)))$-space uniform) circuit of depth $D(n)$ and size $S(n)$, has an interactive proof-system, where the verifier runs in time $(n + D(n)) \cdot \mathsf{polylog}(S(n))$ and the prover runs in time $\mathsf{poly}(S(n))$.

[22] We stress that the interactive proof-system for verifying connectivity is a standard interactive proof-system and not a "proof of proximity" (i.e., not an IPP). Indeed, this is crucial for our application since we use the interactive proof-system for connectivity as a subroutine within our IPP, and the IPP verifier should reject if at any point it encounters a pair of vertices that are disconnected (even if the pair is "close" to being connected).

4.3.2 MAPs for ROBPs

We observe that Theorem 4.2 follows almost directly from the proof of Theorem 4.4, when restricted to the case $r = 1$. Indeed, the only two gaps (which are easily resolved) are:

1. Interaction: Theorem 4.4 (restricted to $r = 1$) guarantees a 1-round IPP for languages recoginzed by ROBPs. In general, a 1-round IPP is not necessarily an MAP, since it may include a message sent from the verifier to the prover. Nevertheless, the order of the messages in our protocol is such that first the prover sends a message to the verifier and then the verifier responds. The last message can clearly be avoided and so we obtain an MAP.

2. Dependence on the Proximity Parameter in the Proof Length: Recall that there is a linear dependence on $1/\varepsilon$ in the communication complexity in Theorem 4.4, due to the $O(1/\varepsilon)$ parallel repetitions that were used. However, for MAPs, parallel repetition can be performed without increasing the proof length, since the proof is a deterministic function of the input. Hence, we can save the additional $O(1/\varepsilon)$ factor that is used for general IPPs.

4.3.3 MAPs and IPPs for Affine Spaces

In this section, as an example, we show how Theorems 4.2 and 4.4 can yield MAPs and IPPs for any *affine space*.

Before proceeding to the proof, we remark that Rothblum *et al.* [RVW13] identified a specific affine space, called PVAL, as being "complete" for the construction of IPPs for the class NC.[23] They constructed an IPP for PVAL and thereby obtain IPPs for all of NC. Interestingly, PVAL is an affine space and so the results of this section yield an alternative IPP for it. Unfortunately though, the parameters obtained by our IPP are inferior[24] to those of [RVW13] and do not yield an alternative IPP for NC.

Definition 4.9. *Let $\mathbb{F}$ be a finite field, $n \in \mathbb{N}$ and $t \in [n]$. An* affine subspace *of the vector space $\mathbb{F}^n$, denoted $\mathsf{AffineSpace}_{A,b}$, is parametrized by a matrix $A \in \mathbb{F}^{t \times n}$ and a vector $b \in \mathbb{F}^t$ and consists of all strings $x \in \mathbb{F}^n$ such that $Ax = b$.*

Our construction of an IPP for every affine space follows directly from Theorem 4.4 by showing that membership in an affine subspaces can be recognized by a small-width OBDD.

Proposition 4.10. *Let $\mathbb{F}$ be a finite field, $n \in \mathbb{N}$ and $t \in [n]$. For every $A \in \mathbb{F}^{t \times n}$ and $b \in \mathbb{F}^t$, there exists a width $|\mathbb{F}|^{O(t)}$ OBDD that accepts $\mathsf{AffineSpace}_{A,b}$.*

[23] The language PVAL is parameterized by a sequence of points in a finite vector space and a sequence of values, and consists of all strings x whose low degree extension $\mathsf{LDE}(x)$ is equal to the given sequence of values at the corresponding sequence of points

[24] More specifically, for a PVAL instance parameterized by t points, the communication complexity in our protocol is $O(t \cdot 1/\varepsilon \cdot \mathsf{polylog}(n))$, whereas in [RVW13] it is $O(t \cdot (1/\varepsilon)^{o(1)} \cdot \mathsf{polylog}(n))$. Our result is insufficient since in the context of the proof of IPPs for NC, $t = \sqrt{n}$ and $\varepsilon = 1/\sqrt{n}$.

Proof. We describe a deterministic streaming algorithm for deciding membership in $\mathsf{AffineSpace}_{A,b}$. The algorithm gets access to a stream of n fields elements, reads the input element-by-element (in order) and stores a total of t field elements at any given time. Transforming the latter into an OBDD, as required, is straightforward.[25]

Denote the columns of A by $a_1, \ldots, a_n \in \mathbb{F}^t$. The algorithm maintains a vector $c \in \mathbb{F}^t$ which is initialized to 0. The streaming algorithm reads the input $x \in \mathbb{F}^n$ element-by-element and after reading the i^{th} element, the algorithm sets $c \leftarrow c + x_i a_i$ (where the addition is over $\mathbb{F}$). In the end, it holds that

$$c = \sum_{i=1}^{n} x_i a_i = Ax$$

and therefore it suffices for the algorithm to accept if $c = b$ and reject otherwise.

□

By applying Theorem 4.4, we obtain the following corollary.

Corollary 4.5. *Let $\mathbb{F}$ be a finite field, $n \in \mathbb{N}$ and $t \in [n]$. For every $A \in \mathbb{F}^{t \times n}$, $b \in \mathbb{F}^t$ and for every $k = k(n) \geq 2$ and $r = r(n) \geq 1$ such that $k^r \leq n$, there exists an r-round IPP for $\mathsf{AffineSpace}_{A,b}$ with communication complexity $O\big((rk \cdot t \log |\mathbb{F}|) \cdot \varepsilon^{-1}\big)$ and query complexity $O\left(\frac{n}{k^r} \cdot \varepsilon^{-1}\right)$. Furthermore, the IPP is public-coin and has one-sided error.*

4.4 MAPs and IPPs for Context-Free Languages

In this section we prove Theorem 4.3 by constructing an IPP for any context-free language. As noted in the introduction, the proof of Theorem 4.1 will follow as a special case of this IPP.

The proof of Theorem 4.3 extensively uses the notions of a *partial derivation* and a *partial derivation language*. Recall that a partial derivation of a grammar G is a derivation, according to the production rules of G, in which not all variables are expanded. Our notion of a *partial derivation language* is more complex. In particular, it does *not* refer to the language that consist of all possible partial derivations of the grammar (i.e., $\{x \in (\Sigma \cup V)^* : A_{\mathsf{start}} \overset{*}{\Rightarrow} x\}$). Rather, we define a partial derivation language as a language that consists of the subsequence of *terminal symbols* that correspond to partial derivations that start at some fixed variable. Furthermore, we consider only partial derivations in which the subsequence of variables in the partial derivation occur in specific locations. More concretely, a partial derivation language is parameterized by (1) a start variable A_0; (2) the number of terminals m; (3) a sequence of ℓ locations $i_1, \ldots, i_\ell$; and (4) a corresponding sequence of variables $A_1, \ldots, A_\ell$. The language consists of strings z of length m such that the string $z' = z[1, i_1 - 1] \circ A_1 \circ z[i_1, i_2 - 1] \circ \cdots \circ A_\ell \circ z[i_\ell, m]$ can be derived from A_0. More formally,

[25] Loosely speaking, each layer of the OBDD will consist of the $2^{O(t \log |\mathbb{F}|)}$ possible configurations of the streaming algorithm (which include both its current state and possibly some of the bits of the element that is currently being read).

Definition 4.11 (Partial Derivation Language)**.** *A* partial derivation language *of the grammar $G = (V, \Sigma, R, A_{\mathsf{start}})$ is a language $\mathcal{L} \subseteq \Sigma^m$, parameterized by indices $1 \leq i_1 \leq \ldots \leq i_\ell \leq m$ and variables $A_0, \ldots, A_\ell \in V$ such that*

$$\mathcal{L} \stackrel{\text{def}}{=} \left\{ z \in \Sigma^m \;:\; A_0 \stackrel{*}{\Rightarrow} z[1, i_1 - 1] \circ A_1 \circ z[i_1, i_2 - 1] \circ \cdots \circ A_\ell \circ z[i_\ell, m] \right\}.$$

The concise description of a partial derivation language $\mathcal{L} \subseteq \Sigma^m$, parameterized by $\overline{i} = (i_1, \ldots, i_\ell)$ and $\overline{A} = (A_0, \ldots, A_\ell)$, is denoted by $\langle \mathcal{L} \rangle \stackrel{\text{def}}{=} \left(m, \overline{i}, \overline{A}\right)$.

We stress that $z \in \mathcal{L}$, where $\mathcal{L}$ is a partial derivation language such that $\langle \mathcal{L} \rangle = \big(m, (i_1, \ldots, i_\ell), (A_0, \ldots, A_\ell)\big)$, means that z is a string of *terminal* symbols such that $A_0 \stackrel{*}{\Rightarrow} z'$, where z' is an interleaving of z and $A_1, \ldots, A_\ell$, in which A_j appears in coordinate $i_j + j - 1$. Indeed, there is a natural 1-1 correspondence between the indices $i_j \in [m]$ that are the locations in z in which the variables should be inserted, and the indices $i'_j \in [m + \ell]$, where $i'_j \stackrel{\text{def}}{=} i_j + j - 1$, that are the locations in the string $z' = z[1, i_1 - 1] \circ A_1 \circ z[i_1, i_2 - 1] \circ \cdots \circ A_\ell \circ z[i_\ell, m]$ in which the fixed variables appear.

Our construction of an IPP is recursive, and to facilitate the recursion, as discussed in Section 4.1.2.2, it will be useful for us to construct an IPP for *partial derivation languages* rather than just *context-free languages*. Additionally, as discussed in Section 4.1.2, the IPP will be proximity oblivious[26] (see Section 4.2.1.2). That is, we prove the following (more general) lemma:

Lemma 4.12. *Let G be a context-free grammar, let $\mathcal{L}$ be a* partial derivation language *corresponding to G, parameterized by $\langle \mathcal{L} \rangle = \big(n, (i_1, \ldots, i_\ell), (A_0, \ldots, A_\ell)\big)$. For every integers $k \geq 2$ and $r \geq 1$ such that $k^r \leq n$, there exists an r-round proximity oblivious* IPP *for $\mathcal{L}$ with detection probability $\rho(\varepsilon) = \varepsilon$, communication complexity $O(rk \log(n + \ell))$ and query complexity $O\big(\frac{n+\ell}{k^r}\big)$. Furthermore, the proximity oblivious* IPP *is public-coin.*

Theorem 4.3 follows directly from Lemma 4.12 by observing that (1) every context-free language is a partial derivation language, without any fixed variables (i.e., $\ell = 0$), and (2) we can transform any proximity oblivious IPP into a standard IPP (by repeating the former $O(1/\varepsilon)$ times in parallel).

Lemma 4.12 is proved in Sections 4.4.1 and 4.4.2. Specifically, in Section 4.4.1, which contains the more involved (and interesting) part of the proof, we show a scheme for partitioning partial derivation languages into several smaller partial derivation languages. Then, in Section 4.4.2 we use this partition to construct an IPP for partial derivation languages (which is a fairly straightforward implementation of the outline presented in Sections 4.1.2.2 and 4.1.2.3), as well as describe the steps required to derive an MAP (thereby proving Theorem 4.1). Finally, in Section 4.4.3 we show how to improve the efficiency of the foregoing MAP for the Dyck languages (i.e., the languages of balanced parentheses expressions).

[26] In contrast to the case of ROBPS (see Section 4.3), here we can directly use Definition 4.2 without any modifications.

4.4.1 Partitioning Partial Derivation Languages

Let $\mathcal{L} \subseteq \Sigma^n$ be a partial derivation language[27] of a context-free grammar $G = (V, \Sigma, R, A_{\mathsf{start}})$, parameterized by $\langle \mathcal{L} \rangle = \big(n, (i_1, \ldots, i_\ell), (A_0, \ldots, A_\ell)\big)$, and let $d = O(1)$ be the length of the longest production rule in R (so that every $x \in \mathcal{L}$ has a derivation tree with arity at most d).

In this section we describe a technique for partitioning $\mathcal{L}$ into several partial derivation languages $\mathcal{L}_1, \ldots, \mathcal{L}_k$ (of shorter strings), while preserving distances. That is, inputs x that belongs to $\mathcal{L}$ will be partitioned into k parts such that for every $j \in [k]$, the j^{th} part of x belongs to $\mathcal{L}_j$, whereas, for inputs x that are far from $\mathcal{L}$, the j^{th} part of x will be far from $\mathcal{L}_j$, for an average j. Later, in Section 4.4.2, we use this partition to construct an IPP for $\mathcal{L}$. (See Sections 4.1.2.2 and 4.1.2.3 for a high-level overview.)

The partition, which will be constructed jointly by the IPP prover and verifier, has two different representations. The first representation, which we call the *interval representation*, is a concise representation that is generated by the prover and sent to the verifier. The advantage of this representation is has a simple *syntactic* structure. The second representation, which is the actual partition, will be derived by the verifier from the interval representation. The main advantage of the latter representation is that it facilitates the verification of the *semantic* relation of the partition to the main input x.

We begin by describing the procedure that is used to generate the *interval representation* of the partition. The procedure, called `Generate-Intervals`(x, t), is given as input $x \in \mathcal{L}$ (recall that $\mathcal{L}$ is parameterized by $\langle \mathcal{L} \rangle = \big(n, (i_1, \ldots, i_\ell), (A_0, \ldots, A_\ell)\big)$) and a parameter $t \in [n']$, where $n' \stackrel{\text{def}}{=} n + \ell$ and t specifies the desired size of each part in the partition. We assume for simplicity that $t \geq 2d$, and the case that $t < 2d = O(1)$ will be handled separately (and trivially) in Section 4.4.2. First, the procedure constructs[28] a derivation tree T corresponding to the derivation $A_0 \stackrel{*}{\Rightarrow} x'$, where $x' \stackrel{\text{def}}{=} x[1, i_1 - 1] \circ A_1 \circ x[i_1, i_2 - 1] \circ \cdots \circ A_\ell \circ x[i_\ell, n]$ ($A_0 \stackrel{*}{\Rightarrow} x'$ follows from the fact that $x \in \mathcal{L}$). Next, using Lemma 4.5, the procedure finds $k = O(n'/t)$ rooted subtrees[29] $T_1, \ldots, T_k$ of T such that (1) every vertex of T belongs to at least one of the subtrees, and (2) for each $i < j$ either T_i and T_j are disjoint or T_i is a subtree of T_j. The procedure outputs $\overline{I} = (I_1, \ldots, I_k) \in ([n']^2)^k$ and $\overline{B} = (B_1, \ldots, B_k) \in V^k$ where B_j is the label of the root of T_j and $I_j \subseteq [n']$ is the minimal interval that contains all the leaves of T_j, for every $j \in [k]$. Each pair of intervals is either disjoint or contained in one other. The `Generate-Intervals` procedure is detailed in Fig. 4.1.

To see that `Generate-Intervals` halts with $k \leq \frac{n'}{t/d-1} \leq 2d \cdot \frac{n'}{t}$ intervals, observe that in each iteration the number of leaves of the tree T' (defined in Step 3a) decreases

[27] We suggest to the reader to consider the case that $\mathcal{L}$ is a context-free language (i.e., no variables are fixed) at first reading, since it is somewhat simpler. However, we stress that we have to handle general partial derivation languages for the recursion to go through.

[28] Although our focus is not on computational complexity, we remark that such a derivation tree can be constructed in time $\mathsf{poly}(n')$, see [HMU06] for details.

[29] Recall that we define a *subtree* of a tree T as a tree consisting of a node in T together with *all* of its descendants, see Section 4.2.3.

`Generate-Intervals`(x,t)

Input: $x \in \mathcal{L}$ (where $\mathcal{L}$ is a partial derivation language parameterized by $\big(n,(i_1,\ldots,i_\ell),(A_0,\ldots,A_\ell)\big)$) and $t \in [2d,n']$, where $n' = n+\ell$.

1. Construct a derivation tree T of arity d, with n' leaves, corresponding to the derivation $A_0 \overset{*}{\Rightarrow} x[1,i_1-1] \circ A_1 \circ x[i_1,i_2-1] \circ \cdots \circ A_\ell \circ x[i_\ell,n]$ (according to the grammar G).
2. Set $j = 1$.
3. Repeat: (prior to the j^{th} iteration, we have already constructed subtrees $T_1,\ldots,T_{j-1}$ of T).
   (a) Construct a tree T' from T by removing all the vertices of $T_{j'}$ except for the root of $T_{j'}$, for every $j' \in [j-1]$. Note that there is a natural correspondence between the vertices of T' and the vertices of T from which they were copied.
   (b) If the number of leaves of T' is less than t, then exit the loop.
   (c) Applying Lemma 4.5 to T', with size parameter t, find a subtree of T' with t' leaves such that $t' \in [t/d,t]$. Denote the root of this subtree by v'. Let v be the vertex in T that corresponds to v', and define T_i as the subtree of T rooted at v.
   (d) Increment j by 1.
4. Set $k = j$ and $T_k = T$.
5. For every $j \in [k]$, let B_j be the label of (i.e., the variable associated with) the root of T_j, and let $I_j \subseteq [n']$ be the minimal interval that contains all the leaves of T_j in T.
6. Output $(\overline{I},\overline{B})$, where $\overline{I} = (I_1,\ldots,I_k)$ and $\overline{B} = (B_1,\ldots,B_k)$.

Figure 4.1: The `Generate-Intervals` Procedure for the Partial Derivation Language $\mathcal{L}$.

additively by at least $t/d-1$ and that we assumed that $t \geq 2d$.

As noted above, the output $(\overline{I},\overline{B})$ of `Generate-Intervals` is in the first representation of the partition, which we called the interval representation. Next, we show a transformation $\mathcal{T}$ (which will be applied by the IPP verifier) that transforms the interval representation of the partition into an actual partition of the main input x.

Actually, instead of partitioning the input x into parts $S_1,\ldots,S_k \subseteq [n]$, it will be more convenient to view the partition as a partition of the *terminal* coordinates of $x' = x[1,i_1-1] \circ A_1 \circ x[i_1,i_2-1] \circ \cdots \circ A_\ell \circ x[i_\ell,n]$.[30] That is, instead of a partition of $[n]$, we will find a partition of $[n']\backslash\{i'_1,\ldots,i'_\ell\}$, where $i'_j \overset{\text{def}}{=} i_j + j - 1$, for every $j \in [\ell]$ (indeed, the non-terminal coordinates of x' are precisely $\{i'_1,\ldots,i'_\ell\}$).

Our aim is to design a transformation $\mathcal{T}$ that maps $(\overline{I},\overline{B})$ into a partition $S_1,\ldots,S_k$ of $[n']\backslash\{i'_1,\ldots,i'_\ell\}$, where the parts have roughly the same length, together with (concise descriptions of) partial derivation languages $\mathcal{L}_1,\ldots,\mathcal{L}_k$ that satisfy the following

[30] Of course, the distinction disappears in the simpler case that $\mathcal{L}$ is a context-free language (i.e., $\ell = 0$).

conditions:

- **Completeness:** If $x \in \mathcal{L}$ and $(\overline{I}, \overline{B})$ is the output of `Generate-Intervals`(x, t), then $x'[S_j] \in \mathcal{L}_j$, for every $j \in [k]$.
- **Soundness:** If x is ε-far from $\mathcal{L}$, then for *every* $(\overline{I}, \overline{B}) \in ([n']^2)^k \times V^k$ it holds that $x'[S_j]$ is ε-far from $\mathcal{L}_j$, for an average $j \in [k]$ (where the average is weighted based on the lengths of the parts).

We begin with a high-level overview of the transformation $\mathcal{T}$ in the special and slightly simpler case that $\mathcal{L}$ is a context-free language (i.e., $\ell = 0$). In this case, given input $(\overline{I}, \overline{B})$, where $\overline{I} = (I_1, \ldots, I_k)$ and $\overline{B} = (B_1, \ldots, B_k)$, the transformation first constructs a partition of $[n]$ into k parts $S_1, \ldots, S_k$ by setting $S_j = I_j \setminus (I_1 \cup \cdots \cup I_{j-1})$, for every $j \in [k]$. The transformation outputs $S_1, \ldots, S_k$ as well as (concise) descriptions of k partial derivation languages $\mathcal{L}_1, \ldots, \mathcal{L}_k$ such that for every $j \in [k]$, the language $\mathcal{L}_j$ is a partial derivation language corresponding to a partial derivation starting from B_j into strings that have variables B_{j_i} at fixed coordinates corresponding to the relative position of all subintervals I_{j_i} of I_j. The transformation also checks that the languages $\mathcal{L}_1, \ldots, \mathcal{L}_k$ are non-empty so that the distance of $x'[S_j]$ from the corresponding language $\mathcal{L}_j$ is well defined (this check is indeed necessary — see discussion in Section 4.1.2).

The case that $\mathcal{L}$ is a partial derivation language (rather than a context-free language) is quite similar, where a fairly minor complication that arises is that we need to remove the non-terminal coordinates from the partition, and so we set $S_j = I_j \setminus (I_1 \cup \cdots \cup I_{j-1} \cup \{i'_1, \ldots, i'_\ell\})$. For technical reasons, it is more convenient for us to view each one of the non-terminal coordinates $i'_1, \ldots, i'_\ell$ as an additional *artificial* singleton interval. The transformation $\mathcal{T}$ is detailed in Fig. 4.2, and the *completeness* and *soundness* requirements (which were stated loosely above) are stated formally in the following two lemmas (Lemmas 4.13 and 4.14).

Lemma 4.13 (Completeness of $\mathcal{T}$)**.** *For every $x \in \mathcal{L}$ (where $\mathcal{L}$ is a partial derivation language parameterized by $\big(n, (i_1, \ldots, i_\ell), (A_0, \ldots, A_\ell)\big)$) and parameter $t \in [2d, n']$, if $(\overline{I}, \overline{B}) \in ([n']^2)^k \times V^k$ is the output of* `Generate-Intervals`(x, t)*, then the transformation $\mathcal{T}(\overline{I}, \overline{B})$ does not reject, but rather outputs $\big((S_1, \langle \mathcal{L}_1 \rangle) \ldots, (S_k, \langle \mathcal{L}_k \rangle)\big)$ such that for every $j \in [k]$:*

1. *$\mathcal{L}_j \subseteq \Sigma^{|S_j|}$ is a partial derivation language on strings of length $n_j = |S_j|$ with ℓ_j fixed variables such that $n_j + \ell_j \leq t$; and,*
2. *$x'[S_j] \in \mathcal{L}_j$, where $x' = x[1, i_1 - 1] \circ A_1 \circ x[i_1, i_2 - 1] \circ \cdots \circ A_\ell \circ x[i_\ell, n]$.*

Proof. Let $x \in \mathcal{L}$ and let $(\overline{I}, \overline{B})$ be the output of `Generate-Intervals`(x, t), where $\overline{I} = (I_1, \ldots, I_k)$ and $\overline{B} = (B_1, \ldots, B_k)$. Since $I_k = [n']$ and $B_k = A_0$, the transformation $\mathcal{T}(\overline{I}, \overline{B})$ does not reject, but rather outputs $\big((S_1, \langle \mathcal{L}_1 \rangle), \ldots, (S_k, \langle \mathcal{L}_k \rangle)\big)$. Let $I'_1, \ldots, I'_{\ell+k}$ be as defined in $\mathcal{T}$ (see Fig. 4.2).

The fact that, for every $j \in [k]$, it holds that $\mathcal{L}_j \subseteq \Sigma^{|S_j|}$ is a partial derivation language on strings of length n_j with ℓ_j fixed variables such that $n_j + \ell_j \leq t$ follows from the fact

The Transformation $\mathcal{T}(\overline{I}, \overline{B})$

Input: $\overline{I} = (I_1, \ldots, I_k) \in ([n']^2)^k$ and $\overline{B} = (B_1, \ldots, B_k) \in V^k$ (recall that $n' = n + \ell$ and that $\langle \mathcal{L} \rangle = \big(n, (i_1, \ldots, i_\ell), (A_0, \ldots, A_\ell)\big)$).

1. Check that $(\overline{I}, \overline{B})$ is well formed: for every $j < i$ either $I_j \subsetneq I_i$ or $I_j \cap I_i = \emptyset$, and $I_k = [n']$ and $B_k = A_0$ (recall that $A_0 \in V$ is a variable such that all partial derivations in $\mathcal{L}$ start from A_0). If any test fails, then reject[a] and halt.
2. For $j \in [\ell]$, let $I'_j = \{i_j\}$.
3. For $j \in [k]$, let $I'_{\ell+j} = I_j$.
4. For every $j \in [k]$:
   (a) Let $I'_{j,1}, \ldots, I'_{j,\ell_j}$ be the ordered sequence (from left to right) of all maximal (strict) sub-intervals of $I_j = I'_{\ell+j}$ from the set of intervals $\{I'_1, \ldots, I'_{\ell+k}\}$. That is, all intervals (in order) from the set of intervals $\{I'_1, \ldots, I'_{\ell+k}\}$ that are strictly contained in I_j but are not contained in any other interval that is strictly contained in I'_j.[b]
   (b) Let $S_j = I_j \backslash (I'_{j,1} \cup \cdots \cup I'_{j,\ell_j})$.[c]
   (c) For every $s \in [\ell_j]$, let $i_{j,s} \in [|I_j|]$ be the *relative* starting position of the sub-interval $I'_{j,s}$ inside I_j, let $i'_{j,s} = i_{j,s} - \sum_{s'<s} |I'_{j,s'}|$, and let $B'_{j,s}$ be the label of the root of the subtree that corresponds to the interval $I'_{j,s}$. Define the following partial derivation language of G:
   $$\mathcal{L}_j \stackrel{\text{def}}{=} \left\{ w \in \Sigma^{|S_j|} \;:\; B_j \stackrel{*}{\Rightarrow} w[1, i'_{j,1}-1] \circ B'_{j,1} \circ w[i'_{j,1}, i'_{j,2}-1] \circ \cdots \circ B'_{j,\ell_j} \circ w[i'_{j,\ell_j}, |S'_j|] \right\}$$
   (see also Fig. 4.3). That is, $\langle \mathcal{L}_j \rangle = \big(|S_j|, (i'_{j,1}, \ldots, i'_{j,\ell_j}), (B'_{j,1}, \ldots, B'_{j,\ell_j})\big)$.
   (d) If $\mathcal{L}_j = \emptyset$, then reject and halt.[d]
5. Output $\big((S_1, \langle \mathcal{L}_1 \rangle), \ldots, (S_k, \langle \mathcal{L}_k \rangle)\big)$.

[a] In case the reader is bothered by the fact that the transformation may "reject", we can easily avoid rejecting by outputting instead some canonical representation of a "partition" that will always be rejected by the IPP verifier.

[b] In other words, an interval $I' \in \{I'_1, \ldots, I'_{\ell+k}\}$ is contained in the sequence if and only if $I' \subsetneq I_j$ and $I' \cap I'' \neq I'$, for every $I'' \in \{I'_1, \ldots, I'_{\ell+k}\} \backslash \{I'\}$ such that $I'' \subsetneq I_j$.

[c] Equivalently, $S_j = I_j \setminus (I'_1 \cup \cdots \cup I'_{\ell+j-1})$. We use the slightly more complicated definition to facilitate the proof.

[d] This check, which only requires access to $\langle L_j \rangle$ and the grammar G, can be done in $\mathsf{poly}(n')$ time.

Figure 4.2: The Transformation $\mathcal{T}$.

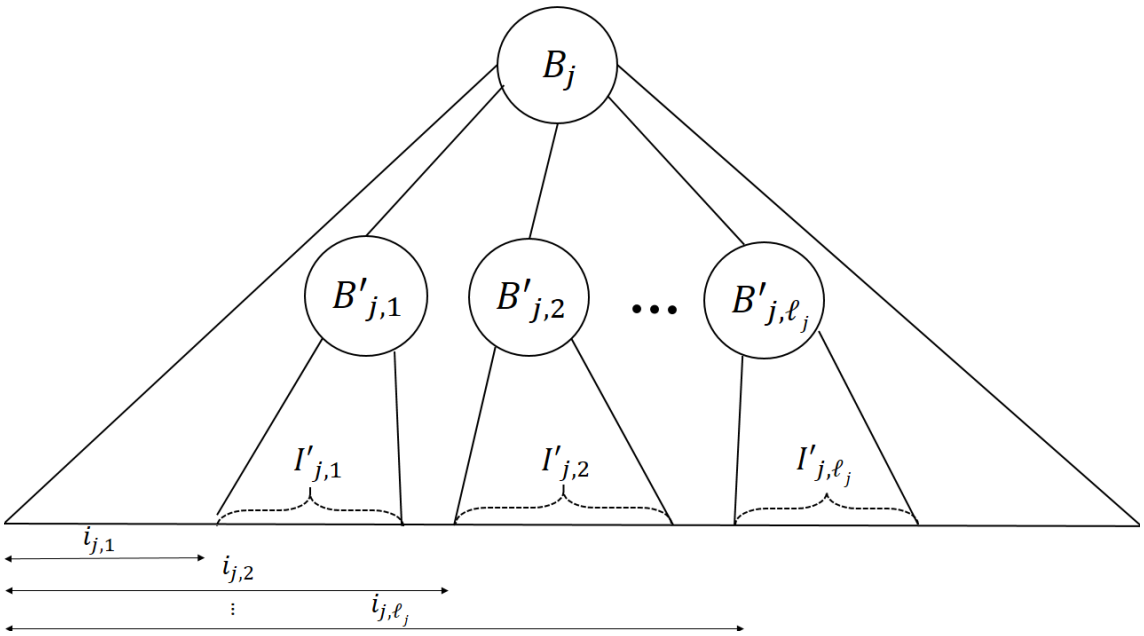

Figure 4.3: The partial derivation tree that describes the partial derivation $B_j \overset{*}{\Rightarrow} w[1, i'_{j,1} - 1] \circ B'_{j,1} \circ w[i'_{j,1}, i'_{j,2} - 1] \circ \cdots \circ B'_{j,\ell_j} \circ w[i'_{j,\ell_j}, |S'_j|]$.

that the quantity $n_j + \ell_j$ corresponds to the number of leaves of the subtree that was constructed in Item 3c in the `Generate-Intervals` procedure (recall that this subtree had at most t leaves).

To complete the proof of Lemma 4.13, we need to show that $x'[S_j] \in \mathcal{L}_j$, where $x' \overset{\text{def}}{=} x[1, i_1 - 1] \circ A_1 \circ x[i_1, i_2 - 1] \circ \cdots \circ A_\ell \circ x[i_\ell, n]$, for every $j \in [k]$. Let $j \in [k]$, and let ℓ_j, $i'_{j,1}, \ldots, i'_{j,\ell_j}, B'_{j,1}, \ldots, B'_{j,\ell_j}$ be as in Fig. 4.2. Let $w = x'[S_j]$, and observe that by construction,

$$B_j \overset{*}{\Rightarrow} w[1, i'_{j,1} - 1] \circ B'_{j,1} \circ w[i'_{j,1}, i'_{j,2} - 1] \circ \cdots \circ B'_{j,\ell_j} \circ w[i'_{j,\ell_j}, |S_i|].$$

Hence, $w \in \mathcal{L}_j$ and completeness follows.

□

Lemma 4.14 (Soundness of $\mathcal{T}$). *For every $\varepsilon \in [0, 1]$, every $x \in \Sigma^n$ that is ε-far from $\mathcal{L}$ (parameterized by $\langle \mathcal{L} \rangle = \big(n, (i_1, \ldots, i_\ell), (A_0, \ldots, A_\ell)\big)$) and every $(\overline{I}, \overline{B}) \in ([n']^2)^k \times V^k$, it holds that $\mathcal{T}(\overline{I}, \overline{B})$ either rejects or outputs a sequence $\big((S_1, \langle \mathcal{L}_1 \rangle), \ldots, (S_k, \langle \mathcal{L}_k \rangle)\big)$ such that:*

1. *The sets $S_1, \ldots, S_k \subseteq [n'] \backslash \{i'_1, \ldots, i'_\ell\}$ form a partition of $[n'] \backslash \{i'_1, \ldots, i'_\ell\}$.*
2. *It holds that*

$$\mathop{\mathbf{E}}_{j \sim \mathcal{D}} \left[\Delta_{\mathsf{REL}} \left(x'[S_j], \mathcal{L}_j \right) \right] \geq \varepsilon,$$

where $x' = x[1, i_1 - 1] \circ A_1 \circ x[i_1, i_2 - 1] \circ \cdots \circ A_\ell \circ x[i_\ell, n]$ and $\mathcal{D}$ is a distribution over $[k]$ such that $\Pr_{j \sim \mathcal{D}}[j = j'] = |S_{j'}|/n$ for every $j' \in [k]$.

Proof. Let $x \in \Sigma^n$ and let $\overline{I} = (I_1, \ldots, I_k) \in ([n']^2)^k$ be a sequence of intervals and $\overline{B} = (B_1, \ldots, B_k) \in V^k$ a sequence of variables such that the transformation $\mathcal{T}(\overline{I}, \overline{B})$ does not reject and outputs $\big((S_1, \langle \mathcal{L}_1 \rangle), \ldots, (S_k, \langle \mathcal{L}_k \rangle)\big)$. Let $I'_1, \ldots, I'_{\ell+k}$ be as defined in $\mathcal{T}$ (see Fig. 4.2).

To see that $S_1, \ldots, S_k$ form a partition of $[n'] \setminus \{i'_1, \ldots, i'_\ell\}$, observe that for each $j \in [k]$, it holds that $S_j = I_j \setminus (I'_{j,1} \cup \cdots \cup I'_{j,\ell_j})$, where $I'_{j,1}, \ldots, I'_{j,\ell_j}$ are the ordered sequence (from left to right) of all maximal sub-intervals of I'_j out of $I'_1, \ldots, I'_{\ell+k}$ (i.e., all intervals that are contained in I_j but are not contained in any other interval that is strictly contained in I_j). Thus, the S_j's are disjoint. Furthermore, since $I'_{\ell+k} = [n']$, for every index $i \in [n']$ there exists $j \in [\ell + k]$ such that $i \in I'_j$. Hence, either $i \in \{i'_1, \ldots, i'_\ell\}$ (in case $j \in [\ell]$) or $i \in S_{j'}$ for some $j' \in [k]$, and so $S_1, \ldots, S_k$ form a partition of $[n'] \setminus \{i'_1, \ldots, i'_\ell\}$.

For every $j \in [k]$, let $\varepsilon_j = \Delta_{\mathsf{REL}}(x'[S_j], \mathcal{L}_j)$. Let $\mathcal{D}$ be the distribution as in the lemma's statement (i.e., $\Pr_{j \sim \mathcal{D}}[j = j'] = |S_{j'}|/n$, for every $j' \in [k]$). Suppose that $\mathbf{E}_{j \sim \mathcal{D}}[\varepsilon_j] < \varepsilon$, for some $\varepsilon \in [0, 1]$, where $x' = x[1, i_1 - 1] \circ A_1 \circ x[i_1, i_2 - 1] \circ \cdots \circ A_\ell \circ x[i_\ell, n]$. We will show that x is ε-close to $\mathcal{L}$.

For every $j \in [k]$, since the transformation *explicitly* checks[31] (in Step 4d) that $\mathcal{L}_j \neq \emptyset$, there exists a string $z_j \in \Sigma^{|S_j|}$ such that $z_j \in \mathcal{L}_j$ and $\Delta_{\mathsf{REL}}(x'[S_j], z_j) = \varepsilon_j$ (i.e., $z_j \in \mathcal{L}_j$ minimizes the distance of $x'[S_j]$ to $\mathcal{L}_j$).

Using $z_1, \ldots, z_k$, we construct a string $z \in \mathcal{L}$ that is ε-close to x as follows. Let $z \in \Sigma^n$ such that the string $z' = z[1, i_1 - 1] \circ A_1 \circ z[i_1, i_2 - 1] \circ \cdots \circ A_\ell \circ z[i_\ell, n]$ satisfies $z'[S_j] = z_j$, for every $j \in [k]$. (The fact that such a string z exists follows from the fact that $S_1, \ldots, S_k$ are a partition of $n' \setminus \{i'_1, \ldots, i'_\ell\}$.)

Observe that $\Delta_{\mathsf{REL}}(x, z) = \Delta_{\mathsf{REL}}(x', z') \leq \mathbf{E}_{j \sim \mathcal{D}}\left[\Delta_{\mathsf{REL}}(x'[S_j], z'[S_j])\right] = \mathbf{E}_{j \sim \mathcal{D}}[\varepsilon_j] < \varepsilon$ and so x is ε-close to z. By applying the following claim, with respect to $j = k$, and using the fact that the transformation explicitly checks that $I_k = [n']$ and $B_k = A_0$, we obtain that $A_0 \stackrel{*}{\Rightarrow} z'$, and therefore $z \in \mathcal{L}$. Hence x is ε-close to a string $z \in \mathcal{L}$, and soundness follows.

Claim 4.14.1. *For every $j \in [k]$, it holds that $B_j \stackrel{*}{\Rightarrow} z'[I_j]$.*

Proof. We prove the claim by induction on j. Let $j \in [k]$, and suppose that the claim holds for every $j' < j$. Let $y = z'[S_j]$. Note that $y \in \mathcal{L}_j$. We show that $B_j \stackrel{*}{\Rightarrow} z'[I_j]$.

Recall that $I'_1, \ldots, I'_{\ell+k}$ were fixed above as in Fig. 4.2. That is, for $j \in [\ell]$, it holds that $I'_j = \{i_j\}$, and for $j \in [\ell + 1, \ell + k]$ it holds that $I'_j = I_{j-\ell}$.

Let $I'_{j,1}, \ldots, I'_{j,\ell_j}$ be the ordered maximal sub-intervals (in the set $\{I'_1, \ldots, I'_{\ell+k}\}$) of I_j. By the construction of $\mathcal{T}$ it holds that

$$\mathcal{L}_j = \left\{ w \in \Sigma^{|S_j|} : B_j \stackrel{*}{\Rightarrow} w[1, i'_{j,1} - 1] \circ B'_{j,1} \circ w[i'_{j,1}, i'_{j,2} - 1] \circ \cdots \circ B'_{j,\ell_j} \circ w[i'_{j,\ell_j}, |S_j|] \right\},$$

where $i_{j,s}$ is the *relative* starting position of the interval $I'_{j,1}$ inside I_j, $i'_{j,s} \stackrel{\text{def}}{=} i_{j,s} - \sum_{s' < s} |I'_{j,s'}|$ and $B'_{j,s}$ is the label of the subtree that corresponds to the interval $I'_{j,s}$, for

[31] Indeed, this was the reason that we added this additional check, and without it soundness would not hold. See further discussion in Section 4.1.2.

every $s \in [\ell_j]$. Therefore, since $y \in \mathcal{L}_j$, it holds that

$$B_j \overset{*}{\Rightarrow} y[1, i'_{j,1} - 1] \circ B'_{j,1} \circ y[i'_{j,1}, i'_{j,2} - 1] \circ \cdots \circ B'_{j,\ell_j} \circ y[i'_{j,\ell_j}, |S_i|]. \tag{4.3}$$

On the other hand, for every $i \in [\ell_j]$, it holds that

$$B'_{j,s} \overset{*}{\Rightarrow} z'[I'_{j,s}], \tag{4.4}$$

where Eq. (4.4) follows from the inductive hypothesis and from the fact that $B'_{j,s} = A_{j,s}$ and $z'[I'_{j,s}] = z'_{j,s} = A_{j,s}$. for $s \in [\ell_j]$.

By combining Eq. (4.3), Eq. (4.4), and the definition of $i'_{j,s}$ we obtain that

$$B_j \overset{*}{\Rightarrow} y[1, i'_{j,1}] \circ z'[I'_{j,1}] \circ y[i'_{j,1}, i'_{j,2} - 1] \circ \ldots z'[I'_{j,\ell_j}] \circ y[i'_{j,\ell_j}, |S_i|].$$

The claim follows by observing that

$$z'[I_j] = y[1, i'_{j,1}] \circ z'[I'_{j,1}] \circ y[i'_{j,1}, i'_{j,2} - 1] \circ \ldots z'[I'_{j,\ell_j}] \circ y[i'_{j,\ell_j}, |S_i|],$$

and therefore $B_j \overset{*}{\Rightarrow} z'[I_j]$. □

This completes the proof of Lemma 4.14 □

4.4.2 IPP for Partial Derivation Languages

Using Lemmas 4.13 and 4.14, we complete the proof of Lemma 4.12 (which is a relatively straightforward implementation of the ideas outlined in Section 4.1.2).

Proof of Lemma 4.12. Let $G = (V, \Sigma, R, A_{\mathsf{start}})$ be a context-free grammar. We construct a proximity oblivious IPP for every partial derivation language $\mathcal{L} \subseteq \Sigma^n$ of the grammar G.

The proximity oblivious IPP has two parameters: r which is the round complexity, and k which roughly corresponds to the amount of communication in each round. The IPP runs recursively, where each round of communication proceeds as follows. The (honest) prover uses the `Generate-Intervals` procedure on its input x and parameter $t = n'/k$ (where $n' = n+\ell$), to obtain $\left(\overline{I}, \overline{B}\right)$ and sends $\left(\overline{I}, \overline{B}\right)$ to the verifier. The verifier applies the transformation $\mathcal{T}(\overline{I}, \overline{B})$ to derive the partition $S_1, \ldots, S_k$ and the corresponding partial derivation languages $\mathcal{L}_1, \ldots, \mathcal{L}_k$. Then, the verifier selects at random $j \in [k]$ and sends j to the prover (where j is distributed according to D as above). The two parties then recurse on input $x'[S_j]$, where $x' \overset{\text{def}}{=} x[1, i_1 - 1] \circ A_1 \circ x[i_1, i_2 - 1] \circ \cdots \circ A_\ell \circ x[i_\ell, n]$, with respect to the partial derivation language $\mathcal{L}_j$. The recursion stops once either:

1. $n' \leq O(k)$ (i.e., the input is very short), in which case the prover can send $x^* = x$ to the verifier.[32] Then, the verifier checks that $x^* \in \mathcal{L}$ and that x^* is consistent with x at a randomly selected coordinate; or,

[32] This check is to ensure that the parameter $t = n'/k$ is larger than $2d$.

2. r rounds have passed, in which case the verifier reads its entire input x (which has shortened by a multiplicative factor of roughly k in each step of the recursion) and verifies that $x \in \mathcal{L}$.

The IPP for $\mathcal{L}$, denoted CFL-IPP, is detailed in Fig. 4.4.

Without loss of generality, we can measure the complexity of the protocol only when the verifier interacts with the *honest* prover (see discussion in Section 4.2.1). It can be easily verified that the round complexity is at most r rounds. By Lemma 4.13, the protocol recurses on a partial derivation language $\mathcal{L}_j$ on strings of length n_j with ℓ_j fixed variables such that $n_j + \ell_j \leq n'/k$. Hence, after at most r rounds, the current input length has length at most n'/k^r, where $n' = n + \ell$, and so the query complexity of the IPP is $O(n'/k^r)$. Since in each round the communication is at most $O(k \log n')$, the communication complexity of the IPP is $O(rk \log n')$.

Completeness. Let $\mathcal{L}$ be a partial derivation language, with $\langle \mathcal{L} \rangle \stackrel{\text{def}}{=} \left(n, \bar{i}, \overline{A}\right)$, and let $x \in \mathcal{L}$. We show that perfect completeness hold by induction on r. For $r = 0$ or $n' = O(p)$, perfect completeness follows from the fact that $\mathcal{V}$ just checks that $x \in \mathcal{L}$. For $r > 1$ (with $n'/k \geq 2d$), by Lemma 4.13, the verifier produces $\big((S_1, \langle \mathcal{L}_1 \rangle), \ldots, (S_k, \langle \mathcal{L}_k \rangle)\big)$ such that $\mathcal{L}_j$ is a partial derivation language and $x'[S_j] \in \mathcal{L}_j$, for every $j \in [k]$ (in particular, $\mathcal{L}_j \neq \emptyset$). Hence, by the inductive hypothesis, the verifier in the $r - 1$ round protocol for $\mathcal{L}_j$ will accept on input $x'[S_j]$ with probability 1.

Soundness. Soundness follows from the following lemma, which is proved by induction on the number of rounds r.

Lemma 4.15. *Let $\mathcal{L}$ be a partial derivation language, and let $k \geq 1$ and $r \geq 0$. For every $\varepsilon \in [0, 1]$ and every x that is ε-far from $\mathcal{L}$, and for every cheating prover strategy P^* it holds that:*

$$\Pr\left[\left(V, P^*\right)(x) = 0\right] \geq \varepsilon,$$

where $\mathcal{V}$ is the verifier in $\mathsf{CFL\text{-}IPP}^{\mathcal{L}}_{r,p}$ (see Fig. 4.4).

Proof. We first consider the trivial case that $n' = O(k)$. In this case, if x^* is ε-close to x, then $x^* \notin \mathcal{L}$ (since x is ε-far from $\mathcal{L}$) and the verifier rejects with probability $1 \geq \varepsilon$. Otherwise, x^* is ε-far from $\mathcal{L}$ and the verifier rejects with probability at least ε when checking the consistency of x^* and x.

We proceed to the more interesting case, in which $n'/k > 2d$, and prove by induction on r. For $r = 0$, the verifier ignores the prover and reads all of x. Hence, if $B(x) \neq 1$, then the verifier rejects with probability 1.[33]

For $r \geq 1$, let $\varepsilon \in [0, 1]$, let $x \in \Sigma^n$ be ε-far from $\mathcal{L}$, and let P^* be a *deterministic* cheating prover strategy for the protocol $\mathsf{CFL\text{-}IPP}^{\mathcal{L}}_{r,k}$ of Fig. 4.4 (with r rounds). Let $(\overline{I}, \overline{B})$ be the first message sent by P^* to $\mathcal{V}$. Assume that the invocation of the transformation

[33] In the trivial case that $\varepsilon = 0$ (i.e., $B(x) = 1$), the verifier also satisfies the requirement, since it rejects with probability at least $0 = \varepsilon$.

The Protocol $\mathsf{CFL\text{-}IPP}^{\mathcal{L}}_{k,r}$

Parameters: $\mathcal{L} \subseteq \Sigma^n$ is a partial derivation language, with $\langle \mathcal{L} \rangle = \big(n, (i_1, \ldots, i_\ell), (A_0, \ldots, A_\ell)\big)$, the parameters $k, r \in \mathbb{N}$ correspond (roughly) to the amount of communication in each round and to the number of rounds, respectively. Let $n' = n + \ell$.

Prover's Input: Direct access to $x \in \mathcal{L}$, with $n \stackrel{\text{def}}{=} |x|$.

Verifier's Input: Oracle access to x, and direct access to $\langle \mathcal{L} \rangle$.

1. If $r = 0$, then the verifier $\mathcal{V}$ checks whether $x \in \mathcal{L}$ by explicitly reading all of x. If $x \in \mathcal{L}$, then $\mathcal{V}$ accepts, otherwise it rejects, and in either case both parties terminate the protocol.
2. If $n' = O(k)$, the prover sends $x^* = x$ to $\mathcal{V}$. The verifier $\mathcal{V}$ accepts if $x^* \in \mathcal{L}$ *and* x^* agrees with x at a randomly chosen coordinate. Otherwise $\mathcal{V}$ rejects, and in either case both parties terminate the protocol.
3. The Prover $\mathcal{P}$:
   (a) Invoke $\texttt{Generate-Intervals}(x, n'/k)$ to obtain $\big(\overline{I}, \overline{B}\big)$.
   (b) Send $\big(\overline{I}, \overline{B}\big)$ to $\mathcal{V}$.
4. The Verifier $\mathcal{V}$:
   (a) Invoke $\mathcal{T}\big(\overline{I}, \overline{B}\big)$. If the transformation rejects, then immediately reject and halt. Otherwise, denote the output of the transformation by $\big((S_1, \langle \mathcal{L}_1 \rangle), \ldots, (S_k, \langle \mathcal{L}_k \rangle)\big)$.[a]
   (b) Select $j \sim \mathcal{D}$, where $\mathcal{D}$ is the distribution in the statement of Lemma 4.14 (i.e., $\Pr_{j \sim \mathcal{D}}[j = j'] = |S_{j'}|/n$, for every $j' \in [k]$).
   (c) Send j to $\mathcal{P}$.
5. Both parties (recursively) invoke $\mathsf{CFL\text{-}IPP}^{\mathcal{L}_j}_{r-1,k}$ on input $x'[S_j]$.

[a] The reader may note that, in contrast to Fig. 4.1, the verifier does not check that $\mathcal{L}_j \neq \emptyset$, for every $j \in [k]$. This check is actually performed *within* the transformation $\mathcal{T}$ (see Step 4d in Fig. 4.2).

Figure 4.4: IPP for Context-Free Languages

$\mathcal{T}\left(\overline{I}, \overline{B}\right)$ does not reject (otherwise the verifier rejects with probability 1, and we are done), and denote its output by $\left((S_1, \langle \mathcal{L}_1 \rangle), \ldots, (S_k, \langle \mathcal{L}_k \rangle)\right)$.

For every $j \in [k]$, let $\varepsilon_j = \Delta_{\mathsf{REL}}\left(x'[S_j], \mathcal{L}_j\right)$ denote the relative distance of $x'[S_j]$ from $\mathcal{L}_j$, and let $\mathcal{D}$ be the distribution as in $\mathsf{CFL\text{-}IPP}^{\mathcal{L}}_{r,k}$. By Lemma 4.14, it holds that

$$\mathop{\mathbf{E}}_{j \sim \mathcal{D}}[\varepsilon_j] \geq \varepsilon. \tag{4.5}$$

For every $j \in [k]$, let P^*_j be the residual $r-1$ round strategy of P^* after receiving the message j from $\mathcal{V}$ in the first round, and let $\mathcal{V}_j$ be the residual strategy of $\mathcal{V}$ after fixing its first message to j. Observe that, by construction, $\mathcal{V}_j$ is simply the strategy of the verifier in the protocol $\mathsf{CFL\text{-}IPP}^{\mathcal{L}_j}_{k,r-1}$. Hence, by the inductive hypothesis, for every $j \in [k]$ it holds that

$$\Pr\left[\left(\mathcal{V}_j, \mathcal{P}^*_j\right)(x'[S_j]) = 0\right] \geq \varepsilon_j. \tag{4.6}$$

Using Eqs. (4.5) and (4.6) we obtain that:

$$\Pr[\left(V, P^*\right)(x) = 0] = \mathop{\mathbf{E}}_{j \sim \mathcal{D}}\left[\Pr[\left(\mathcal{V}_j, \mathcal{P}^*_j\right)(x'[S_j]) = 0]\right] \geq \mathop{\mathbf{E}}_{j \sim \mathcal{D}}[\varepsilon_j] \geq \varepsilon, \tag{4.7}$$

and the lemma follows. □

This concludes the proof of Lemma 4.12 and Theorem 4.3. □

Remark 4.16 (Computational Complexity)**.** *The* IPP *prover in Fig. 4.4 can be implemented in time* $\mathsf{poly}(n,k,r)$*. As for the* IPP *verifier, Step 4d in Fig. 4.2 can be implemented in time* $\mathsf{poly}(n)$*, and so we obtain a total running-time of* $\mathsf{poly}(n,k,r)$*, which is super-linear. We remark that for context-free languages whose* partial derivation languages *are themselves context-free languages, we can actually do better and obtain running time* $\mathsf{poly}(\log n, k, r)$ *(an example for such a context-free language is the language of balanced parentheses expressions, see Section 4.4.3). See Section 4.D for details.*

Alternatively, by increasing the round complexity of our IPP*, we can also obtain sublinear time verification. The technique is similar to that described in Remark 4.7. More specifically, we can implement Step 4d in Fig. 4.2 (i.e., checking that a given partial derivation language is non-empty (which is the main bottleneck in our* IPP*)) via an interactive proof-system. To do so, we first construct a (logspace) uniform low-depth circuit that, given the description of a partial derivation language, outputs* 1 *if and only if the language is non-empty. An efficient interactive proof-system follows from the efficient interactive proof-system for low-depth computation of Goldwasser et al. [GKR08, Theorem 1]. Details follow.*

Fix the grammar $G = (V, \Sigma, R, A_{\mathsf{start}})$ *and consider a description* $(m, \overline{i}, \overline{A})$ *of a partial derivation language, where* $\overline{i} = (i_1, \ldots, i_\ell)$ *and* $\overline{A} = (A_0, \ldots, A_\ell)$*. Given* $(m, \overline{i}, \overline{A})$*, the circuit first constructs a string* $z \in (V \cup \{*\})^{m+\ell}$*, where* $'*'$ *is some character that does not belong to* $V \cup \Sigma$ *and* $z \stackrel{\text{def}}{=} *^{i_1-1} \circ A_1 \circ *^{i_2-i_1} \circ \cdots \circ A_\ell \circ *^{m-i_\ell+1}$*. The circuit then checks whether* z *can be derived from* A_0*, according to an auxiliary (unary) grammar* G'*, which is identical to* G *except that all the terminals are replaced by the unique terminal* $'*'$*. By*

a result of Ruzzo [Ruz81], membership in context-free languages can be computed by a (logspace uniform) NC_2 *circuit, and so we obtain a (*$O(\log(m) + \log(|\ell|))$*-space uniform) circuit that checks if the partial derivation language is non-empty, in depth* $\mathsf{polylog}(m+\ell)$ *and size* $\mathsf{poly}(m, \ell)$*.*

Given the above circuit, we can use [GKR08, Theorem 1] to obtain an interactive proof-system in which the verifier runs in $\ell \cdot \mathsf{poly}(\log(\ell), \log(m))$ *time and the prover runs in time* $\mathsf{poly}(m, \ell)$*. We note that using this proof-system inside our* IPP *increases the round complexity of our* IPP *by a poly-logarithmic factor.*

Remark 4.17 (MAPs for Context-Free Languages)**.** *Theorem 4.2 follows directly from the proof of Lemma 4.12, while noting that the two issues the arise in the case of* MAP*s for ROBPs (see Section 4.3.2) apply also here and can be resolved similarly.*

4.4.3 Improved MAPs for Specific Context-Free Languages

In this section we show that the efficiency of the MAPs for general context-free languages (i.e., Theorem 4.1) can be improved for context-free languages whose corresponding partial derivation languages have *efficient* testers (which do not use a proof). Most notably, we obtain such an improvement for the Dyck languages (i.e., languages of balanced parentheses expressions).

Recall that in the proof of Theorem 4.1, given the MAP proof, the MAP verifier (implicitly) constructs a partition $S_1, \ldots, S_k$ of $[n]$ and partial derivation languages $\mathcal{L}_1, \ldots, \mathcal{L}_k$. Then, the verifier chooses an index $j \in [k]$ at random and checks whether $x[S_j] \in \mathcal{L}_j$ by explicitly reading all of $x[S_j]$. However, by Lemma 4.14, the MAP verifier does not really have to check that $x[S_j] \in \mathcal{L}_j$ *exactly*, but rather it suffices to check that $x[S_j]$ is *close* to $\mathcal{L}_j$. Since no non-trivial tester is known for general context-free languages (let alone for their corresponding partial derivation languages), we could not use this fact to our advantage in the general case. However, for some *specific* languages, such as the Dyck languages, more efficient testers are known and we can utilize them to improve the efficiency of our MAPs.

A technical difficulty that we encounter when taking this approach is that when testing whether $x[S_j]$ is close to $\mathcal{L}_j$ it is not a priori clear which value of the proximity parameter the verifier should use (recall that Lemma 4.14 only guarantees that $x[S_j]$ is ε-far for an *average* $j \in [k]$ but not necessarily for every $j \in [k]$). Of course, if $\mathcal{L}_j$ has a *proximity-oblivious tester*, then the issue is mute and we can just run the tester directly. In the more general case, we can simply apply an averaging argument. The naive averaging argument shows that for an $\varepsilon/2$ fraction of $j \in [k]$, it holds that $x[S_j]$ is $\varepsilon/2$ far from $\mathcal{L}_j$. However, by applying a more refined averaging argument, due to Levin [Lev85] (see [Gol14, Appendix A.2]), we obtain an additional improvement.

Lemma 4.18. *Let* G *be a context-free grammar and* $\alpha \geq 0$ *and* $\beta \geq 1$ *be constants. Suppose that every partial derivation language of* G *(as in Definition 4.11) has a property tester with query complexity* $O\left(m^{\alpha} \cdot \delta^{-\beta}\right)$ *for inputs of length* m *and proximity parameter* $\delta > 0$*. Then, for every* $k \geq 1$ *the language* $\mathcal{L}$ *has an* MAP *with proof complexity* $O(k \log n)$

and query complexity $O\big((n/k)^{\alpha} \cdot \varepsilon^{-\beta} \cdot \log^2(1/\varepsilon)\big)$. *Furthermore, if* $\alpha = 0$, *then the query complexity is at most* $O\big((n/k)^{1-1/\beta} \cdot \varepsilon^{-1} \cdot \log^3(1/\varepsilon)\big)$.

The MAP in Lemma 4.18 has one-sided error if and only if the testers for the partial derivation languages have one-sided errors. However, even if the resulting MAP has two-sided error, a one-sided error MAP (with only a poly-logarithmic overhead) can be obtained by applying a generic transformation from two-sided error MAPs into one-sided error MAPs (see of [GR13b, Theorem 4.3]).

Note that the alternative bound for $\alpha = 0$ improves over the general case only for sub-constant values of the proximity parameter (i.e., $\varepsilon < (n/k)^{-1/\beta} \cdot \mathsf{polylog}(n)$). The bound is obtained by observing that, for very small values of the proximity parameter, it is advantageous to read the entire input rather than apply the tester. We defer the proof of Lemma 4.18, which is relatively straightforward, to Section 4.C.

Using Lemma 4.18 we now show how to construct an improved MAP for the Dyck languages. Loosely speaking, the κ^{th}*-order Dyck language* consists of all of strings that form a balanced parenthesis expression with κ distinct types of parentheses. The Dyck languages can be defined via a context-free grammar as follows.

Definition 4.19. *Let* $\kappa \in \mathbb{N}$ *be a constant. The* κ^{th}-order Dyck language, *denoted* Dyck_κ, *is the language generated by the context-free grammar* $G_{\mathsf{Dyck}_\kappa} = (V, \Sigma_\kappa, R, A_{\mathsf{start}})$, *where* $V = \{A\}$, $A_{\mathsf{start}} = A$, $\Sigma_\kappa = \{`[_1', `]_1', `[_2', `]_2', \ldots, `[_\kappa', `]_\kappa'\}$, *and the production rules* R *consist of: (1)* $A \Rightarrow [_i\, A\,]_i$ *for every* $i \in [\kappa]$, *(2)* $A \Rightarrow AA$, *(3)* $A \Rightarrow \lambda$, *where* λ *denotes the empty string.*

Alon *et al.* [AKNS00] showed a tester (with two-sided error) for the first order Dyck language (i.e., Dyck_1) with query complexity $\tilde{O}(1/\varepsilon^2)$. As for higher order Dyck languages, Parnas *et al.* [PRR01] showed that any Dyck language (of any fixed order) can be tested (with two-sided error) by making $O(n^{2/3} \cdot \varepsilon^{-3})$ queries.[34] Furthermore, by the following proposition, the foregoing results can be extended to the case of *partial derivation languages* of the Dyck languages (with respect to the foregoing grammars).

Proposition 4.20. *Let* $m, \kappa \in \mathbb{N}$. *If* $\mathcal{L} \subseteq (\Sigma_\kappa)^m$ *is a partial derivation language of the grammar* G_{Dyck_κ}, *then* $\mathcal{L}$ *is equal to* $\mathsf{Dyck}_\kappa \cap (\Sigma_\kappa)^m$.

Proof. Let $\mathcal{L} \subseteq (\Sigma_\kappa)^m$ be a partial derivation language of G_{Dyck_κ} such that $\langle \mathcal{L} \rangle = \big(m, (i_1, \ldots, i_\kappa), (A, \ldots, A)\big)$ (here we used the fact that the grammar G_{Dyck_κ} uses only a single variable – A).

On one hand, if $x \in \mathcal{L}$, then $A \overset{*}{\Rightarrow} x[1, i_1 - 1] \circ A \circ x[i_1, i_2 - 1] \circ \cdots \circ A \circ x[i_\ell, m]$. Using the production rule $A \Rightarrow \lambda$ we have that $A \overset{*}{\Rightarrow} x[1, i_1 - 1] \circ x[i_1, i_2 - 1] \circ \cdots \circ x[i_\ell, m] = x$ and therefore $x \in \mathsf{Dyck}_\kappa \cap (\Sigma_\kappa)^m$.

On the other hand, if $x \in \mathsf{Dyck}_\kappa \cap (\Sigma_\kappa)^m$, then $A \overset{*}{\Rightarrow} x$. The following claim shows that, for the Dyck grammars, we can generate a partial derivation in which A is inserted in any desired sequence of positions. Therefore, $A \overset{*}{\Rightarrow} x[1, i_1 - 1] \circ A \circ x[i_1, i_2 - 1] \circ \cdots \circ A \circ x[i_\ell, m]$, which implies that $x \in \mathcal{L}$.

[34] For perspective, recall that Parnas *et al.* [PRR01] also showed that, for $\kappa \geq 2$, any tester (which does not use a proof) for Dyck_κ must make at least $\tilde{\Omega}(n^{1/11})$ queries.

Claim 4.20.1. *Let $\alpha \in (\Sigma_\kappa \cup \{A\})^{m'}$, for some $m' \in \mathbb{N}$, and let $i \in [m']$. If $A \overset{*}{\Rightarrow} \alpha$, then $A \overset{*}{\Rightarrow} \alpha[1, i-1] \circ A \circ \alpha[i, m']$.*

Proof. Since $A \overset{*}{\Rightarrow} \alpha$ (according to the grammar G_{Dyck_κ}), there exists a corresponding partial derivation tree T, in which all internal vertices are labeled by the variable A and each leaf is labeled by either 'A', '$[_j$', '$]_j$', for some $j \in [\kappa]$. We prove the claim by extending T into a partial derivation tree T' that corresponds to the partial derivation $A \overset{*}{\Rightarrow} \alpha[1, i-1] \circ A \circ \alpha[i, m']$.

Denote the i^{th} leaf of T by v and denote v's parent by u. The specific way in which T' is constructed from T depends on whether the label of v is 'A', '$[_j$' or '$]_j$' (for some $j \in [\kappa]$), and is detailed in Fig. 4.5.

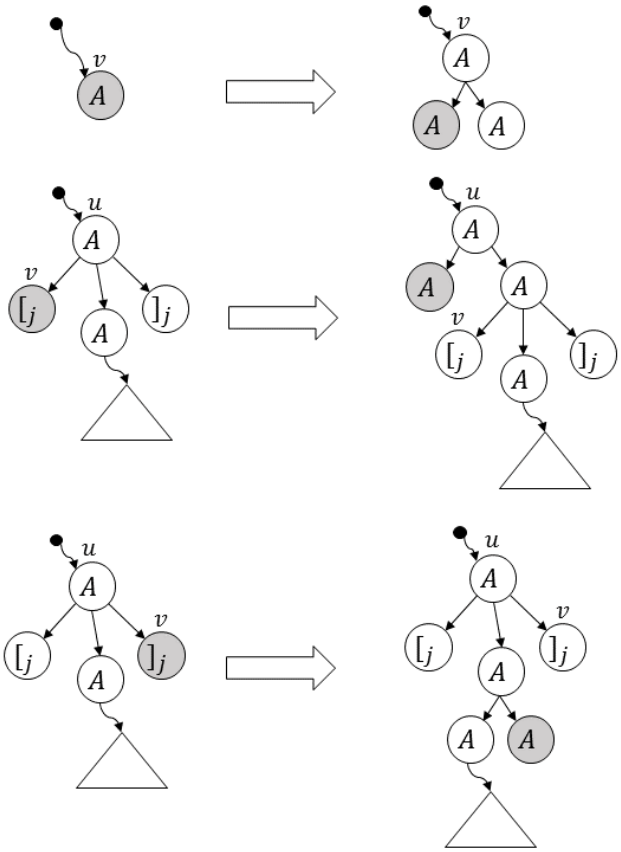

Figure 4.5: Construction of T' from T. The original tree T is on the left, and the new tree T' is on the right. In each case the i^{th} leaf of the tree has a shaded background, both in T and in T' (note that in all cases the i^{th} leaf of T is v and the i^{th} leaf of T' is labeled by A, the newly inserted symbol).

□

This concludes the proof of Proposition 4.20. □

Thus, the property testers of [PRR01] for the Dyck languages are also testers for the partial derivation languages (of the Dyck languages), and we obtain the following result.

Theorem 4.6. *Let $\kappa \geq 2$. For every p such that $2 \leq p \leq n$, there exists an* MAP *for* Dyck_κ *that uses a proof of length $O(p \log n)$ and has query complexity $O\big((n/p)^{2/3} \cdot \varepsilon^{-3} \cdot \log^2(1/\varepsilon)\big)$. Furthermore, there exists an* MAP *with* one-sided error *for* Dyck_κ *that uses a proof of length $O(p \log n + \mathsf{polylog}(n))$ and has query complexity $(n/p)^{2/3} \cdot \varepsilon^{-3} \cdot \mathsf{polylog}(n)$.*

The furthermore clause is obtained by applying the generic transformation from one-sided error MAP into two-sided error MAP (see [GR13b, Theorem 4.3]) and using the fact that without loss of generality we may assume that $\varepsilon \geq 1/n$ (and so $\log^2(1/\varepsilon) \leq \mathsf{polylog}(n)$). We conclude this section with some second order remarks.

Improvement for Dyck_1 and $\varepsilon \ll 1/\sqrt{n}$. For Dyck_1 (i.e., $\kappa = 1$), and for small values of the proximity parameter (i.e., $\varepsilon < \frac{1}{\sqrt{n} \cdot \mathsf{polylog}(n)}$) we can improve Theorem 4.6, by using the tester of Alon *et al.* [AKNS00] (which has query complexity $\tilde{O}(1/\varepsilon^2)$). Using the special case of Lemma 4.18, we obtain query complexity $O\big(\sqrt{n/p} \cdot \varepsilon^{-1} \cdot \log^3(1/\varepsilon)\big)$ with a proof of length $O(p \log n)$.

Extension to IPPs. The idea of applying non-trivial testers can also be used to obtain improved IPPs, by applying the tester after the last round of interaction (instead of running the trivial tester that reads the entire (current) input). The savings in this case are less significant since the query and communication complexities of our IPPs are already fairly small. Hence, we only elaborate briefly on these IPPs below.

If the partial derivation languages of the grammar have *proximity-oblivious* testers, then the latter can simply be employed in the last step of the recursion in Fig. 4.4. However, if only standard testers (which are not proximity oblivious) are available, then we can generalize the strategy in the proof of Lemma 4.18 by applying an averaging argument in each step of the recursion, while incurring an $\tilde{O}(1/\varepsilon)$ multiplicative overhead in each round. Unfortunately, the latter strategy results in an exponential dependence on the round complexity of the protocol.

Computational Complexity for Dyck Languages. In general, as noted in Remark 4.16, the running time of the verifier in Fig. 4.4 is $\mathsf{poly}(n)$ (because it verifies that each of the languages $\mathcal{L}_1, \ldots, \mathcal{L}_k$ is non-empty). However, as shown in Proposition 4.20, for the Dyck languages, the partial derivation languages $\mathcal{L}_1, \ldots, \mathcal{L}_k$ are themselves Dyck languages. Since the Dyck language on m-bit strings is non-empty if and only if m is even, the running time of the verifier can be reduced to $\mathsf{poly}(\log n, k, r)$ (see also Section 4.D).

The MAP proof in Theorem 4.6 is generated efficiently (i.e., in time $\mathsf{poly}(n)$) for every context-free language, and in particular for the Dyck language. However, for the furthermore clause of Theorem 4.6, we apply the transformation of [GR13b, Theorem 4.3], which in general does not preserve *computational* efficiency of the proof generating procedure. Hence, we do not obtain an MAP for the Dyck languages that simultaneously has both one-sided error and an efficient procedure of generating the MAP proof.

Appendix for Chapter 4

4.A Parallel Repetition of IPPs

The k-fold parallel repetition of an IPP $(\mathcal{V}_1, \mathcal{P}_1)$ is an IPP $(\mathcal{V}_k, \mathcal{P}_k)$ in which the two parties perform k parallel repetitions of $(\mathcal{V}_1, \mathcal{P}_1)$, using independent random coins for each invocation. Note that the query and communication complexities of $(\mathcal{V}_k, \mathcal{P}_k)$ are k times the query and communication complexities of $(\mathcal{V}_1, \mathcal{P}_1)$, respectively. The verifier $\mathcal{V}_k$ accepts if $\mathcal{V}_1$ accepts in a majority of the k invocations. For our applications it suffices to focus on the case that $(\mathcal{V}_1, \mathcal{P}_1)$ has a one-sided error, in which case $\mathcal{V}_k$ can just check that $\mathcal{V}_1$ accepts in *all* the k invocations.

It is clear that if $(\mathcal{V}_1, \mathcal{P}_1)$ has perfect completeness, then so does $(\mathcal{V}_k, \mathcal{P}_k)$. The main challenge is in proving that the soundness error decreases exponentially with k since if P^* is the optimal cheating strategy against $\mathcal{V}$, it is not a priori clear that the optimal cheating strategy against $\mathcal{V}_k$ is k independent copies of $\mathcal{P}^*$.

Nevertheless, the following lemma, taken verbatim from [Gol99, Lemma C.1] shows that the soundness error for any interactive machine $\mathcal{V}_k$ does decrease exponentially.

Lemma 4.21 ([Gol99, Lemma C.1])**.** *Let $\mathcal{V}_1$ be an interactive machine, and $\mathcal{V}_k$ be an interactive machine obtained from $\mathcal{V}_1$ by playing k versions of $\mathcal{V}_1$ in parallel. Let*

$$p_1(x) \stackrel{\text{def}}{=} \max_{\mathcal{P}^*} \{\Pr[(\mathcal{P}^*, \mathcal{V}_1)(x) = 1]\}, \text{ and}$$
$$p_k(x) \stackrel{\text{def}}{=} \max_{\mathcal{P}^*} \{\Pr[(\mathcal{P}^*, \mathcal{V}_k)(x) = 1]\}.$$

Then,

$$p_k(x) = (p_1(x))^k.$$

We stress that Lemma 4.21 holds for any x and is independent of the operation of $\mathcal{V}_1$. It holds as long as $\mathcal{V}_k$ executes k independent copies of $\mathcal{V}_1$ and accepts if all copies accept. Hence, it holds also when $\mathcal{V}_1$ is an IPP verifier; in that case $\mathcal{V}_k$ has query complexity that is k times that of $\mathcal{V}_1$.

4.B Computing ROBPs in Low-Depth

For any branching program B (including branching programs that are not *read-once*), we show that the language $\mathcal{L}_B = \{x \in \{0,1\}^* : B(x) = 1\}$ can be recognized by

a $\mathsf{poly}(|B|, n)$-size circuit of depth $O((\log(|B|))^2)$ (with fan-in 2). We stress that the branching program B is fixed and the circuit only gets x as input. For simplicity, we assume without loss of generality that B has a *unique* accepting sink (otherwise we can add a new unique accepting sink t and have all former accepting sinks direct to t).

The idea (which is in essence the folklore proof that (non-deterministic) log-space is contained in NC_2) proceeds as follows. First, based on the input x (and the fixed branching program B), compute a $|B| \times |B|$ matrix M_x whose $(u,v)^{\text{th}}$ entry is 1 if the branching program traverses from the vertex $u \in B$ to $v \in B$ on input x *in a single step*. In addition, for every sink $t \in B$ we set the $(t,t)^{\text{th}}$-entry of M_x to 1 (these correspond to self loops). All other entries of M_x are set to 0. Given input x, the matrix M_x (which is a permutation matrix) can be computed by a constant-depth circuit of size $\mathsf{poly}(|B|)$ (in fact, every entry in M_x is either a fixed constant, or equal to some variable or its negation).

Observe that for every $k \geq 1$, the $(u,v)^{\text{th}}$-th entry of $(M_x)^k$ is equal to 1 if and only if the branching program traverses from u to v, on input x, in k steps (or at most k steps if v is a sink). Hence, to check whether the source s leads to the (unique) *accepting* sink t on input x, it suffices to check whether the $(s,t)^{\text{th}}$-th entry of $(M_x)^{|B|}$ is equal to 1. Using repeated squaring we can compute $(M_x)^{|B|}$ in $O(\log^2(|B|))$ depth and we obtain a circuit as required.

4.C Proof of Lemma 4.18

We proceed to describe the MAP, which is similar to the MAP of Theorem 4.1 except that we use the guaranteed property testers for the partial derivation languages. Given $x \in \mathcal{L}$, the MAP proof is the output $(\overline{I}, \overline{B})$ of `Generate-Intervals`(x,t) (see Fig. 4.1), where $t = n/k$ and as in the proof of Theorem 4.1 we assume that $t \geq 2d$. The MAP verifier, given direct access to $(\overline{I}, \overline{B})$ and oracle access to $x \in \Sigma^n$, first runs $\mathcal{T}(\overline{I}, \overline{B})$ to obtain $(S_1, \langle \mathcal{L}_1 \rangle), \ldots, (S_\ell, \langle \mathcal{L}_\ell \rangle)$ and rejects if $\mathcal{T}$ rejects. Otherwise, the verifier runs the following procedure for every $j \in \left[\lceil \log_2(2/\varepsilon) \rceil \right]$:

1. Select uniformly at random $O\left(\frac{\log(1/\varepsilon)}{2^j \varepsilon}\right)$ indices in $[\ell]$. Denote the chosen indices by $\mathcal{I}$.

2. For every index $i \in \mathcal{I}$, run the property tester for $\mathcal{L}_i$ on input $x[S_i]$ (while simulating its oracle queries with queries to x), with respect to proximity parameter 2^{-j} and with completeness and soundness errors $\mathsf{poly}(\varepsilon)$ (as usual, the latter can be obtained by taking the majority of $O(\log(1/\varepsilon))$ independent tests). If the tester rejects then reject and halt.

If none of the above test fails then the verifier accepts.

We first show that completeness and soundness hold and later show that the query complexity is as stated.

Completeness. If $x \in \mathcal{L}$, by Lemma 4.13, the transformation $\mathcal{T}$ produces as output $\big((S_1, \langle \mathcal{L}_1 \rangle), \ldots, (S_k, \langle \mathcal{L}_k \rangle)\big)$ such that $\mathcal{L}_j$ is a partial derivation language and $x[S_j] \in \mathcal{L}_j$, for every $j \in [k]$. Since the tester for each partial derivation language $\mathcal{L}_j$ has completeness error $\mathsf{poly}(\varepsilon)$ and we perform Step 2 $O(\varepsilon^{-1} \cdot \log^2(1/\varepsilon))$ times in total, the verifier accepts in all tests with probability at least $2/3$. Furthermore, if the testers for the partial derivation languages have a one-sided error, then the MAP verifier accepts with probability 1 and otherwise we can apply a generic transformation (as discussed in the beginning of the proof) to obtain a one-sided error.

Soundness. Let $x \in \Sigma^n$ that is ε-far from $\mathcal{L}$, and let $(\overline{I}, \overline{B})$ be an alleged proof. By Lemma 4.14, the transformation $\mathcal{T}$ either rejects (in which case the verifier rejects and we are done), or produces $\big((S_1, \langle \mathcal{L}_1 \rangle), \ldots, (S_k, \langle \mathcal{L}_\ell \rangle)\big)$, where $S_1, \ldots, S_\ell$ form a partition of $[n]$ and $\mathcal{L}_j$ is a partial derivation language, such that x is ε-far from $\big\{z \in \Sigma^n : \forall j \in [k], z[S_j] \in \mathcal{L}_j\big\}$. The following claim, which is a refined averaging argument, shows that either there are many indexes $i \in [k]$ such $x[S_i]$ is mildly far from $\mathcal{L}_i$ or there are few indexes $i \in [\ell]$ such that $x[S_i]$ is extremely far from $\mathcal{L}_i$ (or anything in between).

Lemma 4.22 (Precision Sampling)**.** *There exists $j^* \in [\lceil \log_2 2/\varepsilon \rceil]$ such that for a $\frac{2^{j^*}\varepsilon}{4 \cdot \lceil \log_2(2/\varepsilon) \rceil}$ fraction of the indexes $i \in [\ell]$ it holds that $x[S_i]$ is 2^{-j^*}-far from $\mathcal{L}_i$.*

For completeness, we provide the proof of Lemma 4.22, which is standard.

Proof. Let $d \stackrel{\text{def}}{=} \lceil \log_2(2/\varepsilon) \rceil$. Recall that $\Delta_{\mathsf{REL}}(z, W)$ is the minimal *relative* Hamming distance of z from the set W. For every $k \in [d]$, let

$$B_k \stackrel{\text{def}}{=} \left\{ i \in [\ell] : \Delta_{\mathsf{REL}}(x[S_i], \mathcal{L}_i) \in \left(2^{-k}, 2^{-(k-1)}\right] \right\},$$

and let $B_{d+1} = [\ell] \backslash (\cup_{i \in [d]} B_k)$. Note that the sets $B_0, \ldots, B_d, B_{d+1}$ form a partition $[\ell]$. Also note that by our setting of d, for every $i \in B_{d+1}$ it holds that $x[S_i]$ is $\varepsilon/2$-close to $\mathcal{L}_i$.

Suppose towards a contradiction that for every $k \in [d]$ it holds that $|B_k| < \frac{2^k \varepsilon}{4d} \cdot \ell$. Using the fact that for every $i \in B_k$ it holds that $x[S_i]$ is $2^{-(k-1)}$-close to $\mathcal{L}_i$, we obtain that

$$\begin{aligned}
\Delta_{\mathsf{REL}}(x, \mathcal{L}) &\leq \frac{1}{\ell} \sum_{i=1}^{\ell} \Delta_{\mathsf{REL}}(x[S_i], \mathcal{L}_i) \\
&= \frac{1}{\ell} \sum_{i \in B_{d+1}} \Delta_{\mathsf{REL}}(x[S_i], \mathcal{L}_i) + \frac{1}{\ell} \sum_{k \in [d]} \sum_{i \in B_k} \Delta_{\mathsf{REL}}(x[S_i], \mathcal{L}_i) \\
&\leq \frac{|B_{d+1}|}{\ell} \cdot \frac{\varepsilon}{2} + \frac{1}{\ell} \sum_{k \in [d]} 2^{-(k-1)} \cdot |B_k| \\
&< \frac{\varepsilon}{2} + \sum_{k \in [d]} \frac{\varepsilon}{2d} \\
&= \varepsilon,
\end{aligned}$$

in contradiction to our assumption that x is ε-far from $\mathcal{L}$. □

Next, consider the execution of iteration j^* of the verifier, where j^* is as guaranteed by Lemma 4.22. Since the verifier selects uniformly at random $O\left(\frac{\log(1/\varepsilon)}{2^{j^*}\varepsilon}\right)$ indices in $[k]$, with probability at least $9/10$ it selects at least one index $i \in [k]$ such that $x[S_i]$ is 2^{-j^*}-far from $\mathcal{L}_i$. In this case, the tester for $\mathcal{L}_i$, with respect to proximity parameter 2^{-j^*} will reject $x[S_i]$ with probability $1 - \mathsf{poly}(\varepsilon)$. Thus, the verifier rejects x with probability at least $(1 - \mathsf{poly}(\varepsilon)) \cdot 9/10 \geq 2/3$.

Query Complexity. Recall that we assumed that every partial derivation language has a tester with query complexity $Q(m, \delta) = O(m^\alpha \cdot \delta^{-\beta})$, for inputs of length m with respect to proximity parameter $\delta > 0$. By definition, it holds that $|S_i| \leq t = n/p$, for every $i \in [\ell]$. Thus, the query complexity is at most

$$\begin{aligned}\sum_{j\in[\lceil \log_2 2/\varepsilon\rceil]} \sum_{i\in\mathcal{I}} \left(\log(1/\varepsilon) \cdot Q\left(n/k, 2^{-j}\right)\right) &= O\left(\log(1/\varepsilon) \cdot \sum_{j\in[\lceil \log_2 2/\varepsilon\rceil]} 2^{j\beta} \cdot \frac{\log(1/\varepsilon)}{2^j \cdot \varepsilon} \cdot (n/k)^\alpha\right) \\ &= O\left((n/k)^\alpha \cdot \frac{(\log(1/\varepsilon))^2}{\varepsilon} \cdot \sum_{j\in[\lceil \log_2 2/\varepsilon\rceil]} 2^{(\beta-1)j}\right) \\ &= O\left((n/k)^\alpha \cdot \varepsilon^{-\beta} \cdot \log^2(1/\varepsilon)\right).\end{aligned}$$

For the particular case in which $\alpha = 0$, we tighten the analysis for small values of ε by noting that the query complexity for any language is upper bounded by the size of the object:

$$\begin{aligned}\sum_{j\in[\lceil \log_2 2/\varepsilon\rceil]} \sum_{i\in\mathcal{I}} \left(\log(1/\varepsilon) \cdot Q\left(n/k, 2^{-j}\right)\right) &= O\left(\log(1/\varepsilon) \cdot \sum_{j\in[\lceil \log_2 2/\varepsilon\rceil]} \frac{\log(1/\varepsilon)}{2^j \cdot \varepsilon} \cdot \min\left(n/k, 2^{j\beta}\right)\right) \\ &= O\left((n/k)^{1-1/\beta} \cdot \varepsilon^{-1} \cdot \log^3(1/\varepsilon)\right),\end{aligned}$$

where the last equality follows since $\min\left(n/k, 2^{j\beta}\right) \leq (n/k)^{1-1/\beta} \cdot \left(2^{j\beta}\right)^{1/\beta}$, for every $j \geq 1$ (while using the fact that $\beta \geq 1$). Note that $\log^3(1/\varepsilon) \leq \mathsf{polylog}(n)$ since without loss of generality we may assume that $\varepsilon \geq 1/n$.

4.D Efficient Verification for Special Context-Free Languages

As stated in Remark 4.16, in this section we show that for special context-free languages we can improve the running time of the verifier in Fig. 4.4 from $\mathsf{poly}(n, k, r)$ to $\mathsf{poly}(\log n, k, r)$. Specifically, we refer to context-free languages whose *partial derivation*

languages are themselves context-free languages (e.g., the Dyck language, see Proposition 4.20).

The crucial step in improving the verifier's running-time is an efficient implementation of Item 4d in Fig. 4.2. In the general case, this step can be implemented in time $\mathsf{poly}(n)$, but we show that if the partial derivation languages are context-free languages, then we obtain running time $\mathsf{polylog}(n)$.

Lemma 4.23. *For every context-free language $\mathcal{L}$ over an alphabet Σ, there exist an algorithm that given an integer $n \in \mathbb{N}$, runs in time $\mathsf{polylog}(n)$ and accepts if and only if $\mathcal{L} \cap \Sigma^n \neq \emptyset$.*

Proof. Let G be a context-free grammar that accepts $\mathcal{L}$, and let G' be the context-free grammar that is obtained from G by replacing all the terminal symbols in G by a single terminal symbol, denoted 0. Note that $\mathcal{L} \cap \Sigma^n \neq \emptyset$ if and only if G' accepts 0^n.

Observe that the language $\mathcal{L}'$ accepted by G' is a *unary* context-free language. Ginsburg and Rice [GR62] showed that such a language must be *regular*.

Proposition 4.24 ([GR62])**.** *Every* unary *context-free language is* regular.

Hence, there exists a finite-state automaton over the unary alphabet that accepts $\mathcal{L}'$. Such an automaton can be viewed as a directed graph with a single outgoing edge from each node. Hence, the graph is a directed path (from the start node) of length a feeding into a directed cycle of length b, and some of the nodes are accepting. Hence, the accepted lengths have the form $j + i \cdot b$, where $j \in [a + b - 1]$ and $i \geq 0$.

The lemma follows by observing that an algorithm can easily check in $\mathsf{polylog}(n)$ time if the given input n has the desired form, by checking if $n - j$ is divisible by b, for the specific set of $j \in [a + b - 1]$ that correspond to accepting nodes of the automaton. □

Chapter 5

Arguments of Proximity

5.1 Introduction

With the prominent use of computers, tremendous amounts of data are available. For example, hospitals have massive amounts of medical data. This data is very precious as it can be used, for example, to learn important statistics about various diseases. This data is often too large to store locally, and thus is often stored on cloud platforms (or external servers). As a result, if a hospital (which has bounded storage and bounded computational power), wishes to perform some computation on its medical data, it would need to delegate this computation to the cloud. Since the cloud's computation may be faulty, the party delegating the computation (say, the hospital), may want a proof that the computation was done correctly. It is important that this proof can be verified very efficiently, and that the prover's running time is not much larger than the time it takes to perform the computation, since otherwise, the solution will not be practical.

This problem is closely related to the problem of computation delegation, where a weak client delegates a computation to a powerful server, and the server needs to provide the client with a proof that the computation was done correctly. In contrast to the current setting, in the setting of computation delegation, the input is thought of as being small and the computation is thought of as being large. The client (verifier) is required to run in time that is proportional to the input size (but much smaller than the time it takes to do the computation), and the powerful server (prover) runs in time polynomially related to the time it takes to do the computation. Indeed the problem of computation delegation is extremely important, and received a lot of attention (e.g., [GKR08, Mic94, Gro10, GGP10, CKV10, AIK10, GLR11, Lip12, BCCT12a, DFH12, BCCT12b, GGPR12, PRV12, KRR13a, KRR13b]).

In reality, however, the input (data) is often very large, and the client cannot even store the data. Hence, we seek a solution in which the client runs in time that is *sub-linear* in the input size. The question is:

If the client cannot read the data, how can he verify the correctness of a computation on the data?

The work of [CKLR11], on memory delegation, considers this setting where the input (thought of as the client's memory) is large, and the client cannot store it locally. However, in memory delegation, it is assumed that the client (verifier) stores a short "commitment" of the input, and then can verify computations in sub-linear time. However, computing such a commitment takes time at least linear in the input length, which is infeasible in many settings.

Recently, Rothblum, Vadhan and Wigderson [RVW13], in their work on interactive proofs of proximity (IPP, a notion first studied by Ergün, Kumar and Rubinfeld [EKR04]), provide a solution where the verifier does not need to know such a commitment. Without such a commitment, the verifier cannot be sure that the computation is correct (since he cannot read the entire input), however they guarantee that the input is "close" to being correct. More specifically, they construct an interactive proof system for every language computable by a (log-space uniform) low depth circuit, where the verifier is given *oracle access* to the input (the data), and the verifier can check whether the input is *close* to being in the language in *sub-linear* time in the input (and linear time in the depth of the computation). We note that in many settings where the data is large (such as medical data) and the goal is to compute some statistics on this data, an approximate solution is acceptable.

The work of [RVW13] is the starting point of our work.

5.1.1 Our Results in a Nutshell

We depart from the interactive proof of proximity setting, and consider *arguments of proximity*. In contrast to proofs of proximity, in an argument of proximity, soundness is required to hold only against *computationally bounded* cheating provers. Namely, the soundness guarantee is that any bounded cheating prover can convince the verifier to accept an input that is far from the language (in Hamming distance) only with small probability. By relaxing the power of the prover we obtain stronger results.

We construct *one-round* arguments of proximity for every deterministic language (without a dependency on the depth). Namely, fix any $t = t(n)$ and any language $\mathcal{L} \in \mathsf{DTIME}(t(n))$, we construct a one-round argument of proximity for $\mathcal{L}$ where the verifier runs in time $o(n) + \mathsf{polylog}(t)$, assuming the existence of a sub-exponentially secure fully homomorphic encryption (FHE) scheme.

Our one-round argument of proximity is constructed in two steps, and follows the outline of the recent works of Kalai *et al.* [KRR13a, KRR13b]. These works first show how to construct an MIP for all deterministic languages, that is sound against *no-signaling strategies*. Such no-signaling soundness is stronger than the typical notion of soundness, and is inspired by quantum physics and by the principal that information cannot travel faster than light (see Section 5.3.5 for the definition, and [KRR13a, KRR13b] for more background on this notion). They then show how to convert these no-signaling MIPs into one-round arguments.

As our first step, we combine the interactive proof of proximity (IPP) of [RVW13],

and the no-signaling MIP construction of [KRR13b], to obtain a no-signaling *multi-prover interactive proof of proximity* (MIPP). (See Theorem 5.2 and Theorem 5.12.) This construction combines techniques and results of [RVW13] and [KRR13b], and may be of independent interest.

Then, similarly to [KRR13a], we show how to convert any no-signaling MIPP to a one-round argument of proximity. (See Theorem 5.1 and Theorem 5.16.) This transformation relies on a heuristic developed by Aiello *et al.* [ABOR00], which uses a (computational) PIR scheme (or a fully homomorphic encryption scheme) to convert any MIP into a one-round argument. This heuristic was proven to be secure in [KRR13a] if the underlying MIP is secure against no-signaling strategies. We extend the result of [KRR13a] to the proximity setting.

Finally, we provide a negative result, which shows that the parameters we obtain for MIPP and the parameters obtained in [RVW13], are somewhat tight. (See Theorem 5.4, and Theorems 5.7 and 5.8.) Proving such a lower bound was left as an open problem in [RVW13]. This part contains several new ideas, and is the main technical contribution of this work.

We also show that the parameters in our one-round argument of proximity are somewhat optimal, for arguments which have adaptive soundness and are proven to be adaptively sound via a black-box reduction to a falsifiable assumption; see Section 5.2 and Section 5.4.3 for details.

Linear-time delegation. We observe that both proofs and arguments of proximity, aside from being natural notions, can also be used as tools to obtain new results for delegating computation in the standard setting (i.e., where soundness is guaranteed for *every* $x \notin \mathcal{L}$). More specifically, using our results on arguments of proximity and the [RVW13] results on interactive proofs of proximity for low-depth circuits, we can construct (standard) one-round argument-systems for any deterministic computation, and interactive proof systems for low-depth circuits, where the verifier truly runs in *linear-time*. In contrast, the results of [GKR08] and [KRR13b] only give a *quasi-linear* time verifier.[1] See Section 5.2 for details.

5.1.2 Our Results in More Detail

Our main result is a construction of a one-round argument of proximity for any deterministic language. Here, and throughout this work, we use n to denote the input length. Let $t = t(n)$, let $\mathcal{L} \in \mathsf{DTIME}(t)$ be a language. For a proximity parameter $\varepsilon = \varepsilon(n) \in (0, 1)$, we denote by ε-IPP an interactive proof for testing ε-proximity to $\mathcal{L}$.[2] Similarly we denote by ε-MIPP a multi-prover interactive proof for testing ε-proximity to $\mathcal{L}$.

[1] Actually, by an observation of Vu *et al.* [VSBW13] (see also [Tha13, Lemma 3]), the verifier in the [GKR08] protocol can be directly implemented in linear-time. However the latter implementation would only guarantee *constant* soundness error.

[2] A string $x \in \{0,1\}^n$ is ε-close to $\mathcal{L}$ if there exists $x' \in \{0,1\}^n \cap \mathcal{L}$ such that $\triangle(x, x') \leq \varepsilon n$, where $\triangle$ denotes the Hamming distance between the two strings.

Theorem 5.1 (Informal, see Theorem 5.16)**.** *Suppose that there exists a sub-exponentially secure* FHE*. Fix a proximity parameter* $\varepsilon \stackrel{\text{def}}{=} n^{-(1-\beta)}$*, for some sufficiently small* $\beta > 0$*, and a security parameter* τ *(polynomially related to* n*).*

There exists a 1-round argument of ε*-proximity for* $\mathcal{L}$*, where the verifier runs in time* $n^{1-\gamma} + \mathsf{polylog}(t) + \mathsf{poly}_{\mathsf{FHE}}(\tau)$*, where* $\gamma > 0$ *is a constant and* $\mathsf{poly}_{\mathsf{FHE}}$ *is a polynomial that depends only on the* FHE *scheme, and makes* $n^{1-\gamma} + \mathsf{polylog}(t)$ *oracle queries to the main input. The prover runs in time* $\mathsf{poly}(t)$*. The total communication is of length* $\mathsf{poly}_{\mathsf{FHE}}(\tau)$*.*

Note that for languages in $\mathsf{DTIME}(2^{n^\alpha})$ for sufficiently small $\alpha > 0$ (and in particular for languages in P), the verifier in Theorem 5.1 runs in *sub-linear* time.

As mentioned previously, this result is obtained in two steps. We first construct an MIPP that is sound against no-signaling strategies, and then show how to convert any such MIPP into a one-round argument of proximity.

Theorem 5.2 (Informal, see Theorem 5.12)**.** *Fix a proximity parameter* $\varepsilon = \varepsilon(n) \in (0,1)$*. There exists an* ε-MIPP *that is secure against no-signaling strategies, where the verifier makes* $q = (1/\varepsilon)^{1+o(1)}$ *oracle queries to the input, the communication complexity* $c = (\varepsilon n)^2 \cdot n^{o(1)} \cdot \mathsf{polylog}(t)$ *and the running time of the verifier is* $(\varepsilon n)^2 \cdot \mathsf{polylog}(t) + \left(\frac{1}{\varepsilon} + \varepsilon n\right)^{1+o(1)}$*.*

We then show how to convert any no-signaling ε-MIPP to a one-round argument of ε-proximity. In the following we say that a fully homomorphic encryption scheme (FHE) is (T, δ) secure if every family of circuits of size T can break the semantic security of the FHE with probability at most δ.

Theorem 5.3 (Informal, see Theorem 5.18)**.** *Fix a proximity parameter* $\varepsilon = \varepsilon(n) \in (0,1)$*. Suppose that the language* $\mathcal{L}$ *has an* ℓ*-prover* ε-MIPP *that is sound against* δ*-no-signaling strategies, with communication complexity* c*. Suppose that there exists a* $(T, \delta/\ell)$*-secure* FHE*, where* $T \geq 2^c$*. Then* $\mathcal{L}$ *has a 1-round argument of* ε*-proximity where the running time of the prover and verifier and the communication complexity of the argument system, are proportional to those of the underlying* MIPP *scheme.*

We note that the parameters in Theorem 5.2 are somewhat similar to the parameters of the interactive proof of proximity (IPP) in [RVW13]. In particular, in both constructions it holds that $c \cdot q = \Omega(n)$. The work of [RVW13] shows that this lower bound of $c \cdot q = \Omega(n)$ is inherent for IPPs with 2-messages (and that a weaker bound holds for IPPs with a constant number of rounds), and left open the question of whether this lower bound is inherent for general (multi-round) IPPs.

We resolve this question by showing that for every ε-IPP, and every ε-MIPP that is sound against no-signaling strategies, it must be the case that $c \cdot q = \Omega(n)$. For this result we assume the existence of exponentially hard pseudorandom generators.

Theorem 5.4 (Informal, see Theorem 5.7 and Theorem 5.8)**.** *Assume the existence of exponentially hard pseudorandom generators. There exists a constant* $\varepsilon > 0$ *such that for every* $q = q(n) \leq n$*, there exists a language* $\mathcal{L} \in \mathsf{P}$ *such that for every* ε-IPP *for* $\mathcal{L}$

, and for every ε-MIPP for $\mathcal{L}$ that sound against no-signaling adversaries, it holds that $q \cdot c = \Omega(n)$, where q is the query complexity and c is the communication complexity.

In fact, assuming a slightly stronger cryptographic assumption, we can replace $\mathcal{L} \in \mathsf{P}$ with $\mathcal{L} \in \mathsf{NC}_1$ (which shows that the [RVW13] upper bound for log-space uniform NC is essentially tight). See Section 5.4 for details.

We note that the [RVW13] lower bound for 2-message IPPs is unconditional (and in particular they do not assume that the verifier is *computationally* bounded). It remains an interesting open problem to obtain an *unconditional* lower bound for multi-message IPPs.

The parameters we obtain for the one-round argument also satisfy $q \cdot c = \Omega(n)$. We show that these parameters are close to optimal for arguments with adaptive soundness, that are proven sound via a black-box reduction to falsifiable assumptions. We refer the reader to Section 5.4.3 for details.

Finally, using the [RVW13] protocol or the protocol of Theorem 5.1 we construct delegation schemes in which the verifier runs in *linear-time*.

Theorem 5.5 (Informal, see details in Section 5.2)**.** *For every language in (logspace-uniform) NC there exists an interactive proof system in which the verifier runs in time $O(n)$ and the prover runs in time $\mathsf{poly}(n)$.*

Theorem 5.6 (Informal, see details in Section 5.2)**.** *Assume that there exists a sub-exponentially secure FHE. Then, for every language in P there exists a 1-round argument-system in which the verifier runs in time $O(n)$ and the prover runs in time $\mathsf{poly}(n)$.*

5.1.3 Related Work

As mentioned above, the work of [RVW13] and [KRR13a, KRR13b] are most related to ours. Both our work, and the work of [RVW13], lie in the intersection of property-testing and computation delegation. As opposed to property testing, where an algorithm is required to decide whether an input is close to the language *on its own* in sub-linear time, in our work the algorithm receives a proof, and only needs to verify correctness of the proof in sub-linear time. Thus, our task is significantly easier than the task in property testing. Indeed we get much stronger results. In particular, the works on property testing typically get sub-linear algorithms for specific languages, whereas our result holds *for all deterministic languages*.[3]

Another very related problem is that of constructing a *probabilistically checkable proof of proximity* (PCPP) [BSGH+06] (also known as *assignment testers* [DR06]). A PCPP consists of a prover who publishes a long proof, and a verifier, who gets oracle access to this proof and to the instance x, and needs to decide whether x is close to the language in sub-linear time. The significant difference between PCPP and proofs/argument of

[3]Indeed, as shown by Goldwasser, Goldreich and Ron [GGR98], there are properties in very low complexity classes that require $\Omega(n)$ queries and running-time in order to test (without the help of a prover).

proximity is that in the PCPP setting the proof is a fixed string (and cannot be modified adaptively based on the verifier's messages).

The fundamental works of Kilian and Micali [Kil92, Mic94] show how to convert any probabilistically checkable proof (PCP) into a 2-round (4-message) argument. As pointed out by [RVW13], their transformation can be also used to convert any PCPP into a 2-round argument of proximity. Thus, obtaining a 2-round argument of proximity follows immediately by applying the transformation of [Kil92, Mic94] to any PCPP construction. Moreover, the parameters of the resulting 2-round argument are optimal (up to logarithmic factors); i.e., the query complexity, the communication complexity and the runtime of the verifier is $\mathsf{poly}(\log(t), \tau)$ where t is the time it takes to compute if x is in the language, and where τ is the security parameter.

The focus of this work is on constructing *one-round* arguments of proximity. Unfortunately, our parameters do not match those of the two-round arguments of proximity outlined above. However, we show that using our techniques (i.e., of constructing one-round arguments of proximity from no-signaling MIPPs), our parameters are almost optimal.

Other works that are related to ours are the work of Gur and Rothblum [GR13b] on non-interactive proofs of proximity, and of Fischer *et al.* [FGL14] on partial testing. The former studies an NP version of property testing (which can be thought of as a 1-message variant of IPP), whereas the latter studies a model of property testing in which the tester needs to only accept a sub-property (we note that the two notions, which were developed independently, are tightly related, see [GR13b, FGL14] for details).

Organization

In Section 5.2 we give a high level view of our techniques. In Section 5.3 we formally define arguments of proximity and the other central definitions that are used throughout this work. In Section 5.4 we show our lower bounds. In Section 5.5 we construct a no-signaling MIPP for every language in P. Lastly, in Section 5.6 we show how to transform the no-signaling MIPP into an argument of proximity.

5.2 Our Techniques

5.2.1 Our Positive Results

To construct arguments of proximity for languages in $\mathsf{DTIME}(t)$, we adapt the technique of [KRR13a] to the "proximity" setting. That is, we first construct an MIPP that has soundness against no-signaling strategies and then employ the technique of Aiello *et al.* [ABOR00] to obtain an argument of proximity. We elaborate on these two steps below. In what follows, we focus for simplicity on languages in P, though everything extends to languages in $\mathsf{DTIME}(t)$.

No-Signaling MIPPs for P. Our first step (which is technically more involved) is a construction of MIPPs that are sound against no-signaling strategies for any language

$\mathcal{L} \in \mathsf{P}$. This construction is inspired by (and reminiscent of) the IPP construction of [RVW13]. The starting point for the [RVW13] IPP is the "Muggles" protocol of Goldwasser *et al.* [GKR08], whereas our starting point is the no-signaling MIP of [KRR13b].

The main technical difficulty in using both the [GKR08] and [KRR13b] protocols by a sublinear time verifier is that in both protocols, the verifier needs to compute an error corrected encoding of the input x. More specifically, the verifier needs to compute the low degree extension of x, denoted LDE_x (see Section 5.3.8 for a definition of LDE). Since, by design, error-correcting codes are very sensitive to changes in the input, a sub-linear algorithm has no hope of computing LDE_x.

The key point is that in both the [GKR08] and the [KRR13b] protocols, it suffices for the verifier to check the value of LDE_x at relatively few *randomly* selected points (this property was also used by [CKLR11] in their work on memory delegation). Hence, it will be useful for us to view both the [GKR08] and [KRR13b] protocols as protocols for producing a sequence of points J in the low degree extension of x and a sequence of corresponding values $\vec{v}$ with the following properties:

- If $x \in \mathcal{L}$ and the prover(s) honestly follow the protocol then $\mathsf{LDE}_x(J) = \vec{v}$.
- If $x \notin \mathcal{L}$ then no matter what the cheating prover does (resp., no-signaling cheating prover do), with high probability the verifier outputs $J, \vec{v}$ such that $\mathsf{LDE}_x(J) \neq \vec{v}$.

Hence, the verifiers in both protocols first run this subroutine to produce J and $\vec{v}$ and then accept if and only if $\mathsf{LDE}_x(J) = \vec{v}$. Remarkably, in both cases, in the protocol that produces J and $\vec{v}$, the verifier does not need to access x.

The next step in [RVW13] is a parallel repetition of the foregoing protocol in order to reduce the soundness error. Once the soundness error is sufficiently small, [RVW13] argue that for every x that is ε-far from $\mathcal{L}$, no matter what the cheating prover does (in the parallel repetition of the base protocol), the verifier will output $J, \vec{v}$ such that not only $\mathsf{LDE}_x(J) \neq \vec{v}$, but furthermore, x is far from *any* x' such that $\mathsf{LDE}_{x'}(J) = \vec{v}$. This steps simply follows by taking a union bound over all x' that are close to x.

We borrow this step almost as-is from [RVW13] except for the following technical difficulty - it is not known whether parallel repetition decreases the soundness error of no-signaling MIP protocols.[4] However, we observe that the [KRR13b] protocol already allows for sufficient flexibility in choosing its soundness error so that the parallel repetition step can be avoided.

The last step of [RVW13] is designing an IPP protocol for a language that they call $\mathsf{PVAL}_{J,\vec{v}}$ (for "polynomial evaluation"). This language, parameterized by J and $\vec{v}$, consists of all strings x such that $\mathsf{LDE}_x(J) = \vec{v}$. Using this IPP for PVAL, the IPP verifier for a language $\mathcal{L}$ first runs the (parallel repetition of the) [GKR08] protocol, to produce $J, \vec{v}$ as above. Then, the IPP verifier runs the $\mathsf{PVAL}_{J,\vec{v}}$ protocol and accepts if and only if the PVAL-verifier accepts. If $x \in \mathcal{L}$ then we know that $\mathsf{LDE}_x(J) = \vec{v}$ and therefore the PVAL-verifier will accept, whereas if x is far from $\mathcal{L}$ then x is far from $\mathsf{PVAL}_{J,\vec{v}}$ and therefore

[4] Holenstein [Hol09] showed a parallel repetition theorem for no-signaling 2-prover MIPs. It is not known whether this result can be extended to 3 or more provers.

the PVAL-verifier will reject. Hence the (parallel repetition of the) [GKR08] protocol is sequentially composed with the IPP for PVAL.

For the no-signaling case, we also use the [RVW13] IPP protocol for PVAL. A technical difficulty that arises is that in contrast to the IPP setting in which sequential composition (of two interactive proofs) is trivial, here we need to compose a 1-round no-signaling MIP with an IPP protocol, to produce a no-signalling MIPP. We indeed prove that such a composition holds thereby constructing a no-signaling MIPP as we desire. See Section 5.5 for details.

From No-Signaling MIPP to Arguments of Proximity. The transformation from a no-signaling MIPP to an argument of proximity is based on the assumption that there exists a fully homomorphic encryption scheme (or alternatively, a computational private information retrieval scheme) and is practically identical to that in [KRR13a]. More specifically, the argument's verifier uses the MIPP verifier to generate a sequence of queries $q_1, \ldots, q_\ell$ to the ℓ provers. It encrypts each query using a fresh encryption key as follows: $\hat{q}_i \leftarrow Enc_{k_i}(q_i)$. The argument's verifier sends all the encrypted queries to the prover. Given $\hat{q}_1, \ldots, \hat{q}_\ell$, the prover uses the homomorphic evaluation algorithm to compute the MIPP answers "underneath" the encryption. It sends these answers back to the verifier, which can decrypt the encrypted answers and decide. As in [KRR13a] we show that if the MIPP is sound against no-signaling strategies then, assuming the semantic security of the FHE, the resulting protocol is sound against computationally bounded adversaries. See Section 5.6 for details.

Linear-time delegation. We show that using the foregoing one-round argument of proximity for every language $\mathcal{L} \in \mathsf{P}$ and good error-correcting codes, one can easily construct a one-round delegation protocol where the verifier runs in *linear* time (in contrast, the verifier in [KRR13b] runs in *quasi*-linear time). A similar observation, in the context of PCPs, was previously pointed out by [EKR04].

Let $\mathcal{L} \in \mathsf{P}$ and consider $\mathcal{L}' = \{\mathsf{ECC}(x) : x \in \mathcal{L}\}$ where ECC is an error correcting code with constant rate, constant relative distance, linear-time encoding and polynomial-time decoding[5]. Then, $\mathcal{L}' \in \mathsf{P}$ and so it has an argument of proximity with a sublinear-time verifier. We construct a delegation scheme for $\mathcal{L}$ by having both the verifier and the prover compute $x' = \mathsf{ECC}(x)$ and run the argument of proximity protocol with respect to x'. Since the argument of proximity verifier runs in sublinear time, and $\mathsf{ECC}(x)$ can be computed in linear-time, the resulting delegation verifier runs in linear-time. Soundness follows from the fact that a cheating prover that convinces the argument-system verifier to accept $x \notin L$ can be used to convince the argument-of-proximity verifier to accept $\mathsf{ECC}(x)$ which is indeed far from $\mathcal{L}'$.

A similar result can be obtained for interactive proofs for low-depth computation based on the results of [RVW13] by using an error-correcting code that can be decoded in logarithmic-depth (such a code was constructed by Spielman [Spi96]).

[5]Such codes are known to exist, see, e.g., [Spi96].

5.2.2 Our Negative Results

We prove that assuming the existence of exponentially hard pseudorandom generators, there exists a constant $\varepsilon > 0$ for which there does not exist a no-signaling ε-MIPP for all of P with query complexity q and communication complexity c such that $q \cdot c = o(n)$ (where n is the input length). We also show a similar result for ε-IPP.

We start by focusing on our lower bound for MIPP. The high-level idea is the following: Suppose (towards contradiction) that every language in P has a no-signaling MIPP with query complexity q and communication complexity c where $q \cdot c = o(n)$. The fact that $q = o(n)$ implies that (for every language in P), there is some set of coordinates $S \subseteq [n]$ of size $O(n/q)$ that with high (constant) probability the verifier does not query.

As a first step, suppose for the sake of simplicity that there is a fixed (universal) set of coordinates $S \subseteq [n]$ such that with high probability the verifier never queries the coordinates in S, for every language in P (for example, if the verifier's queries are non-adaptive and are generated before it communicates with the prover, then such a set S must exist). We derive a contradiction by showing that one can use the no-signaling MIPP to construct a no-signaling MIP for languages in $\mathsf{NP} \setminus \mathsf{P}$ with communication $c = o(n)$. The latter was shown to be impossible, assuming that $\mathsf{NP} \not\subseteq \mathsf{DTIME}(2^{o(n)})$ [DLN+04] (see also [Ito10]).

The basic idea is the following: Take any language $\mathcal{L} \in \mathsf{NP} \setminus \mathsf{P}$ that is assumed to be hard to compute in time $2^{o(n)}$, and convert it into the language $\mathcal{L}' \in \mathsf{P}$, defined as follows: $x' \in \mathcal{L}'$ if and only if x'_S is a valid witness of $x'_{[n]\setminus S}$ in the underlying NP language $\mathcal{L}$. The no-signaling MIP for $\mathcal{L}$ will simply be the no-signaling ε-MIPP for $\mathcal{L}'$, where the MIP verifier simulates the ε-MIPP verifier with oracle access to x' where $x'_{[n]\setminus S} = x$, and $x'_S = 0^{|S|}$. Note that the MIP verifier, which takes as input x (supposedly in $\mathcal{L}$), cannot (efficiently) generate a corresponding witness w and set $x'_S = w$. But the point is that it does not need to, since S was chosen so that with high probability the MIPP verifier for $\mathcal{L}'$ will not query x' on coordinates in S.

There are several problems with this approach. First, the witness can be very long compared to x, and the set S may be very small compared to n. In this case we will not be able to fit the entire witness in the coordinate set S. Second, after running the MIPP, the verifier is convinced that x' is close to an instance in $\mathcal{L}'$. However, this does not imply that x is in $\mathcal{L}$ (and can only imply that x is close to $\mathcal{L}$).

One can fix these two problems with a single solution: Instead of setting $x'_{[n]\setminus S} = x$ we set $x'_{[n]\setminus S} = \mathsf{ECC}(x)$, where ECC is a error-correcting code with efficient encoding, that is resilient to 2ε-fraction of errors. Now, we can take $\mathsf{ECC}(x)$ so that $|\mathsf{ECC}(x)|$ is very large compared to $|w|$, so that we can fit all of the witness in the coordinate set S. Moreover, if $|\mathsf{ECC}(x)| > |w|$ then if x' is ε-close to $\mathcal{L}'$ then $x'_{[n]\setminus S}$ is 2ε-close to $\mathcal{L}$. This, together with the fact that $\mathsf{ECC}(x)$ is resilient to 2ε-fraction of errors implies that the encoded element is indeed in $\mathcal{L}$.

The foregoing idea indeed seems to work if there was a fixed (universal) set S that the MIPP verifier does not query (with high probability). However, this is not necessarily the case, and this set S may be different for different languages in P. In particular, we

cannot claim that for the language $\mathcal{L}'$ the set S is exactly where the witness lies. Namely, it may be that the verifier in the underlying MIPP always queries some coordinates in S.

We solve this problem by using repetitions. Namely, every element $x' \in \mathcal{L}'$ will consist of many instances (encoded using an error-correcting code) along with many witnesses; i.e., $x' = (\mathsf{ECC}(x_1, \ldots, x_m), w_1, \ldots, w_m)$, where each w_j is a witness for the NP statement $x_j \in \mathcal{L}$. Now, suppose that the verifier makes q queries to x' (where $q = o(n)$). Then if we take $m = 4q$ then we know that 3/4 of the (x_j, w_j)'s are not queried.

As above, we derive a contradiction by showing that one can use the no-signaling MIPP to construct a no-signaling MIP for languages in $\mathsf{NP} \setminus \mathsf{P}$ with $o(n)$ communication, (which is known to be impossible for languages that cannot be computed in time $2^{o(n)}$ [DLN+04, Ito10]). However, now the no-signaling MIP construction will be different: Given an instance x (supposedly in $\mathcal{L}$), the MIP verifier will choose a random $i^* \in_R [m]$, along with m random instance and witness pairs $(x_1, w_1), \ldots, (x_m, w_m)$, where $x_{i^*} = x$ and w_{i^*} can be arbitrary (assumed not to be queried).

We need to argue that with probability at least 3/4 the verifier will not query the coordinates of w_{i^*}, and thus with probability at least 3/4 the MIP verifier will successfully simulate the MIPP verifier. If the queries of the MIPP verifier were chosen before interacting with the prover then this would follow immediately from the fact that $i^* \in [m]$ is chosen at random. However, the MIPP verifier may choose its oracle queries after interacting with the MIPP provers, and therefore we need to argue that the MIPP provers also do not know i^*. Note that the MIPP provers see all of $x_1, \ldots, x_m$. Hence, in order to claim that the provers cannot guess i^* it needs to be the case that x is distributed identically to the other $x_1, \ldots, x_m$.

Hence, we seek a language $\mathcal{L} \in \mathsf{NP} \setminus \mathsf{P}$ for which there exists a distribution $\mathcal{D}$ (distributed over $\mathcal{L}$) such that:

1. It is computationally hard to distinguish between $x \in_R \mathcal{D}$ and $x \notin \mathcal{L}$ (i.e., $\mathcal{L}$ is hard on the average); and

2. $x \in_R \mathcal{D}$ can be sampled together with a corresponding NP witness.

We note that the first requirement is needed to obtain a contradiction (and replaces the weaker assumption that $\mathcal{L} \in \mathsf{NP} \setminus \mathsf{P}$) whereas the second assumption is required so that we can sample $x_1, \ldots, x_m$ (together with the corresponding witnesses) so that MIPP protocol cannot distinguish between x and any of the x_j's (thereby hiding i^*). In can be easily verified that both requirement are met by considering $\mathcal{D}$ which is the output of a cryptographic pseudorandom generator (PRG). Hence the language $\mathcal{L}$ that we use is precisely the output of such a PRG.

Indeed, we can only argue that our no-signaling MIP has *average-case* completeness (with respect to the distribution $\mathcal{D}$), since if $x \in \mathcal{L}$ is distributed differently from $(x_1, \ldots, x_m)$ then the verifier of the MIPP may always query the coordinates where the witness of x is embedded, in which case the MIP verifier will fail to simulate. However, for random $x \in_R \mathcal{L}$ the provers (and verifier) in the MIPP cannot guess i^* with any non-negligible advantage, and therefore the verifier will not query the coordinates of w_{i^*}

with probability at least 3/4, in which case the MIP verifier will succeed in simulating the underlying ε-MIPP verifier. We refer the reader to Section 5.4 for further details.

A Lower Bound for IPP. To obtain a multiplicative lower bound for IPP, we follow the same paradigm outlined above for MIPP's with no-signaling soundness. More specifically, we consider a language $\mathcal{L} \in \mathsf{NP}$ and the corresponding language

$$\mathcal{L}' = \left\{ \left(\mathsf{ECC}(x_1, \ldots, x_m), w_1, \ldots, w_m\right) : \; w_j \text{ is an } \mathsf{NP}\text{-witness for } x_j \right\}$$

as above. We show that an IPP protocol for $\mathcal{L}'$ implies a (standard) interactive-proof for $\mathcal{L}$ with similar communication complexity. Here we obtain a contradiction by arguing that (assuming exponential hardness) there are languages in NP\P for which every interactive proof require $\Omega(n)$ communication. The latter is based on the proof that $\mathsf{IP} \subseteq \mathsf{PSPACE}$ (i.e., the "easy" direction in the $\mathsf{IP} = \mathsf{PSPACE}$ theorem).

Given the [RVW13] positive result of IPP for low depth computations, we would like to show that our lower bound is not just for languages in P but even for languages, say, in NC_1 (thereby showing that the [RVW13] result is tight). To do so we observe that if (1) the error correcting code that we use has an encoding procedure that can be computed by an NC_1 circuit and (2) the cryptographic PRG can be computed in NC_1, then indeed $\mathcal{L}' \in \mathsf{NC}_1$. We refer the reader to Section 5.4.2 for further details.

A Lower Bound for One-Round Arguments of Proximity. For one-round arguments of proximity, we show a similar lower-bound of $q \cdot c = \Omega(n)$, assuming the argument has *adaptive* soundness (as defined in Section 5.4.3), and the proof of (adaptive) soundness is via a *black-box reduction* to some *falsifiable* cryptographic assumption.

Loosely speaking, a cryptographic assumption is falsifiable (a notion due to Naor [Nao03]) if there is an *efficient* way to "falsify it", i.e., to demonstrate that it is false. We note that most standard cryptographic assumptions (e.g., one-way functions, public-key encryption, LWE etc.) are falsifiable. A black-box reduction of one cryptographic primitive to another, is a reduction that, using black-box access to any (possibly inefficient) adversary for the first primitive, breaks the security of the second primitive.

Similarly to the MIPP and IPP lower bounds, we consider the languages $\mathcal{L}$ and $\mathcal{L}'$, as above, where $\mathcal{L} \in \mathsf{NP}$ is exponentially hard on average and $\mathcal{L} \in \mathsf{P}$. We prove that if there exists an adaptively sound one-round argument of proximity for $\mathcal{L}'$ with $q \cdot c = o(n)$ then there exists an adaptively sound one-round argument for $\mathcal{L}$ with $o(n)$ communication (in the crs model).

We then rely on a result of Gentry and Wichs [GW11], which shows that there does not exist a one-round argument for exponentially hard (on average) NP languages, with adaptive soundness and black-box reduction to a falsifiable assumption.

We conclude that P does not have an adaptively sound one-round argument of proximity with $q \cdot c = o(n)$, and a black-box reduction to a falsifiable assumption. We refer the reader to Section 5.4.3 for details.

5.3 Preliminaries

5.3.1 Notation

For $x, y \in \{0,1\}^n$, we denote the Hamming distance of x and y by $\Delta(x,y) \stackrel{\text{def}}{=} |\{i \in [n] \colon x_i \neq y_i\}|$. We say that x is ε-close to y if $\Delta(x,y) \leq \delta$. We say that x is ε-close to a set $S \subseteq \{0,1\}^n$ if there exists $y \in S$ such that x is ε-close to y.

If A is an oracle machine, we denote by $A^x(z)$ the output of A when given oracle access to x and explicit access to z.

For a vector $a = (a_1, \ldots, a_\ell)$ and a subset $S \subseteq [\ell]$, we denote by a_S the sequence of elements of a that are indexed by indices in S, that is, $a_S = (a_i)_{i \in S}$. In general, we denote by a_S a sequence of elements indexed by S, and we denote by a_i the i^{th} coordinate of a vector a.

For a distribution $\mathcal{A}$, we denote by $a \in_R \mathcal{A}$ a random variable distributed according to $\mathcal{A}$ (independently of all other random variables).

We will measure the distance between two distributions by their *statistical distance*, defined as half the l_1-distance between the distributions. We will say that two distributions are δ-*close* if their statistical distance is at most δ.

5.3.2 Arguments of Proximity

An interactive argument of proximity for a language $\mathcal{L}$ consists of a polynomial-time verifier that wishes to verify that x is close (in Hamming distance) to some x' such that $x' \in \mathcal{L}$, and a prover that helps the verifier to decide. The verifier is given as input $n \in \mathbb{N}$, a proximity parameter $\varepsilon = \varepsilon(n) > 0$ and oracle access to $x \in \{0,1\}^n$ (and its oracle queries are counted). The prover gets as input ε and x. The two parties interact and at the end of the interaction the verifier either accepts or rejects. We require that if $x \in \mathcal{L}$ then the verifier accepts with high probability but if x is ε-far from $\mathcal{L}$, then no *computationally bounded* prover can convince the verifier to accept with non-negligible (in n) probability.

We focus on 1-round arguments of proximity systems. Such an argument-system consists of a single message sent from the verifier V to the prover P, followed by a single message sent from the prover to the verifier.

Let $\varepsilon = \varepsilon(n) \in (0,1)$ be a proximity parameter. Let $T : \mathbb{N} \to \mathbb{N}$ and $s : \mathbb{N} \to [0,1]$ be parameters. We say that (V,P) is a one-round argument of ε-proximity for $\mathcal{L}$, with soundness (T,s), if the following two properties are satisfied:

1. **Completeness:** For every $x \in \mathcal{L}$, the verifier $V^x(|x|,\varepsilon)$ accepts with overwhelming probability, after interacting with $P(\varepsilon,x)$.

2. **Soundness:** For every family of circuits $\{P^*_n\}_{n \in \mathbb{N}}$ of size $\mathsf{poly}(T(n))$ and for all sufficiently large $x \notin \mathcal{L}$, the verifier $V^x(|x|,\varepsilon)$ rejects with probability $\geq 1 - s(|x|)$, after interacting with $P^*_{|x|}(\varepsilon,x)$.

5.3.3 Interactive Proofs of Proximity (IPP)

We define interactive proofs of proximity (IPP), following [RVW13]. In an IPP for a language $\mathcal{L}$, a *single* computationally unbounded prover, P, tries to convince a (probabilistic) polynomial-time verifier, V, that the input x is close (in Hamming distance) to some $x' \in \mathcal{L}$. The prover has free access to ε and x. The verifier has free access to $n = |x|$ and ε but only has oracle access to x (and the number of oracle queries is counted).

Let $\varepsilon = \varepsilon(n) \in (0,1)$ be a proximity parameter. We say that (V,P) is an interactive proof of ε-proximity (ε-IPP) for $\mathcal{L}$, with completeness $c \in [0,1]$ and soundness $s \in [0,1]$, if the following properties are satisfied:

1. **Running Time:** The verifier runs in polynomial time, i.e., time polynomial in the communication complexity and the number of oracle queries.
2. **Completeness:** For every $x \in \mathcal{L}$, the verifier V accepts with probability c, after interacting with P.
3. **Soundness:** For every x that is ε-far from $\mathcal{L}$, and any (computationally unbounded, possibly cheating) prover P^*, the verifier V rejects with probability $\geq 1 - s$, after interacting with P^*.

We denote such a proof system by ε-IPP (and omit the soundness and completeness parameters from the notation). We say that the proof-system has *perfect completeness* if completeness hold with probability 1 (i.e. $c = 1$). The parameters we are mainly interested in are the query complexity and the communication complexity.

We say that an IPP is *public-coin* if each one of the verifier's messages is a uniformly random string.

5.3.4 Multi-Prover Interactive Proofs (MIP)

Let $\mathcal{L}$ be a language and let x be an input of length n. In a one-round ℓ-prover interactive proof, ℓ computationally unbounded provers, $P_1, \ldots, P_\ell$, try to convince a (probabilistic) $\mathsf{poly}(n)$-time verifier, V, that $x \in \mathcal{L}$. The input x is known to all parties.

The proof consists of only one round. Given x and its random string, the verifier generates ℓ queries, $q_1, \ldots, q_\ell$, one for each prover, and sends them to the ℓ provers. Each prover responds with an answer that depends only on its own individual query. That is, the provers respond with answers $a_1, \ldots, a_\ell$, where for every i we have $a_i = P_i(q_i)$. Finally, the verifier decides wether to accept or reject based on the answers that it receives (as well as the input x and its random string).

We say that $(V, P_1, \ldots, P_\ell)$ is a one-round multi-prover interactive proof system (MIP) for $\mathcal{L}$, with completeness $c \in [0,1]$ and soundness $s \in [0,1]$ (think of $s < c$) if the following two properties are satisfied:

1. **Completeness:** For every $x \in \mathcal{L}$, the verifier V accepts with probability c, over the random coins of V, $P_1, \ldots, P_\ell$, after interacting with $P_1, \ldots, P_\ell$, where c is a parameter referred to as the *completeness* of the proof system.

2. **Soundness:** For every $x \notin \mathcal{L}$, and any (computationally unbounded, possibly cheating) provers $P_1^*, \ldots, P_\ell^*$, the verifier V rejects with probability $\geq 1 - s$, over the random coins of V, after interacting with $P_1^*, \ldots, P_\ell^*$, where s is a parameter referred to as the *error* or *soundness* of the proof system.

Important parameters of an MIP are the number of provers, the length of queries, the length of answers, and the error. We say that the proof-system has *perfect completeness* If completeness hold with probability 1 (i.e. $c = 1$).

5.3.5 No-Signaling MIP

We will consider a variant of the MIP model, where the cheating provers are more powerful. In the MIP model, each prover answers its own query locally, without knowing the queries that were sent to the other provers. The no-signaling model allows each answer to depend on all the queries, as long as for any subset $S \subset [\ell]$, and any queries q_S for the provers in S, the distribution of the answers a_S, conditioned on the queries q_S, is independent of all the other queries.

Intuitively, this means that the answers a_S do not give the provers in S information about the queries of the provers outside S, except for information that they already have by seeing the queries q_S.

Formally, denote by D the alphabet of the queries and denote by Σ the alphabet of the answers. For every $q = (q_1, \ldots, q_\ell) \in D^\ell$, let $\mathcal{A}_q$ be a distribution over Σ^ℓ. We think of $\mathcal{A}_q$ as the distribution of the answers for queries q.

We say that the family of distributions $\{\mathcal{A}_q\}_{q \in D^\ell}$ is *no-signaling* if for every subset $S \subset [\ell]$ and every two sequences of queries $q, q' \in D^\ell$, such that $q_S = q'_S$, the following two random variables are identically distributed:

- a_S, where $a \in_R \mathcal{A}_q$
- a'_S where $a' \in_R \mathcal{A}_{q'}$

If the two distributions are δ-close, rather than identical, we say that the family of distributions $\{\mathcal{A}_q\}_{q \in D^\ell}$ is *δ-no-signaling*.

An MIP $(V, P_1, \ldots, P_\ell)$ for a language $\mathcal{L}$ is said to have soundness s against no-signaling strategies (or provers) if the following (more general) soundness property is satisfied:

2. **Soundness:** For every $x \notin \mathcal{L}$, and any no-signaling family of distributions $\{\mathcal{A}_q\}_{q \in D^\ell}$, the verifier V rejects with probability $\geq 1 - s$, where on queries $q = (q_1, \ldots, q_\ell)$ the answers are given by $(a_1, \ldots, a_\ell) \in_R \mathcal{A}_q$, and s is the soundness parameter.

If the property is satisfied for any δ-no-signaling family of distributions $\{\mathcal{A}_q\}_{q \in D^\ell}$, we say that the MIP has soundness s against δ-no-signaling strategies (or provers).

5.3.6 MIP of proximity (MIPP)

Let $\mathcal{L}$ be a language, let x be an input of length n (which we refer to as the main input) and let $\varepsilon = \varepsilon(n) \in (0,1)$ be a proximity parameter. In a one-round ℓ-prover interactive proof of proximity, ℓ computationally unbounded provers, $P_1, \ldots, P_\ell$, try to convince a (probabilistic) polynomial-time verifier, V, that the input x is ε-close (in relative Hamming distance) to some $x' \in \mathcal{L}$. The provers have free access to n, ε and x. The verifier has free access to n and ε and oracle access to x (and the number of oracle queries is counted).

We say that $(V, P_1, \ldots, P_\ell)$ is a one-round multi-prover interactive proof system of ε-proximity (ε-MIPP) for $\mathcal{L}$, with completeness $c \in [0,1]$ and soundness $s \in [0,1]$, if the following properties are satisfied:

1. **Running Time:** The verifier runs in polynomial time, i.e., time polynomial in the communication complexity and the number of oracle queries.

2. **Completeness:** For every $x \in \mathcal{L}$ the verifier V accepts with probability c, after interacting with $P_1, \ldots, P_\ell$.

3. **Soundness:** For every x that is ε-far from $\mathcal{L}$, and any (computationally unbounded, possibly cheating) provers $P_1^*, \ldots, P_\ell^*$, the verifier V rejects with probability $\geq 1-s$, after interacting with $P_1^*, \ldots, P_\ell^*$.

We denote such a proof system by ε-MIPP (and omit the soundness and completeness parameters from the notation). We say that the proof-system has *perfect completeness* if completeness hold with probability 1 (i.e. $c = 1$). The parameters we are mainly interested in are the query complexity and the communication complexity.

5.3.7 No-Signaling MIPP

An ε-MIPP, $(V, P_1, \ldots, P_\ell)$ for a language $\mathcal{L}$ is said to have soundness s against no-signaling strategies (or provers) if the following (more general) soundness property is satisfied:

2. **Soundness:** For every x that is ε-far from $\mathcal{L}$, and any no-signaling family of distributions $\{\mathcal{A}_q\}_{q \in D^\ell}$, the verifier V rejects with probability $\geq 1 - s$, where on queries $q = (q_1, \ldots, q_\ell)$ the answers are given by $(a_1, \ldots, a_\ell) \in_R \mathcal{A}_q$, and s is the error parameter.

If the property is satisfied for any δ-no-signaling family of distributions $\{\mathcal{A}_q\}_{q \in D^\ell}$, we say that the MIP has soundness s against δ-no-signaling strategies (or provers).

5.3.8 Low Degree Extension

Let $\mathbb{F}$ be a field and $H \subset \mathbb{F}$ a subset of the field. Fix an integer $m \in \mathbb{N}$. A basic fact is that for every function $\phi : H^m \to \mathbb{F}$, there exists a unique extension of ϕ into a function $\hat{\phi} : \mathbb{F}^m \to \mathbb{F}$ (which agrees with ϕ on H^m; i.e., $\hat{\phi}|_{H^m} \equiv \phi$), such that $\hat{\phi}$ is an m-variate polynomial of degree at most $|H|-1$ in each variable. Moreover, for every $x \in H^m$, there exists a unique m-variate polynomial $\hat{\beta}_x : \mathbb{F}^m \to \mathbb{F}$ of degree $|H|-1$ in each variable, such that for every function $\phi : H^m \to \mathbb{F}$ it holds that

$$\hat{\phi}(z_1, \ldots, z_m) = \sum_{x \in H^m} \hat{\beta}_x(z_1, \ldots, z_m) \cdot \phi(x).$$

The function $\hat{\phi}$ is called the *low degree extension* of ϕ (with respect to $\mathbb{F}, H, m$).

If $x \in \{0,1\}^n$ is a string such that $n = |H|^m$, then we denote by $\mathsf{LDE}_x : \mathbb{F}^m \to \mathbb{F}$ the low degree extension of x when viewed as a function, namely, the function $\mathsf{LDE}_x = \hat{\phi}_x$, where $\phi_x : [n] \to \{0,1\}$ is defined as $\phi_x(i) = x_i$.

5.3.9 Public-Key Encryption and Fully Homomorphic Encryption (FHE)

A *public-key encryption* scheme consists of three probabilistic polynomial-time algorithms $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$. The key generation algorithm Gen, when given as input a security parameter 1^τ, outputs a pair $(\mathsf{pk}, \mathsf{sk})$ of public and secret keys. The encryption algorithm, Enc, on input a public key pk and a message $m \in \{0,1\}^{\mathsf{poly}(\tau)}$, outputs a ciphertext $\hat{m}$, and the decryption algorithm, Dec, when given the ciphertext $\hat{m}$ and the secret key sk, outputs the original message m (with overwhelming probability). We allow the decryption process to fail with negligible probability (over the randomness of all algorithms).

Let $T : \mathbb{N} \to \mathbb{N}$ and $\delta : \mathbb{N} \to [0,1]$ be parameters. A public-key encryption scheme has security (S, δ) if for every family of circuits $\{C_\tau\}_{\tau \in \mathbb{N}}$ of size $\mathsf{poly}(T(\tau))$, for all sufficiently large τ and for any two messages $m, m' \in \{0,1\}^{\mathsf{poly}(\tau)}$ such that $|m| = |m'|$,

$$\left| \Pr_{(\mathsf{pk},\mathsf{sk}) \in_R \mathsf{Gen}(1^\tau)} [C_\tau(\mathsf{pk}, \mathsf{Enc}_{\mathsf{pk}}(m)) = 1] - \Pr_{(\mathsf{pk},\mathsf{sk}) \in_R \mathsf{Gen}(1^\tau)} [C_\tau(\mathsf{pk}, \mathsf{Enc}_{\mathsf{pk}}(m')) = 1] \right| < \delta(\tau)$$

where the probability is also over the random coin tosses of Enc.

Fully homomorphic encryption. The tuple $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval})$ is a *fully-homomorphic encryption scheme* if (1) $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ is a public-key encryption scheme, and (2) for every key-pair (pk, sk), the probabilistic polynomial-time algorithm Eval, on input the public-key pk, a circuit $C : \{0,1\}^k \to \{0,1\}^\ell$, where $k, \ell \leq \mathsf{poly}(\tau)$ (and τ is the security parameter), and a ciphertext $\hat{m}$ that is an encryption of a message $m \in \{0,1\}^k$ with respect to pk, outputs a string ψ such that the following two conditions hold:

- **Homomorphic Evaluation:** $\mathsf{Dec}_{\mathsf{sk}}(\psi) = C(m)$, except with negligible probability (over the coins of all algorithms).

- **Compactness:** The length of ψ is polynomial in τ, k and ℓ (and is independent of the size of C).

5.4 Lower Bound for IPP and No-Signaling MIPP

In this section we prove a lower bound, showing that there does not exist a no-signaling MIPP for all of P with query complexity q and communication complexity c such that $q \cdot c = o(n)$ (where n is the input length). More specifically, for every q we construct a language $\mathcal{L}$ in P and prove that if exponentially hard pseudo-random generators exist then for any no-signaling ε-MIPP for $\mathcal{L}$ with query complexity q and communication complexity c, it must be the case that $q \cdot c = \Omega(n)$. Later, in Section 5.4.2, we show a similar result for IPP.

In what follows we denote by τ the security parameter.

Definition 5.1. *A pseudo-random generator $G : \{0,1\}^n \to \{0,1\}^{\ell(n)}$ (with stretch $\ell(n) > n$) is said to be* exponentially hard *if for every circuit family $\{\mathcal{A}_\tau\}_\tau$ of size $2^{o(\tau)}$,*

$$\left| \Pr_{s \in_R \{0,1\}^\tau} [\mathcal{A}_\tau(1^\tau, G(s)) = 1] - \Pr_{y \in_R \{0,1\}^{\ell(\tau)}} [\mathcal{A}_\tau(1^\tau, y) = 1] \right| = \mathrm{negl}(\tau).$$

Theorem 5.7. *Assume the existence of exponentially hard pseudo-random generators. There exists a constant $\varepsilon > 0$ such that for every $q = q(n) \leq n$, there exists a language $\mathcal{L} \in$ P such that every* MIPP *for testing ε-proximity to $\mathcal{L}$ with completeness $2/3$, soundness $1/3$, query complexity q and communication complexity c it holds that $q \cdot c = \Omega(n)$.*

Remark 1. The above theorem holds with respect to any constant completeness parameter $c > 0$ and constant soundness parameter s such that $s < c$, and we chose $c = 2/3$ and $s = 1/3$ only for the sake of concreteness.

Remark 2. We note that the assumption in Theorem 5.7 can be reduced to sub-exponentially hard pseudo-random generators (i.e., it is infeasible for circuits of size 2^{τ^δ} to distinguish the output of the generator from uniform, for some $\delta > 0$), rather than exponential hardness, at the cost of a weaker implication (i.e., $q \cdot c = \Omega(n^\delta)$).

We refer the reader to Section 5.2 for the high-level overview of the proof.

5.4.1 Proof of Theorem 5.7

We start by defining the notion of average-case no-signaling MIP (in the crs model), which is used in the proof of Theorem 5.7. We note that this average-case completeness seems too weak for applications and we define this weak notion only for the sake of the proof of Theorem 5.7.

Definition 5.2. *An* average-case *no-signaling* MIP *in the* common random string *(*crs*) model, for a language $\mathcal{L}$, with completeness c and soundness s, consists of $(V, P_1, \ldots, P_\ell, \mathsf{crs})$, where as before V is the verifier, $P_1, \ldots, P_\ell$ are the provers, and* crs *is a* common random string *of length* $\mathsf{poly}(n)$*, chosen uniformly at random and given to all parties. In particular, V's queries and decision may depend on the* crs*, and the answers generated by both honest and cheating provers may depend on the* crs*. The following completeness and soundness conditions are required:*

- **Average-case completeness.** *For all sufficiently large $n \in \mathbb{N}$,*

$$\Pr\left[(V, P_1, \ldots, P_\ell)(x, \mathsf{crs}) = 1\right] \geq c,$$

  where the probability is over uniformly distributed *$x \in_R \mathcal{L} \cap \{0,1\}^n$, over uniformly generated $\mathsf{crs} \in_R \{0,1\}^{\mathsf{poly}(n)}$, and over the random coin tosses of the verifier V.*

- **Soundness against no-signaling provers.** *For every $x \notin \mathcal{L}$, and every family of distributions $\{\mathcal{A}_{q,\mathsf{crs}}\}_{q \in D^\ell, \mathsf{crs} \in \{0,1\}^{\mathsf{poly}(n)}}$ such that for every $\mathsf{crs} \in \{0,1\}^{\mathsf{poly}(n)}$ the family of distributions $\{A_{q,\mathsf{crs}}\}_{q \in D^\ell}$ is no-signaling, the verifier V rejects with probability $\geq 1 - s$, where the answers corresponding to (q, crs) are given by $(a_1, \ldots, a_\ell) \in_R \mathcal{A}_{q,\mathsf{crs}}$.*

The following claim, which we use in the proof of Theorem 5.7, follows from [DLN+04] (see also [Ito10]).

Claim 5.2.1. *Suppose that a language $\mathcal{L}$ has an average-case no-signaling* MIP *in the* crs *model, communication complexity $c = c(n)$ (where n is the instance length), and with constant completeness and soundness (where the soundness parameter is smaller than the completeness parameter). Then, there exists a randomized algorithm D that runs in time $\mathsf{poly}(n, 2^c)$ such that:*

- *For every $n \in \mathbb{N}$,*

$$\Pr_{x \in_R \mathcal{L} \cap \{0,1\}^n}[D(x) = 1] \geq 2/3$$

  where the probability is also over the coin tosses of D.

- *For every $x \notin \mathcal{L}$ it holds that*

$$\Pr[D(x) = 1] \leq 1/3$$

  where the probability is over the coins tosses of D.

We note that [DLN+04, Ito10] did not consider the crs model nor average-case completeness, but the claim extends readily to this setting as well.

We are now ready to prove Theorem 5.7.

Proof of Theorem 5.7. Assume that there exists a pseudo-random generator (PRG), denoted by $G : \{0,1\}^\tau \to \{0,1\}^{2\tau}$, that is exponentially secure. Namely, every adversary of size $2^{o(\tau)}$ cannot distinguish between uniformly distributed $r \in_R \{0,1\}^{2\tau}$ and $G(s)$ for uniformly distributed $s \in_R \{0,1\}^\tau$, with non-negligible advantage. For sake of simplicity, we assume that G is injective[6].

Let $\varepsilon > 0$ be a constant for which there exists a (good) error-correcting-code, denoted by ECC, with constant rate and efficient encoding that is resilient to (2ε)-fraction of adversarially chosen errors.

Fix any query complexity $q = o(n)$.[7] We show that there exists a language $\mathcal{L} \in \mathsf{P}$ such that for every no-signaling ε-MIPP for $\mathcal{L}$ with query complexity q and communication complexity c (and completeness $\frac{2}{3}$ and soundness $\frac{1}{3}$) it must be the case that $q \cdot c = \Omega(n)$.

Consider the following language:

$$\mathcal{L} = \left\{(\mathsf{ECC}(r_1, \ldots, r_m), s_1, \ldots, s_m) : \ \forall i \in [m], G(s_i) = r_i\right\},$$

where $m = 4q$ and $\tau = |s_i| = \Theta(n/q)$, where $n = |(\mathsf{ECC}(r_1, \ldots, r_m), s_1, \ldots, s_m)|$. The fact that $|s_i| = \Theta(n/q)$ follows from the fact that ECC has constant rate (i.e., $|\mathsf{ECC}(z)| = O(|z|)$).

The fact that ECC is efficiently decodable and G is efficiently computable implies that $\mathcal{L} \in \mathsf{P}$. Suppose for contradiction that there exists a no-signaling ε-MIPP for $\mathcal{L}$, denoted by $(V, P_1, \ldots, P_\ell)$, with communication complexity c such that $c = o(n/q)$.

Consider the following NP language

$$\mathcal{L}_G = \{r : \exists s \text{ s.t. } G(s) = r\}.$$

Claim 5.2.1, together with the fact that G is exponentially secure, implies that $\mathcal{L}_G$ does not have an average-case MIP in the crs model with soundness against no-signaling strategies, with communication complexity $o(\tau)$ for instances of length τ.

We obtain a contradiction by constructing an average-case MIP in the crs model with soundness against no-signaling strategies, with communication complexity $o(\tau)$. To this end, consider the following MIP in the crs model for $\mathcal{L}_G$, denoted by $(V', P'_1, \ldots, P'_\ell, \mathsf{crs})$.

- The crs consists of m uniformly distributed seeds $s_1, \ldots, s_m \in_R \{0,1\}^\tau$, and a random coordinate $i \in_R [m]$.
- The verifier V', on input $r \in \{0,1\}^{2\tau}$, does the following:
  1. Let $r_i = r$, and for every $j \in [m] \setminus \{i\}$, let $r_j = G(s_j)$.
  2. Emulate V with oracle access to $(\mathsf{ECC}(r_1, \ldots, r_m), s_1, \ldots, s_m)$.
     (Note that with overwhelming probability $r \neq G(s_i)$, and thus $r_i \neq G(s_i)$. However V will not notice this unless it queries coordinates that belong to s_i.)

[6] We note that this assumption can be easily removed by replacing the use of the uniform distribution over the language $\mathcal{L}'$ (defined below) with the distribution $G(s)$ for $s \in_R \{0,1\}^\tau$.

[7] Note that for $q = \Omega(n)$ the theorem is trivially true.

- The provers $P'_1, \ldots, P'_\ell$, emulate $P_1, \ldots, P_\ell$ on input $(\mathsf{ECC}(r_1, \ldots, r_m), s_1, \ldots, s_m)$, while setting $r_i = r$ and setting $s_i = s$ where $r = G(s)$ (assuming that such s exists).[8] If such s does not exist then the provers $P'_1, \ldots, P'_\ell$ send a reject message, and abort.

Note that the communication complexity of $(V', P'_1, \ldots, P'_\ell, \mathsf{crs})$ is equal to the communication complexity of $(V, P_1, \ldots, P_\ell, \mathsf{crs})$, denoted by c. By our assumption, $c = o(n/q) = o(\tau)$, as desired.

We proceed to prove that $(V', P'_1, \ldots, P'_\ell, \mathsf{crs})$ has average-case completeness $\frac{1}{2}$ and soundness $\frac{1}{3}$.

Average-case completeness. We need to prove that

$$\Pr[(V', P'_1, \ldots, P'_\ell)(r, \mathsf{crs}) = 1] \geq \frac{1}{2},$$

where the probability is over *uniformly distributed* $r \in_R (\mathcal{L}_G)_\tau$, over uniformly generated $\mathsf{crs} = (s_1, \ldots, s_m, i)$ where each $s_j \in_R \{0,1\}^\tau$, $i \in_R [m]$, and over the random coin tosses of the verifier V.

Let GOOD denote the event that V' does not query any of the coordinates that belong to s_i, where $i \in [m]$ is the random coordinate chosen by V'. Notice that for every $r \in \mathcal{L}_G$,

$$\Pr\left[(V', P'_1, \ldots, P'_\ell)(r, \mathsf{crs}) = 1 \mid \mathsf{GOOD}\right] =$$
$$\Pr\left[(V, P_1, \ldots, P_\ell)(\mathsf{ECC}(r_1, \ldots, r_m), s_1, \ldots, s_m) = 1 \mid s_i \text{ is not queried}\right] \geq \frac{2}{3}$$

where the probabilities are over a uniformly distributed crs and the random coin tosses of V' and V, and where in the second equation $r_i = r$ and $s_i = s$, where $r = G(s)$. Recall that the fact that $r \in \mathcal{L}_G$ implies that such s exists.

The fact that

$$\Pr[(V', P'_1, \ldots, P'_\ell)(r, \mathsf{crs}) = 1] \geq \Pr[(V', P'_1, \ldots, P'_\ell)(r, \mathsf{crs}) = 1 |\ \mathsf{GOOD}] \cdot \Pr[\mathsf{GOOD}]$$

implies that it suffices to prove that

$$\Pr[\mathsf{GOOD}] \geq \frac{3}{4}, \tag{5.1}$$

where the probability is over uniformly distributed $r \in_R \mathcal{L}_G$, uniformly distributed crs, and over the random coin tosses of V'.

Note that $r_1, \ldots, r_m$ are all distributed identically to r, and thus $V, P_1, \ldots, P_\ell$, which all receive as input $(\mathsf{ECC}(r_1, \ldots, r_m), s_1, \ldots, s_m)$, where $r_i = r$, do not have any advantage in guessing i (here we crucially use the fact that the MIPP provers are not given access to the crs). Therefore, since V makes at most q queries, and since $m = 4q$, it follows from the union bound that V queries any location of s_i with probability at most $\frac{q}{m} = \frac{1}{4}$, thereby establishing Eq. (5.1).

[8] This step can be done by a brute force search (since the honest provers are also computationally unbounded). Nevertheless, we note that typically in proof-systems for language in NP the prover is given the NP witness and so this step can also be done efficiently.

Soundness against No-Signaling Strategies. We prove that for every $r \notin \mathcal{L}_G$, every $\mathsf{crs} = (s_1, \ldots, s_m, i)$, and every no-signaling cheating strategy $P^{\mathsf{NS}} = (P_1^*, \ldots, P_\ell^*)$,

$$\Pr[(V', P^{\mathsf{NS}})(r, \mathsf{crs}) = 1] \leq \frac{1}{3},$$

where the probability is over the random coin tosses of V' and P^{NS}.

To this end, fix any $r \notin \mathcal{L}_G$ and any $\mathsf{crs} = (s_1, \ldots, s_m, i)$ where each $s_j \in \{0,1\}^\tau$ and $i \in [m]$. Suppose for the sake of contradiction that there exists a no-signaling cheating strategy $P^{\mathsf{NS}} = (P_1^*, \ldots, P_\ell^*)$ such that

$$\Pr[(V', P^{\mathsf{NS}})(r, \mathsf{crs}) = 1] > \frac{1}{3},$$

where the probability is over the random coin tosses of V' and P^{NS}.

Recall that V' runs V on input $(\mathsf{ECC}(r_1, \ldots, r_m), s_1, \ldots, s_m)$, where $r_i = r$ and where $r_j = G(s_j)$ for every $j \in [m] \setminus \{i\}$. We prove that there exists a no-signaling cheating strategy, denoted by $\hat{P}^{\mathsf{NS}}$, such that

$$\Pr[(V, \hat{P}^{\mathsf{NS}})(\mathsf{ECC}(r_1, \ldots, r_m), s_1, \ldots, s_m) = 1] > \frac{1}{3}, \tag{5.2}$$

where the probability is over the random coin tosses of V and $\hat{P}^{\mathsf{NS}}$.

The cheating strategy $\hat{P}^{\mathsf{NS}}$ simply emulates P^{NS}. Namely, $\hat{P}^{\mathsf{NS}}$, upon receiving queries $(q_1, \ldots, q_\ell)$, will emulate $P^{\mathsf{NS}}(r, \mathsf{crs})$ upon receiving $(q_1, \ldots, q_\ell)$, where $r = r_i$ and $\mathsf{crs} = (s_1, \ldots, s_m, i)$. Note that $\hat{P}^{\mathsf{NS}}$ simulates P^{NS} perfectly, and therefore indeed Equation (5.2) holds. Also note that the fact that P^{NS} is a no-signaling strategy immediately implies that $\hat{P}^{\mathsf{NS}}$ is also a no-signaling strategy.

To get a contradiction, it thus remains to show that $(\mathsf{ECC}(r_1, \ldots, r_m), s_1, \ldots, s_m)$ is ε-far from $\mathcal{L}$. Indeed, the fact that ECC is an error correcting code resilient to 2ε-fraction of adversarial errors, together with the fact that $r \notin \mathcal{L}_G$ implies that $(\mathsf{ECC}(r_1, \ldots, r_m), s_1, \ldots, s_m)$ is ε-far from $\mathcal{L}$, as desired.

□

5.4.2 Lower Bound for IPP

In this section we show how to extend Theorem 5.7 to the setting of IPP. As is Theorem 5.7, our lower bound for IPP is based on a cryptographic assumption. Specifically, we assume the existence of an exponentially secure PRG that is computable in NC_1 (defined next).

An exponentially secure PRG in NC_1, is a function $G : \{0,1\}^\tau \to \{0,1\}^{2\tau}$ such that:

1. G is computable by an NC_1 circuit.
2. Every family of circuits of size $2^{o(\tau)}$ cannot distinguish between uniformly distributed $r \in_R \{0,1\}^{2\tau}$ and $G(s)$ for uniformly distributed $s \in_R \{0,1\}^\tau$, with non-negligible advantage.

We note that there are several candidate PRGs in NC_1 (based on assumptions related to factoring, discrete log and lattices, see [AIK06] for details) and that for some of these PRGs there are no known attacks that take time $2^{o(\tau)}$ (and for the other candidates there are no known attacks that take time 2^{n^δ} for some $\delta > 0$, see remark following the statement of Theorem 5.7).

Theorem 5.8. *Assume the existence of an exponentially secure* PRG *in* NC_1. *There exists a constant* $\varepsilon > 0$ *such that for every* $q = q(n) \leq n$, *there exists a language* $\mathcal{L} \in \mathsf{NC}_1$ *such that every* ε-IPP *for* $\mathcal{L}$ *with completeness* $2/3$, *soundness* $1/3$, *query complexity* q *and communication complexity* c *it holds that* $q \cdot c = \Omega(n)$.

The proof of Theorem 5.8 is very similar to the proof of Theorem 5.7 and so we provide only a sketch.

Proof Sketch. Analogously to the proof of Theorem 5.7, we consider the language

$$\mathcal{L} = \left\{(\mathsf{ECC}(r_1, \ldots, r_m), s_1, \ldots, s_m) : \ \forall i \in [m], G(s_i) = r_i\right\},$$

where $G : \{0,1\}^\tau \to \{0,1\}^{2\tau}$ is an exponentially secure PRG and ECC is a (good) efficiently computable error-correcting code with constant rate that is resilient to 2ε-fraction of adversarial error. In addition, we require that both G and ECC are computable in NC_1,[9] and thus $\mathcal{L} \in \mathsf{NC}_1$. We also consider the NP language

$$\mathcal{L}_G = \{r : \exists s \text{ s.t. } G(s) = r\},$$

As in the proof of Theorem 5.7, we assume for contradiction that there exists an ε-IPP for $\mathcal{L}$ with $q \cdot c = o(n)$.[10]

Analogously to the proof of Theorem 5.7, we show that such an ε-IPP for $\mathcal{L}$ implies an interactive proof for $\mathcal{L}_G$ with average-case completeness (i.e., completeness is only guaranteed for random $x \in \mathcal{L}_G$), communication complexity c, and where both verifier and prover have access to a common random string (crs).

The key step in the proof is to show (analogously to Claim 5.2.1) that such an average-case interactive proof (in the crs model) implies a randomized algorithm that runs in time $\mathsf{poly}(2^c, n)$ and with high probability accepts $x \in_R \mathcal{L}_G$ and rejects $x \notin \mathcal{L}_G$.

The latter randomized algorithm is constructed as in the proof that IP $\subseteq$ PSPACE. Namely, we show a randomized prover strategy P^* that (with high probability, almost) maximizes the verifier's acceptance probability for every x (both $x \in \mathcal{L}$ and $x \notin \mathcal{L}$) and can be computed in time $\mathsf{poly}(n, 2^c)$. The prover P^* is defined as follows. To decide its next message at every round of the interaction, P^* estimates for every $\alpha \in \{0,1\}^{\leq c}$, the acceptance probability of the verifier if it sends α as its response, conditioned on the

[9] Such an error correcting code was constructed, for example, by Spielman [Spi96].

[10] We note that if the ε-IPP for $\mathcal{L}$ is a public-coin protocol, then the proof follows from Theorem 5.7 together with a general transformation from public-coin IPP to no-signaling MIPP, (which can be thought of as a special case of Lemma 5.10). In general, one could use the Goldwasser-Sipser [GS86] transformation (to a public-coin protocol) to obtain a version of Theorem 5.8 with slightly worse parameters.

transcript so far. It then sends α that maximizes the acceptance probability. We note that the above sampling can be done by rejection sampling. Since the entire transcript is of length at most c, with high probability it suffices for P^* to sample $\mathsf{poly}(n, 2^c)$ random strings.

The rest of the proof is similar to the proof of Theorem 5.7, and we obtain a contradiction by showing a $2^{o(\tau)}$ randomized algorithm that breaks the pseudorandomness of the PRG. □

5.4.3 Lower Bound for Interactive Arguments of Proximity

In this section we show a significant barrier for constructing arguments of proximity for P with query complexity q and communication complexity c such that $q \cdot c = o(n)$. We interpret this result as showing that, in a sense, Theorem 5.16 is (almost) tight. For this section, we assume that the reader is familiar with definition and results in [GW11].

Throughout this section we assume the existence of an exponentially hard pseudorandom generator $G : \{0,1\}^\tau \to \{0,1\}^{2\tau}$ (see Definition 5.1). As before, we consider the language $\mathcal{L}_G \stackrel{\text{def}}{=} \{r \in \{0,1\}^* : \exists s \text{ s.t. } G(s) = r\}$.

We show that if every language in P has a 1-round argument of proximity with query complexity q and communication complexity c such that $q \cdot c = o(n)$, then there exists a 1-round argument-system for $\mathcal{L}_G$ (albeit only with average-case completeness) where the prover-to-verifier message has length $o(n)$. Such an argument system is known as a SNARG (succinct non-interactive argument-system) in the literature.

Such a SNARG for $\mathcal{L}_G$ stands in contrast to a result of Gentry and Wichs [GW11], which shows that soundness of any SNARG for the language $\mathcal{L}_G$ cannot be shown by a black-box reduction to any falsifiable cryptographic assumption (unless the assumption itself is false). We refer the reader to [GW11] for the definition of black-box reduction and the definition of falsifiable assumptions (originally defined by [Nao03]).

We note that the [GW11] result is only known to hold if the SNARG satisfies a strong notion of soundness, called *adaptive* soundness. A SNARG has adaptive soundness if the verifier's message to the prover does not depend on the input x and the standard computational soundness condition is extended so that the cheating prover may choose $x \notin \mathcal{L}$, on which it wants to cheat, *after* seeing the verifier's message.

We first define what we mean by a falsifiable cryptographic assumption.

Definition 5.3 ([Nao03, GW11])**.** *A* falsifiable cryptographic assumption *consists of an efficient interactive challenger* $\mathbb{C}$ *and a constant* $c \in [0,1)$*. On input a security parameter* τ*, the challenger* $\mathbb{C}(1^\tau)$ *interacts with a machine* $\mathcal{A}(1^\tau)$ *and may output a special symbol* win*. If this occurs, we say that* $\mathcal{A}(1^\tau)$ *wins* $\mathbb{C}(1^\tau)$*. The assumption associated with the tuple* $(\mathbb{C}, c)$ *states that for any efficient* $\mathcal{A}$*, we have* $\Pr[\mathcal{A}(1^\tau) \text{ wins } \mathbb{C}(1^\tau)] \leq c + negl(\tau)$*, where the probability is over the random coins of* $\mathbb{C}$ *and* $\mathcal{A}$*.*

Let $\Pi = (P, V)$ be an adaptively sound argument-system. We say that a (possibly inefficient) algorithm P^* is a Π-adversary if P^* violates the adaptive soundness of Π (while ignoring the efficiency restriction).

Definition 5.4. *A* black-box reduction *showing the soundness of an adaptively sound 1-round argument* $\Pi = (P, V_1, V_2)$ *based on a falsifiable assumption* $(\mathbb{C}; c)$ *is an efficient oracle-machine* $R^{(\cdot)}$ *such that, for every (possibly inefficient)* Π*-adversary* P*, the machine* R^P *breaks the assumption.*

Using these two notions, we can now state the [GW11] result.

Theorem 5.9 (Informal [GW11])**.** *Assume that there exists an exponentially hard pseudo-random generator* $G : \{0,1\}^\tau \to \{0,1\}^{2\tau}$*. The adaptive soundness of any candidate SNARG for the language* $\mathcal{L}_G$ *cannot be shown by a black-box reduction to any falsifiable cryptographic assumption (unless the assumption itself is false). Furthermore, the theorem holds even if the SNARG only has average-case completeness.*

We note that the furthermore clause is not stated explicitly in [GW11], however, it is easy to verify that the result extends also to the case where the SNARG only satisfies average-case completeness.

We show the following:

Theorem 5.10 (Informal)**.** *Assume that there exists an exponentially hard pseudo-random generator* $G : \{0,1\}^\tau \to \{0,1\}^{2\tau}$*. There exists a constant* $\varepsilon > 0$ *such that the following holds. Suppose that every language in* P *has a 1-round argument of* ε*-proximity with adaptive soundness, query complexity* q *and communication complexity* c *such that* $q \cdot c = o(n)$*. Then, there exists a 1-round argument-system for the language* $\mathcal{L}_G$ *(albeit only with average-case completeness), where the prover-to-verifier message has length* $o(n)$*. Furthermore, the adaptive soundness of* $\mathcal{L}_G$ *is shown by a black-box reduction to the adaptive soundness of the argument of proximity for* P*.*

By combining Theorem 5.10 with Theorem 5.9, we obtain the following corollary.

Corollary 5.11 (Informal)**.** *Assume that there exists an exponentially hard pseudo-random generator. There exists a constant* $\varepsilon > 0$ *and a language* $\mathcal{L} \in \mathsf{P}$ *such that the adaptive soundness of any argument of* ε*-proximity for* $\mathcal{L}$ *with query complexity* q *and communication complexity* c *such that* $q \cdot c = o(n)$ *cannot be proven by a black-box reduction to any falsifiable cryptographic assumption (unless the assumption itself is false).*

Proof Sketch of Theorem 5.10. The proof uses similar ideas to those in the proof of Theorem 5.7 (and Theorem 5.8).

Assume that there exists a pseudo-random generator (PRG), denoted by $G : \{0,1\}^\tau \to \{0,1\}^{2\tau}$, that is exponentially secure (see Definition 5.1). For sake of simplicity, we assume that G is injective (see Footnote 6).

Let ECC be an error-correcting code as in the proof of Theorem 5.7. Namely, ECC has constant rate, efficient encoding and is resilient to (2ε)-fraction of adversarially chosen errors, for some constant $\varepsilon > 0$.

Fix any query complexity $q = q(n)$ and consider the following language:

$$\mathcal{L} = \left\{(\mathsf{ECC}(r_1, \ldots, r_m), s_1, \ldots, s_m) : \ \forall i \in [m], G(s_i) = r_i\right\},$$

where $m = 4q$ and $\tau = |s_i| = \Theta(n/q)$, where $n = |(\mathsf{ECC}(r_1, \ldots, r_m), s_1, \ldots, s_m)|$. The fact that $|s_i| = \Theta(n/q)$ follows from the fact that ECC has constant rate (i.e., $|\mathsf{ECC}(z)| = O(|z|)$).

The fact that ECC is efficiently encodable and G is efficiently computable implies that $\mathcal{L} \in \mathsf{P}$. Assume that $\mathcal{L}$ has an argument of ε-proximity, denoted by (V, P), with query complexity q and communication complexity c such that $c = o(n/q)$.

In a similar manner to the proof of Theorem 5.7, we show that the language $\mathcal{L}_G$ has an (average-case) SNARG, in the crs model with communication complexity $o(\tau)$. Note that this implies a SNARG *without a* crs where the prover-to-verifier message is of length $o(\tau)$, since the crs can be made part of the verifier-to-prover message.

Consider the following SNARG for $\mathcal{L}_G$, denoted by (V', P').

- The crs consists of m uniformly distributed seeds $s_1, \ldots, s_m \in_R \{0,1\}^\tau$, and a random coordinate $i \in_R [m]$.
- The verifier V', on input $r \in \{0,1\}^{2\tau}$, does the following:
  1. Let $r_i = r$, and for every $j \in [m] \setminus \{i\}$, let $r_j = G(s_j)$.
  2. Emulate V with oracle access to $(\mathsf{ECC}(r_1, \ldots, r_m), s_1, \ldots, s_m)$.
     (Note that with overwhelming probability $r \neq G(s_i)$, and thus $r_i \neq G(s_i)$. However V will not notice this unless it queries coordinates that belong to s_i.)
- The prover P', which in addition to r and the crs gets access to s, the NP-witness corresponding to r (i.e., $r = G(s)$) emulates the prover P on input $(\mathsf{ECC}(r_1, \ldots, r_m), s_1, \ldots, s_m)$, while setting $r_i = r$ and setting $s_i = s$ where $r = G(s)$.

Note that the communication complexity of (V', P') is equal to the communication complexity of (V, P), denoted by c. By our assumption, $c = o(n/q) = o(\tau)$, as desired.

The proof that (V', P') has average-case completeness $\frac{1}{2}$ and adaptive soundness $\frac{1}{3}$ is analogous to the proof in Theorem 5.7 and so we omit it here. Moreover, it is easy to see that the reduction above is a black-box reduction.

□

5.5 No-signaling MIPP for P

In this section we construct an ε-MIPP for every language in P. Later, in Section 5.6, we show how to transform such an MIPP into a one-round argument of ε-proximity.

Theorem 5.12. *Let $\mathcal{L} \in \mathsf{DTIME}(t(n))$, where $t = t(n)$ satisfies $\mathsf{poly}(n) \leq t \leq \mathsf{exp}(n)$, let k be an integer such that $(\log t)^c \leq k \leq \mathsf{poly}(n)$, where c is some (sufficiently large) universal constant, and let $\varepsilon = \varepsilon(n) \in (0,1)$.*

Then, $\mathcal{L}$ has an ε-MIPP with ℓ provers, where $\ell = (k+\varepsilon n) \cdot \mathsf{polylog}(t)$, perfect completeness and soundness 2^{-k} against δ-no-signaling strategies, where $\delta = 2^{-(k+\varepsilon n)\cdot \mathsf{polylog}(t)}$.

The resulting MIPP has queries and answers (to the provers) of length $(k+\varepsilon n) \cdot \mathsf{polylog}(t) + k \cdot \varepsilon n \cdot (1/\varepsilon)^{o(1)}$. The verifier runs in time $(k+\varepsilon n)^2 \cdot \mathsf{polylog}(t) + k \cdot (1/\varepsilon + \varepsilon n)^{1+o(1)}$ and has query complexity (to the main input) $k \cdot (1/\varepsilon)^{1+o(1)}$. Each of the (honest) provers runs in time $\mathsf{poly}(t,k)$.

Note that the communication complexity of our ε-MIPP is roughly $(\varepsilon n)^2$ and the query complexity is roughly $1/\varepsilon$. Thus, Theorem 5.12 is interesting for $\varepsilon \leq \frac{1}{n^{1/2+\alpha}}$ (for any constant $\alpha > 0$), as it gives an ε-MIPP where the verifier runs in sub-linear time.

In Section 5.4 we show a lower bound, where we prove that for any ε-MIPP (even for constant $\varepsilon > 0$) with query complexity q and communication complexity c, it must be the case that $q \cdot c = \Omega(n)$. Notice that our upper bound in Theorem 5.12 does not quite match the lower bound, since in our upper bound there is a quadratic blowup in communication complexity. If our ε-MIPP above did not have the quadratic blowup in communication complexity, and the communication complexity was roughly εn (or more precisely, was $(k+\varepsilon n) \cdot \mathsf{polylog}(t) \cdot n^{o(1)}$) then our lower bound and upper bound would match.

We note that the reason for the quadratic blowup in communication is inherited from [KRR13b], and we are hopeful that this blowup is not inherent and can be removed, in which case the ε-MIPP in Theorem 5.12 would match our lower bound in Section 5.4 (up to polylogarithmic factors).

The proof of Theorem 5.12 follows the outline of the [RVW13] IPP protocol for languages in (logspace-uniform) NC. We first show that (in a sense) the language PVAL (defined next) is complete for constructing ε-MIPPs for P. Then, using the fact (shown by [RVW13]) that PVAL has an ε-IPP, we construct an ε-MIPP for every language in P by showing a composition lemma between no-signaling MIPPs and IPPs.

5.5.1 Completeness of PVAL

Throughout this section we think of n (which is some sufficiently large integer) as the input size. Let $\mathcal{L} \in \mathsf{DTIME}(t(n))$, where $t = t(n)$ satisfies $\mathsf{poly}(n) \leq t \leq \mathsf{exp}(n)$, let k be an integer such that $(\log t)^c \leq k \leq \mathsf{poly}(n)$, where c is some (sufficiently large) universal constant, and let $\varepsilon = \varepsilon(n) \in (0,1)$.

The following notations and constants are taken (almost verbatim) from [KRR13b]. Let $N = \mathsf{poly}(t)$ be the size of the "augmented" circuit for computing $\mathcal{L}$ as defined in [KRR13b, Section 9] (the actual details of this augmented circuit are not important). Let $H = \{0,1,\ldots,\log N - 1\}$ and let[11] $m = \frac{\log n}{\log \log N}$, so that $n = |H|^m$. (For simplicity

[11] In contrast, in [KRR13b] the value of m is $\frac{\log N}{\log \log N}$. The reason that we take m to be smaller is that

and without loss of generality we assume that $\log N$ and $\frac{\log n}{\log \log N}$ are integers). Let $\mathbb{F}$ be a field, such that $4|H|^{10} \leq |\mathbb{F}| \leq 8(\log N)^{10}$.

Recall that if $x \in \{0,1\}^n$, we denote by $\mathsf{LDE}_x : \mathbb{F}^m \to \mathbb{F}$ the low degree extension of x with respect to $\mathbb{F}$, H and m (see Section 5.3.8).

Definition 5.5. *Let $J \in (\mathbb{F}^m)^\ell$ be a sequence of ℓ points in $\mathbb{F}^m$ and let $\vec{v} \in F^\ell$ be a sequence of ℓ corresponding values. The language $\mathsf{PVAL}_{J,\vec{v}} \subseteq \{0,1\}^n$ is defined as:*

$$\mathsf{PVAL}_{J,\vec{v}} \stackrel{\text{def}}{=} \{x : H^m \to \{0,1\} \mid s.t.\ \mathsf{LDE}_x(J) = \vec{v}\}\,.$$

Remark. The language PVAL obviously also depends on the choice of $\mathbb{F}$, H and m (as defined above) but for sake of brevity we omit them from the notation.

Theorem 5.13 (No-Signaling MIP for Deterministic Languages [KRR13b])**.** *There exists a 1-round protocol for $\mathcal{L} \in \mathsf{DTIME}(t(n))$ between $k \cdot \mathsf{polylog}(t)$ provers and a verifier, where the provers get as input $x \in \{0,1\}^n$, and the verifier only gets the input n. The (honest) provers are allowed to communicate only with the verifier and not with each other. The output of the protocol is a sequence of $k \cdot \mathsf{polylog}(t)$ coordinates $J \in (\mathbb{F}^m)^{k \cdot \mathsf{polylog}(t)}$ and a sequence of values $\vec{v} \in \mathbb{F}^{k \cdot \mathsf{polylog}(t)}$, such that:*

- ***Completeness.*** *If $x \in \mathcal{L}$ and the provers honestly follows the protocol, then $\mathsf{LDE}_x(J) = \vec{v}$ (with probability 1).*
- ***Soundness.*** *If $x \notin \mathcal{L}$, then for any $2^{-k \cdot \mathsf{polylog}(t)}$-no-signaling cheating family of distribution $\mathcal{A}$, with probability at least $1 - 2^{-k}$ (over the verifier's coins and a random sample from $\mathcal{A}$), it holds that $\mathsf{LDE}_x(J) \neq \vec{v}$.*

The verifier runs in time $k^2 \cdot \mathsf{polylog}(t)$ (without accessing x) and the provers run in time $\mathsf{poly}(t,k)$. Each query and answer (to the provers) is of length $k \cdot \mathsf{polylog}(t)$.

Theorem 5.14 (PVAL is Complete for No-Signaling MIPP)**.** *There exists a 1-round protocol for $\mathcal{L} \in \mathsf{DTIME}(t(n))$ between $(k + \varepsilon n) \cdot \mathsf{polylog}(t)$ provers and a verifier, where the provers get as input $x \in \{0,1\}^n$, and the verifier only gets the input n. The (honest) provers are allowed to communicate only with the verifier and not with each other. The output of the protocol is a sequence of $(k+\varepsilon n) \cdot \mathsf{polylog}(t)$ coordinates $J \in (\mathbb{F}^m)^{(k+\varepsilon n) \cdot \mathsf{polylog}(t)}$ and a sequence of values $\vec{v} \in \mathbb{F}^{(k+\varepsilon n) \cdot \mathsf{polylog}(t)}$, such that:*

- ***Completeness.*** *If $x \in \mathcal{L}$ and the provers honestly follows the protocol, then $x \in \mathsf{PVAL}_{J,\vec{v}}$ (with probability 1).*
- ***Soundness.*** *If x is ε-far from $\mathcal{L}$, then for any $2^{-(k+\varepsilon n) \cdot \mathsf{polylog}(t)}$-no-signaling cheating family of distributions $\mathcal{A}$, with probability at least $1 - 2^{-k}$ over the verifier's and provers' coins, it holds that x is ε-far from $\mathsf{PVAL}_{J,\vec{v}}$.*

we will only be interested in the low degree extension of the input $x \in \{0,1\}^n$ rather then the entire computation.

The verifier runs in time $(k+\varepsilon n)^2 \cdot \mathsf{polylog}(t)$ *(without accessing* x*) and the provers run in time* $\mathsf{poly}(t,k)$*. Each query and answer (to the provers) is of length* $(k+\varepsilon n)\cdot\mathsf{polylog}(t)$*.*

Proof. We use the 1-round protocol of Theorem 5.13 with respect to the parameter $k' = k + \varepsilon n \log n$. To show that completeness holds, note that if $x \in \mathcal{L}$, then by Theorem 5.13, it holds that $\mathsf{LDE}_x(J) = \vec{v}$ (with probability 1) and therefore $x \in \mathsf{PVAL}_{J,\vec{v}}$ (with probability 1).

For soundness, let x be ε-far from $\mathcal{L}$ and let $\mathcal{A}$ be a $2^{-(k+\varepsilon n \log n)\cdot\mathsf{polylog}(t)}$-no-signaling (cheating) family of distributions. For every x' that is ε-close to x it holds that $x' \notin \mathcal{L}$ (since x is ε-far from $\mathcal{L}$) and therefore, by Theorem 5.13, with probability $1 - 2^{-k'}$ it holds that $\mathsf{LDE}_{x'}(J) \neq \vec{v}$. Since there are $\binom{n}{\varepsilon n} \leq 2^{\varepsilon n \log n}$ different x' that are ε-close to x, by taking a union bound over all such x', the probability that there exists some x' that is ε-close to x such that $\mathsf{LDE}_{x'}(J) = \vec{v}$, is at most $2^{-k'} \cdot 2^{\varepsilon n \log n} = 2^{-k}$. □

5.5.2 Composing an MIPP with an IPP

In this section we construct an MIPP that is sounds against δ-no-signaling strategies for every language in P. The main step is composing the reduction of Theorem 5.14 with an IPP for PVAL. Such an IPP was constructed by [RVW13]. We first state the [RVW13] result for PVAL and then show how it can be combined with Theorem 5.14 to obtain Theorem 5.12.

Lemma 5.6 (IPP for PVAL [RVW13])**.** *Let* $J \in (\mathbb{F}^m)^\ell$ *be a sequence of* $\ell = O(\varepsilon n \cdot \log n)$ *points in* $\mathbb{F}^m$*, let* $\vec{v} \in F^\ell$ *be a sequence of* ℓ *corresponding values. There exists a public-coin*[12] IPP *for* $\mathsf{PVAL}_{J,V}$*, with perfect completeness, soundness* $1/2$*, query complexity* $(1/\varepsilon)^{1+o(1)}$*, and* $O(\log(1/\varepsilon)/\log\log n)$ *rounds, where* ε *is the proximity parameter and* $n = |H|^m$ *is the input length. The communication complexity is* $\varepsilon n \cdot (1/\varepsilon)^{o(1)}$*, and the verifier runtime is* $(1/\varepsilon + \varepsilon n)^{1+o(1)}$*. The honest prover runtime is* $\mathsf{poly}(n)$*.*

To amplify the soundness of the protocol, we run the protocol for PVAL in parallel k times:

Corollary 5.15 (Parallel Repetition of [RVW13])**.** *Let* $J \in (\mathbb{F}^m)^\ell$ *be a sequence of* $\ell = O(\varepsilon n \cdot \log n)$ *points in* $\mathbb{F}^m$*, let* $\vec{v} \in F^\ell$ *be a sequence of* ℓ *corresponding values and let* $k \in \mathbb{N}$ *be a parameter. There exists a public-coin* IPP *to* $\mathsf{PVAL}_{J,\vec{v}}$*, with perfect completeness, soundness* 2^{-k}*, query complexity* $k \cdot (1/\varepsilon)^{1+o(1)}$*, and* $O(\log(1/\varepsilon)/\log\log n)$ *rounds, where* ε *is the proximity parameter and* $n = |H|^m$ *is the input length. The communication complexity is* $k \cdot \varepsilon n \cdot (1/\varepsilon)^{o(1)}$*, and the verifier runtime is* $k \cdot (1/\varepsilon + \varepsilon n)^{1+o(1)}$*. The honest prover runtime is* $\mathsf{poly}(k,n)$*.*

Proof. Follows from the fact that parallel repetition decreases the soundness error exponentially. This is known for interactive proofs (for a proof, see [Gol08, Exercise 9.1] or [Gol99, Appendix C.1]), and the proof extends readily for IPPs. □

[12] [RVW13] do not explicitly mention that their protocol is public coin, but by inspection it is easy to see that it is indeed public coin.

We proceed to show that above IPP for PVAL can be composed together with the reduction of Theorem 5.14 to construct a no-signaling MIPP for $\mathcal{L}$. To facilitate the proof, we consider a notion of a proof-system that is a hybrid between a no-signaling MIPP and a single-prover public-coin IPP.

Hybrid MIPP. We think of the hybrid MIPP as being composed of $\ell + 1$ provers and a verifier. The verifier first interacts with the first ℓ provers as in a (one-round) MIPP. Then, the verifier continues the interaction with the last prover as a (single-prover) public-coin IPP (which may consist of multiple rounds). Loosely speaking, "Hybrid soundness" for a language $\mathcal{L}$ will require that cheating strategies in which the first ℓ provers are δ-no-signaling (but do not depend on the messages of the last prover, even as a function) and the last prover (of the interactive proof) is classical but is given all the queries of the first ℓ provers as input, cannot convince the verifier to accept x that is ε-far from $\mathcal{L}$ (except with some bounded probability).

Our actual definition of a hybrid MIPP will be slightly different than that described above. Since we deal with a public-coin ℓ'-round protocol, it will be convenient to think of this protocol as an MIPP with ℓ' provers, $P_1, \ldots, P_{\ell'}$, where in the soundness condition we let P_i see all messages seen by provers $P_1, \ldots, P_{i-1}$. Using this modification we can define the hybrid MIPP as an MIPP (i.e., completeness is as in a standard MIPP) that satisfies the following *hybrid soundness* condition.

Definition 5.7. *We say that an* MIPP *for $\mathcal{L}$ with $\ell + \ell'$ provers has soundness s against* (ℓ, ℓ', δ)-hybrid strategies *if the following condition holds:*

- ***Hybrid Soundness.*** *For every $\varepsilon > 0$, every x that is ε-far from $\mathcal{L}$, every δ-no-signaling family of distributions $\mathcal{A} = \{\mathcal{A}_{\mathbf{q}}\}_{\mathbf{q} \in D^{\ell}}$, and every (deterministic) cheating strategies $P_1^*, \ldots, P_{\ell'}^*$, the probability that the verifier accepts when given as input n, ε, answers $\mathbf{a} = (a_1, \ldots, a_{\ell+\ell'}) \in \Sigma^{\ell+\ell'}$ and oracle access to x is at most s, where $\mathbf{q} = (q_1, \ldots, q_{\ell+\ell'}) \in D^{\ell+\ell'}$ are the queries generated by the verifier, $(a_1, \ldots, a_{\ell}) \in_R \mathcal{A}_{(q_1,\ldots,q_{\ell})}$ and for every $i \in [\ell']$ it holds that $a_{\ell+i} = P_i^*(q_1, \ldots, q_{\ell+i}, a_1, \ldots, a_{\ell+i-1})$.*

The following lemma shows that we can compose an MIPP that is sound against δ-no-signaling strategies with an IPP to obtain an MIPP with hybrid soundness. Using this lemma together with Theorem 5.14 and Corollary 5.15 we obtain an MIP with hybrid soundness for every language in P. Later we will show how to transform such a hybrid MIPP into an MIPP that is sound against no-signaling strategies.

Lemma 5.8. *Let $\mathcal{L}$ be a language and let $\{\mathcal{L}'_{\alpha}\}_{\alpha}$ be a family of languages such that the following holds:*

1. *There exists a 1-round protocol between a probabilistic verifier, which takes as input n and $\varepsilon > 0$, and ℓ provers, which take as input $\varepsilon > 0$ and $x \in \{0,1\}^n$, such that at the end of the interaction, the verifier outputs a value α and the following two conditions hold:*

- ***Completeness.*** *For every $\varepsilon > 0$, and every $x \in \mathcal{L}$ if the provers honestly follows the protocol, then with probability 1, it holds that $x \in \mathcal{L}'_\alpha$.*
- ***Soundness.*** *For every $\varepsilon > 0$, every x that is ε-far from $\mathcal{L}$, and every δ-no-signaling cheating family of distributions $\mathcal{A}$, it holds that with probability $\geq 1 - s$ over the verifiers coins and a random sample from $\mathcal{A}$, it holds that x is ε-far from $\mathcal{L}'_\alpha$.*

2. *For every value of α, the language $\mathcal{L}'_\alpha$ has an ℓ' round public-coin* IPP *with prefect completeness and soundness s'.*

*Then, $\mathcal{L}$ has an $\ell + \ell'$-prover ε-*MIPP *with perfect completeness and soundness $s + s'$ against (ℓ, ℓ', δ)-hybrid strategies.*

If the original MIPP *protocol has query alphabet D, answer alphabet Σ (these alphabets refer to the communication with the provers) and randomness complexity r, and the* IPP *verifier has total communication c', then the resulting hybrid* MIPP *has query alphabet $D \cup \{0,1\}^{r+c'}$ and answer alphabet $\Sigma \cup \{0,1\}^{c'}$. If the original* MIPP *verifier runs in time T_V and the* IPP *verifier runs in time T'_V, then the resulting hybrid* MIPP *verifier runs in time $T_V + T'_V$. If the* IPP *verifier has query complexity Q (to the main input), then the resulting* MIPP *verifier has query complexity Q (to the main input). If each of the original* MIPP *provers runs in time T_P and the next message function of the* IPP *verifier can be computed in time T'_P, then each of the resulting hybrid* MIPP *provers runs in time $T_P + T'_P$.*

Proof. Let $(V, P_1, \ldots, P_\ell)$ be a 1-round multi-prover protocol for computing α as above. Let D be the query alphabet and Σ the answer alphabet of the protocol (used in the communication with the provers). We think of V as being composed of two algorithms V_1 and V_2 as follows. The algorithm V_1, given as input $n \in \mathbb{N}$, $\varepsilon > 0$ and a random string r, generates a sequence of queries $\mathbf{q} = (q_1, \ldots, q_\ell) \in D^\ell$ and sends the query q_i to prover P_i. The algorithm V_2, given as input $n \in \mathbb{N}$, $\varepsilon > 0$ and same random string r and a sequence of answers $\mathbf{a} = (a_1, \ldots, a_\ell) \in \Sigma^\ell$ outputs a value α.

Let (V', P') be a public-coin interactive proof of ε-proximity to $\mathcal{L}'_\alpha$, with perfect completeness and soundness s', where α is given as a parameter to both V' and P'. Since V' is public-coin, its messages consist of uniformly random coins and in particular do not depend on the value of α nor on x. We think of the prover algorithm P' as being composed of ℓ' algorithms $P'_1, \ldots, P'_{\ell'}$ where P'_i is simply the next message function of P' (and may depend on α, x and on the first i messages that P' receives).

We use $(V_1, P_1, \ldots, P_\ell)$ and $(V', P'_1, \ldots, P'_{\ell'})$ to construct an $\ell + \ell'$ prover MIP of ε-proximity $(V'', P''_1, \ldots, P''_{\ell+\ell'})$ for $\mathcal{L}$ with soundness $s+s'$ against (ℓ, ℓ', δ)-hybrid strategies. The verifier V'' on input n, $\varepsilon > 0$, a random string (r, r') and oracle access to x, operates as follows. First it runs $V_1(n, \varepsilon, r)$ to obtain a sequence of queries $\mathbf{q} = (q_1, \ldots, q_\ell) \in D^\ell$. For every $i \in [\ell]$ the query q_i is sent to P''_i. The verifier V'' also runs V' on input n, ε to generates the ℓ' messages $(m_1, \ldots, m_{\ell'})$ of V' (as noted above, since m_i is just a random string, we do not need α nor oracle access to x for this step). For every $i \in [\ell']$, V'' sends $(r, m_1, \ldots, m_i)$ to $P''_{\ell+i}$.

For $i \in [\ell]$, the prover P''_i is identical to the prover P_i. For $i \in [\ell']$, the prover $P''_{\ell+i}$, on input $(x, \varepsilon, (r, m_1, \ldots, m_i))$ first computes the value of α (by simulating the provers P_i and the verifier V_1, while using r as its random coins) and then outputs $P'_i(x, \varepsilon, \alpha, m_1, \ldots, m_i)$, where $m_1, \ldots, m_i$ are interpreted as the first i messages in the interactive proof.

To decide whether to accept, given as input n, ε, the random string (r, r'), prover answers $(a_1, \ldots, a_{\ell+\ell'})$, and oracle access to x, the verifier V'' first runs V_2 on input n, ε, r and $(a_1, \ldots, a_\ell)$ to obtain the value α. Then, it outputs the result of V' on input n, ε, r', α while forwarding V' oracle queries to x (using its own oracle queries).

We first argue that the resulting MIPP has perfect completeness and then proceed to prove soundness against (ℓ, ℓ', δ)-no-signaling strategies. To show that completeness holds, observe that when the verifier V'' interacts with the honest provers on input $x \in \mathcal{L}$, since the protocol $(V, P_1, \ldots, P_\ell)$ has perfect completeness, it holds that $x \in \mathcal{L}_\alpha$ and therefore V''_2 runs V' on a value α such that $x \in \mathcal{L}_\alpha$. The perfect completeness of the protocol follows from the prefect completeness of (V', P').

To show that (ℓ, ℓ', δ)-hybrid soundness holds, assume for a contradiction that there exists some δ-no-signaling cheating strategy $\{\mathcal{A}_\omega\}_{\omega \in D^\ell}$ and (deterministic) cheating strategies $P^*_1, \ldots, P^*_\ell$ that fool V'' into accepting some x that is ε-far from $\mathcal{L}$ with probability $\geq s + s'$.

By elementary manipulations we have that:

$$s + s' \leq \Pr[V'' \text{ accepts}] \leq \Pr[V' \text{ accepts} \wedge x \text{ is } \varepsilon\text{-far from } \mathcal{L}_\alpha] + \Pr[x \text{ is } \varepsilon\text{-close to } \mathcal{L}_\alpha]$$

We obtain a contradiction by bounding each term separately and showing that their sum is smaller than $s + s'$.

Claim 5.8.1.

$$\Pr\left[x \textit{ is } \varepsilon\textit{-close to } \mathcal{L}_\alpha\right] < s$$

Proof. Follows directly from the soundness condition of $(V, P_1, \ldots, P_\ell)$. □

Claim 5.8.2.

$$\Pr[V' \textit{ accepts} \wedge x \textit{ is } \varepsilon\textit{-far from } \mathcal{L}_\alpha] < s'$$

Proof. Suppose otherwise. Then by an averaging argument there exist fixed values of $r = r^*$ and $(a_1, \ldots, a_\ell) = (a^*_1, \ldots, a^*_\ell)$ such that:

$$\Pr\left[V' \text{ accepts} \wedge x \text{ is } \varepsilon\text{-far from } \mathcal{L}_\alpha \mid r = r^* \wedge (a_1, \ldots, a_\ell) = (a^*_1, \ldots, a^*_\ell)\right] \geq s'$$

Once r and $(a_1, \ldots, a_\ell)$ are fixed, then $\alpha = V_2(n, \varepsilon, r^*, (a^*_1, \ldots, a^*_\ell))$ is also fixed. Hence, there exists a value of α such that x is ε-far from $\mathcal{L}_\alpha$ and

$$\Pr\left[V' \text{ accepts} \mid r = r^* \wedge (a_1, \ldots, a_\ell) = (a^*_1, \ldots, a^*_\ell)\right] \geq s'$$

We construct a cheating strategy P^{**} that fools V' into accepting that $x \in \mathcal{L}_\alpha$ with probability $\geq s'$, even though x is ε-far from $\mathcal{L}_\alpha$. This contradicts the soundness of the proof-system (V', P'). The prover P^{**} simply simulates P^* conditioned on $r = r^*$ and on $(a_1, \ldots, a_\ell) = (a^*_1, \ldots, a^*_\ell)$.

This concludes the proof of Claim 5.8.2. □

This concludes the proof of Lemma 5.8. □

Lemma 5.8, combined with Theorem 5.14 and Corollary 5.15, implies the following lemma.

Lemma 5.9. *Let $\mathcal{L} \in \mathsf{DTIME}(t(n))$, where $t = t(n)$ satisfies $\mathsf{poly}(n) \leq t \leq \mathsf{exp}(n)$, let k be an integer such that $(\log t)^c \leq k \leq \mathsf{poly}(n)$, where c is some (sufficiently large) universal constant, and let $\varepsilon = \varepsilon(n) \in (0,1)$.*

Then, $\mathcal{L}$ has an ε-MIPP with ℓ provers, where $\ell = (k+\varepsilon n)\cdot\mathsf{polylog}(t)$, with perfect completeness and soundness 2^{-k} against (ℓ,ℓ',δ)-hybrid strategies, where $\ell' = O(\log(1/\varepsilon)/\log\log n)$ and $\delta = 2^{-(k+\varepsilon n)\cdot\mathsf{polylog}(t)}$.

The queries and answers of the MIPP (to the provers) are of length $(k+\varepsilon n)\cdot\mathsf{polylog}(t)+k\cdot\varepsilon n\cdot(1/\varepsilon)^{o(1)}$. The verifier runs in time $(k+\varepsilon n)^2\cdot\mathsf{polylog}(t)+k\cdot(1/\varepsilon+\varepsilon n)^{1+o(1)}$ and has query complexity $k\cdot(1/\varepsilon)^{1+o(1)}$ (to the main input). Each of the (honest) provers runs in time $\mathsf{poly}(t,k)$.

Proof. Follows directly from Lemma 5.8, combined with Theorem 5.14 and Corollary 5.15 (with respect to $k' = (k+\varepsilon\cdot n)\cdot\mathsf{polylog}(t)$). □

The following lemma shows that an MIPP with soundness against hybrid strategies has soundness against no-signaling strategies (with a slight loss in parameters).

Lemma 5.10. *Let $\varepsilon = \varepsilon(n) > 0$. Suppose that $(V, P_1, \ldots, P_{\ell+\ell'})$ is an ε-MIPP for a language $\mathcal{L}$ with soundness s against (ℓ,ℓ',δ)-hybrid strategies. Then, $(V, P_1, \ldots, P_{\ell+\ell'})$ is an ε-MIP of proximity to $\mathcal{L}$ with soundness $s+\ell'\cdot\delta$ against δ-no-signaling strategies.*

Lemma 5.10 follows directly from ℓ' applications of the following proposition.

Proposition 5.11. *Let $\varepsilon = \varepsilon(n) > 0$. Suppose that $(V, P_1, \ldots, P_{\ell+\ell'})$ is an ε-MIPP for a language $\mathcal{L}$ with soundness s against (ℓ,ℓ',δ)-hybrid strategies. Then, $(V, P_1, \ldots, P_{\ell+\ell'})$ is an ε-MIPP for $\mathcal{L}$ with soundness $s+\delta$ against $(\ell+1,\ell'-1,\delta)$-hybrid strategies.*

Proof. To prove the proposition, it suffices to show that V has soundness $s+\delta$ against $(\ell+1,\ell'-1,\delta)$-hybrid strategies. Let x be ε-far from $\mathcal{L}$, let $\mathcal{A} = \{\mathcal{A}_{\mathbf{q}}\}_{q\in D^{\ell+1}}$ be a δ-no-signaling family of distributions and let $P_2^*, \ldots, P_{\ell'}^*$ be cheating strategies, such that $\mathcal{A}, P_2^*, \ldots, P_{\ell'}^*$ violate the $(\ell+1,\ell'-1,\delta)$-hybrid soundness of V with probability $s+\delta$. Using $\mathcal{A}, P_2^*, \ldots, P_\ell^*$ we will break the (ℓ,ℓ',δ)-hybrid soundness of V, contradicting our assumption.

Let $\sigma \in D$ be an arbitrary element in D. For every $\omega \in D^\ell$, let $\mathcal{B}_\omega = \mathcal{A}_{(\omega,\sigma)}$. Let $P_1^*(q_1, \ldots, q_{\ell+1}, a_1, \ldots, a_\ell)$ be defined as follows. The prover P_1^* first samples $(a'_1, \ldots, a'_{\ell+1}) \in_R \mathcal{A}_{q_1,\ldots,q_{\ell+1}}$ *conditioned* on $a'_i = a_i$ for every $i \in [\ell]$ and outputs $a'_{\ell+1}$. In the case that the conditional probability space is empty, P_1^* sends an arbitrary response (which can be thought of as aborting).

Claim 5.11.1. *$\mathcal{B} = \{\mathcal{B}_\omega\}_{\omega\in D^\ell}$ is δ-no-signaling.*

Proof. Let $S \subseteq [\ell]$ and let $\omega_1, \omega_2 \in D^\ell$. Suppose that $(\mathbf{a}_1)_S$ and $(\mathbf{a}_2)_S$ are δ-far, where $\mathbf{a}_1 \in_R \mathcal{B}_{\omega_1}$ and $\mathbf{a}_2 \in_R \mathcal{B}_{\omega_2}$. Hence, the projections of the distributions $\mathcal{A}_{(\omega_1,\sigma)}$ and $\mathcal{A}_{(\omega_2,\sigma)}$ to coordinates in $S \subseteq [\ell+1]$ are δ-far, in contradiction to our assumption that $\mathcal{A}$ is δ-no-signaling. □

Claim 5.11.2. *For every sequence of queries $q_1, \ldots, q_{\ell+1} \in D$, the following two distributions are δ-close:*

- $(a_1, \ldots, a_{\ell+1}) \in_R \mathcal{A}_{(q_1,\ldots,q_{\ell+1})}$.
- $(a'_1, \ldots, a'_{\ell+1})$ *where* $(a'_1, \ldots, \mathbf{a}'_\ell) \in_R \mathcal{A}_{(q_1,\ldots,q_\ell,\sigma)}$ *and* $a'_{\ell+1} = P^*_1(q_1, \ldots, q_{\ell+1}, a'_1, \ldots, a'_\ell)$.

Proof. We first note that $(a_1, \ldots, a_{\ell+1})$ and $(a_1, \ldots, a_\ell, P^*(q_1, \ldots, q_{\ell+1}, a_1, \ldots, a_\ell))$ are identically distributed. Hence, we need to show that the distributions

$$(a_1, \ldots, a_\ell, P^*(q_1, \ldots, q_{\ell+1}, a_1, \ldots, a_\ell))$$

and

$$(a'_1, \ldots, a'_\ell, P^*(q_1, \ldots, q_{\ell+1}, a'_1, \ldots, a'_\ell))$$

are δ-close. The latter follows from the fact that $\mathcal{A}$ is δ-no-signaling. □

By our assumption that $\mathcal{A}$ and $P_2, \ldots, P_\ell$ violate the $(\ell+1, \ell'-1, \delta)$-hybrid soundness of V, it holds that V accepts x with probability at least $s+\delta$ when given answers $(a_1, \ldots, a_{\ell+\ell'})$ where $(a_1, \ldots, a_{\ell+1}) \in_R \mathcal{A}_{(q_1,\ldots,q_{\ell+1})}$ and $a_{\ell+i} = P_i(q_1, \ldots, q_{\ell+i}, a_1, \ldots, a_{\ell+i-1})$ for $i \in \{2, \ldots, \ell\}$, where $q_1, \ldots, q_{\ell+\ell'}$ are V's queries (to the provers). Hence, by Claim 5.11.2, the verifier accepts x with probability at least s when given answers $(a'_1, \ldots, a'_{\ell+\ell'})$ where where $(a'_1, \ldots, \mathbf{a}'_\ell) \in_R \mathcal{A}_{(q_1,\ldots,q_\ell,\sigma)}$ and $a_{\ell+i} = P_i(q_1, \ldots, q_{\ell+i}, a'_1, \ldots, a'_{\ell+i-1})$ for $i \in [\ell]$. □

We conclude this section by noting that Theorem 5.12 follows directly from Lemma 5.9 combined with Lemma 5.10.

5.6 Arguments of Proximity for P

In this section we construct arguments of proximity for every deterministic language:

Theorem 5.16. *Suppose that $\mathcal{L} \in \mathsf{DTIME}(t(n))$, where $t = t(n)$ satisfies $\mathsf{poly}(n) \leq t \leq \mathsf{exp}(n)$. Let β and β' be constants such that $0 < \beta < \beta' < 1/2$. Fix a proximity parameter $\varepsilon \stackrel{\text{def}}{=} n^{-(1-\beta)}$ and a security parameter $\tau \stackrel{\text{def}}{=} n^{2\beta'+o(1)} \cdot \mathsf{polylog}(t)$.*

If there exists a $\left(T, 2^{-\sqrt{\log T}}\right)$-secure FHE, where $T(\tau) = 2^{\tau^{\beta/\beta'}}$, then $\mathcal{L}$ has a 1-round argument of ε-proximity with soundness $\left(T, 2^{-n^{o(1)} \cdot \mathsf{polylog}(t)}\right)$.

The verifier runs in time $n^{1-\beta+o(1)} \cdot \mathsf{polylog}(t) + \mathsf{poly}_{\mathsf{FHE}}(\tau)$, where $\mathsf{poly}_{\mathsf{FHE}}$ is a polynomial that depends only on the FHE scheme, and makes $n^{1-\beta+o(1)} \cdot \mathsf{polylog}(t)$ oracle queries to the main input. The prover runs in time $\mathsf{poly}(t)$. The total communication is of length $\mathsf{poly}_{\mathsf{FHE}}(\tau)$.

We also state the immediate corollary for languages in P:

Corollary 5.17. *Let $\mathcal{L} \in \mathsf{P}$. Let β and β' be any constants such that $0 < \beta < \beta' < 1/2$. Fix a proximity parameter $\varepsilon = n^{-(1-\beta)}$ and a security parameter $\tau = n^{2\beta'+o(1)}$.*

If there exists a $\left(T, 2^{-\sqrt{\log T}}\right)$-secure FHE, where $T(\tau) = 2^{\tau^{\beta/\beta'}}$, then $\mathcal{L}$ has a 1-round argument of ε-proximity with soundness $\left(T, 2^{-n^{o(1)}}\right)$.

The verifier runs in time $n^{1-\beta+o(1)} + \mathsf{poly}_{\mathsf{FHE}}(\tau)$, where $\mathsf{poly}_{\mathsf{FHE}}$ is a polynomial that depends only on the FHE scheme, and makes $n^{1-\beta+o(1)}$ oracle queries to the main input. The prover runs in time $\mathsf{poly}(n)$. The total communication is of length $\mathsf{poly}_{\mathsf{FHE}}(\tau)$.

We note that by setting β and β' to be sufficiently small, assuming the existence of a sub-exponentially secure FHE, we obtain an argument of proximity with a *sub-linear* time verifier.

To prove Theorem 5.16, we show (in Theorem 5.18) how to transform any MIPP that has soundness against no-signaling strategies into a 1-round argument of proximity, using a fully-homomorphic encryption scheme (FHE) . The transformation is similar to (and is based on) the corresponding transformation from [KRR13a] from MIPs (i.e., without proximity) that have soundness against no-signaling strategies into 1-round argument-systems. Theorem 5.16 will follow by applying the transformation of Theorem 5.18 to the MIPP of Theorem 5.12.

Theorem 5.18. *Let $\varepsilon = \varepsilon(n) > 0$ be a proximity parameter. Suppose that the language $\mathcal{L}$ has an ℓ prover ε-MIPP that has soundness s against δ-no-signaling strategies. Let D be the query alphabet and Σ be the answer alphabet of the MIPP (used in the interaction with the provers). Let $\tau = \tau(n)$ be a security parameter, where n denotes the input length. For every $T = T(\tau) \geq \tau$ such that $T \geq \max(n, 2^{\ell \log(|\Sigma|)})$ and $\delta' = \delta'(\tau)$ such that $\delta' \leq \delta/\ell$, if there exists a (T, δ')-secure FHE, then the language $\mathcal{L}$ has a 1-round argument of ε-proximity with soundness (T, s).*

If the MIPP verifier runs in time T_V, then the running time of the resulting verifier is $T_V + T_{\mathsf{FHE}}(\tau, \ell, \log(|D|), \log(|\Sigma|))$, where T_{FHE} is a polynomial that depends only on the encryption scheme (and not on the language $\mathcal{L}$). If the MIPP verifier makes Q oracle queries to the main input then the resulting verifier also makes Q oracle queries to the main input. If the running time of each MIPP prover is T_P, then the running time of the resulting prover is $\mathsf{poly}\big(T_P, \tau, n, \ell, \log(|D|), \log(|\Sigma|)\big)$. The total communication in the resulting argument-system is of length $\mathsf{poly}(\tau, \ell, \log(|D|), \log(|\Sigma|))$.

Proof. Let $(V, P_1, \ldots, P_\ell)$ be an ℓ prover MIPP for $\mathcal{L}$ with soundness s against δ-no-signaling strategies. Let D be the query alphabet and Σ be the answer alphabet. Since $(V, P_1, \ldots, P_\ell)$ is a 1-round protocol, it will be convenient for us to think of V as being composed of two algorithms that use the same randomness, V_1 and V_2. The first algorithm, V_1, on input n, the proximity parameter $\varepsilon > 0$, the random string r and oracle access to x, outputs a sequence of ℓ queries $\mathbf{q} \in D^\ell$. The second algorithm, V_2, on input n, proximity parameter ε, the same random string r, the answers $\mathbf{a} \in \Sigma^\ell$ and oracle access to x, outputs a bit that represents whether it believes that $x \in \mathcal{L}$ or x is ε-far from $\mathcal{L}$.

We assume without loss of generality that the (honest) provers algorithms $P_1, \ldots, P_\ell$ are deterministic.

Let $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval})$ be an FHE and let $\tau = \tau(n)$ be a security parameter. We use the MIPP and FHE to construct an argument of proximity (V', P') as follows. The verifier, given as input n, a proximity parameter $\varepsilon > 0$, a random string r and oracle access to $x \in \{0,1\}^n$, first invokes V_1 on input n, $\varepsilon >$ while forwarding V_1's oracle queries to x (i.e., its own oracle), to obtain a sequence of ℓ queries $\mathbf{q} = (q_1, \ldots, q_\ell) \in D^\ell$. Then, for every $i \in [\ell]$, the verifier invokes $\mathsf{Gen}(1^\tau)$, where $\tau = \tau(n)$, to obtain a key-pair $(\mathsf{pk}_i, \mathsf{sk}_i)$. The verifier then runs $\mathsf{Enc}_{\mathsf{pk}_i}(q_i)$ to obtain a ciphertext $\hat{q}_i$, for every $i \in [\ell]$. We denote $\mathsf{pk} = (\mathsf{pk}_1, \ldots, \mathsf{pk}_\ell)$, and $\hat{\mathbf{q}} \stackrel{\text{def}}{=} (\hat{q}_1, \ldots, \hat{q}_\ell)$. The verifier sends $(\mathsf{pk}, \hat{\mathbf{q}})$ to P'.

The prover P' is given as input x, a proximity parameter $\varepsilon > 0$, and a message $(\mathsf{pk}, \hat{\mathbf{q}})$ from the verifier. For every $i \in [\ell]$, let $\mathbb{C}_{x,i} : D \to \Sigma$ be a Boolean circuit that on input q computes the function $P_i(x, q)$. For every $i \in [\ell]$, the prover P' computes $\hat{a}_i = \mathsf{Eval}(\mathsf{pk}_i, C_{x,i}, \hat{q}_i)$. The prover sends $\hat{\mathbf{a}} \stackrel{\text{def}}{=} (\hat{a}_1, \ldots, \hat{a}_\ell)$ to the verifier.

The verifier V', given the message $\hat{\mathbf{a}}$ from the prover, computes $a_i = \mathsf{Dec}_{\mathsf{sk}_i}(\hat{a}_i)$, for every $i \in [\ell]$. The verifier outputs the result of $V_2^x(n, \varepsilon, (a_1, \ldots, a_\ell), r)$, where r is the same random string used by V_1 to generate the queries, and the superscript denotes oracle access to x.

We proceed to show that (V', P') is an argument of proximity with soundness (T, s) (see definition in Section 5.3.2).

Completeness. Let $x \in \mathcal{L}$. By the construction and the correctness of the FHE protocol, for every $i \in [\ell]$ it holds that $a_i = P_i(x, q_i)$, with overwhelming probability. When V_2 is invoked with the answers of the honest provers $P_1, \ldots, P_\ell$, by the (perfect) completeness of the MIPP, the verifier V outputs 1. Hence, V' accepts with overwhelming probability.

Soundness. Fix any $\varepsilon = \varepsilon(n) > 0$. Assume for the sake of contradiction that there exists a family of circuits $\{P_n^*\}_{n \in \mathbb{N}}$ of size at most $\mathsf{poly}(T(n))$ such that there exist infinitely many x that are ε-far from $\mathcal{L}$ such that

$$\Pr[(P_{|x|}^*, V')(x) = 1] > s, \tag{5.3}$$

where $(P_{|x|}^*, V')(x)$ denotes the output of V' after interacting with the prover $P_{|x|}^*$ where V' has oracle access to x and $P_{|x|}^*$ has direct access to x (and the probability is over the random coins of V'). We show a contradiction by constructing a δ-no-signaling (cheating) strategy that fools the underlying MIPP verifier V into accepting some x that is ε-far from $\mathcal{L}$ with probability greater than s.

For every x that is ε-far from $\mathcal{L}$, consider an MIPP prover strategy $\mathcal{A}^{(x)} \stackrel{\text{def}}{=} \{\mathcal{A}_{\mathbf{q}}^{(x)}\}_{\mathbf{q} \in D^\ell}$, where for every $\mathbf{q} = (q_1, \ldots, q_\ell) \in D^\ell$, the distribution $\mathcal{A}_{\mathbf{q}}^{(x)}$ is sampled as follows. First, for every $i \in [\ell]$ invoke $\mathsf{Gen}(1^\tau)$, where $\tau = \tau(|x|)$, to obtain $(\mathsf{pk}_i, \mathsf{sk}_i)$ and compute $\hat{q}_i \in_R \mathsf{Enc}_{\mathsf{pk}_i}(q_i)$. Then, compute $\hat{\mathbf{a}} = (\hat{a}_1, \ldots, \hat{a}_\ell) \in_R P_{|x|}^*(x, (\mathsf{pk}, \hat{\mathbf{q}}))$, where $\mathsf{pk} = (\mathsf{pk}_1, \ldots, \mathsf{pk}_\ell)$ and $\hat{\mathbf{q}} = (\hat{q}_1, \ldots, \hat{q}_\ell)$, and for every $i \in [\ell]$, set $a_i = \mathsf{Dec}_{\mathsf{sk}_i}(\hat{a}_i)$. Finally, output $\mathbf{a} \stackrel{\text{def}}{=} (a_1, \ldots, a_\ell)$.

By the definition of $\mathcal{A}^{(x)}$ and V' and using Eq. (5.3), for infinitely many x that are ε-far from $\mathcal{L}$, it holds that

$$\Pr_{\substack{r \\ \mathbf{a}\in_R \mathcal{A}^{(x)}_{\mathbf{q}}}}[V_2^x(n,\varepsilon,\mathbf{a},r)=1] = \Pr\left[(P^*_{|x|},V')(x)=1\right] > s$$

where $\mathbf{q} = V_1(x,r)$ and the superscript denotes oracle access to x. It remains to show that for every sufficiently large x that is ε-far from $\mathcal{L}$, the strategy $\mathcal{A}^{(x)}$ is δ-no-signaling.

We need to prove that for all sufficiently large x that are ε-far from $\mathcal{L}$, every $S \subseteq [\ell]$, and every two sequences of queries $\mathbf{q} = (q_1,\ldots,q_\ell) \in D^\ell$ and $\mathbf{q}' = (q'_1,\ldots,q'_\ell) \in D^\ell$ such that $\mathbf{q}_S = \mathbf{q}'_S$ (i.e., $q_i = q'_i$ for all $i \in S$), the following two distributions are δ-close:

- $\mathbf{a}_S$ where $\mathbf{a} \in_R \mathcal{A}^{(x)}_{\mathbf{q}}$; and
- $\mathbf{a}'_S$ where $\mathbf{a}' \in_R \mathcal{A}^{(x)}_{\mathbf{q}'}$.

Toward this end, assume for a contradiction that for infinitely many x this is not the case. That is, for infinitely many x, there exist corresponding S, $\mathbf{q}$, $\mathbf{q}'$ and a distinguisher $\mathcal{D}_x$ such that

$$\left|\Pr_{\mathbf{a}\in_R\mathcal{A}^{(x)}_{\mathbf{q}}}[\mathcal{D}_x(\mathbf{a}_S)=1] - \Pr_{\mathbf{a}'\in_R\mathcal{A}^{(x)}_{\mathbf{q}'}}[\mathcal{D}_x(\mathbf{a}'_S)=1]\right| > \delta. \tag{5.4}$$

Since $\mathcal{D}_x$ takes as input a string of length at most $\ell\cdot\log(|\Sigma|)$, it can be implemented by a circuit of size at most $\mathsf{poly}(2^{\ell\cdot\log(|\Sigma|)})$. We use $\{\mathcal{D}_x\}_x$ to construct a family of circuits $\{\mathbb{C}_\tau\}_\tau$ that breaks the security of the underlying FHE scheme.

For every x as above and for $\tau = \tau(|x|)$, let $\mathbb{C}_\tau$ be a circuit that takes as input a set of public-keys $\{\mathsf{pk}_i\}_{i\in[\ell]\setminus S}$ (with respect to security parameter τ) and a set of ciphertexts $\{c_i\}_{i\in[\ell]\setminus S}$. We show that the circuit $\mathbb{C}_\tau$ distinguishes between the case that (1) each c_i was sampled from $\mathsf{Enc}_{\mathsf{pk}_i}(q_i)$; and the case that (2) each c_i was sampled from $\mathsf{Enc}_{\mathsf{pk}_i}(q'_i)$. The circuit $\mathbb{C}_\tau$ works as follows:

1. For every $i \in S$, sample $(pk_i, sk_i) \in_R \mathsf{Gen}(1^\tau)$ and $c_i \in_R \mathsf{Enc}_{pk_i}(q_i)$. Set $\mathsf{pk} = (pk_1,\ldots,pk_\ell)$ and $\mathbf{c} = (c_1,\ldots,c_\ell)$ (note that for $i \notin S$, the values pk_i and c_i are given as input to the circuit).
2. Compute $\hat{\mathbf{a}} \stackrel{\text{def}}{=} (\hat{a}_1,\ldots,\hat{a}_\ell) = P^*_{|x|}(x,\mathsf{pk},\mathbf{c})$, where $P^*_{|x|}(x,\mathsf{pk},\mathbf{c})$ denotes the output of $P^*_{|x|}$ given as input x and a message $(\mathsf{pk},\mathbf{c})$. (Note that x is fixed inside the description of $\mathbb{C}_\tau$.)
3. For every $i \in S$, set $a_i = \mathsf{Dec}_{sk_i}(\hat{a}_i)$.
4. Output $\mathcal{D}_x(\mathbf{a}_S)$ where $\mathbf{a}_S = (a_i)_{i\in S}$.

Note that $\mathbb{C}_\tau$ has size $\mathsf{poly}\big(2^{\ell\cdot\log(|\Sigma|)},\tau,T(\tau),|x|\big) \le \mathsf{poly}(T(\tau))$.

By Eq. (5.4), the circuit $\mathbb{C}_\tau$ distinguishes between the two cases with probability δ for infinitely many values of τ. By a standard hybrid argument we obtain a circuit that breaks the semantic security of the encryption scheme with distinguishing gap at least $\delta/\ell \ge \delta'(\tau)$ in contradiction to our assumption. Thus, we obtain that for all sufficiently large $x \notin \mathcal{L}$, the prover strategy $\mathcal{A}^{(x)}$ is δ-no-signaling and the lemma follows. □

5.6.1 Proof of Theorem 5.16

Suppose that $\mathcal{L} \in \mathsf{DTIME}(t(n))$, where $t = t(n)$ satisfies $\mathsf{poly}(n) \leq t \leq \mathsf{exp}(n)$. Let β and β' be constants such that $0 < \beta < \beta' < 1/2$. Fix a proximity parameter $\varepsilon \stackrel{\text{def}}{=} n^{-(1-\beta)}$ and a security parameter $\tau \stackrel{\text{def}}{=} n^{2\beta'+o(1)} \cdot \mathsf{polylog}(t)$.

Let $T(\tau) = 2^{\tau^{\beta/\beta'}}$ and fix $k \stackrel{\text{def}}{=} n^{o(1)} \cdot \mathsf{polylog}(t)$ and $\delta \stackrel{\text{def}}{=} 2^{-\sqrt{\log T}}$. Note that the following inequalities hold:

1. $(\log(t))^c \leq k \leq \mathsf{poly}(n)$, where the constant c is as in Theorem 5.12.
2. $T \geq \max\left(n, 2^{(k+\varepsilon n)^2 \cdot \mathsf{polylog}(t) \cdot n^{o(1)}}\right)$.
3. $\delta \leq 2^{-(k+\varepsilon n)\mathsf{polylog}(t)}$.

By applying Theorem 5.12 (with soundness parameter k) to the language $\mathcal{L}$, we obtain an ε-MIPP for $\mathcal{L}$ with ℓ provers, where $\ell = (k+\varepsilon n) \cdot \mathsf{polylog}(t)$, perfect completeness and soundness 2^{-k} against δ-no-signaling strategies, where $\delta = 2^{-(k+\varepsilon n)\cdot\mathsf{polylog}(t)}$.

The queries and answers (to the provers) are of length $(k+\varepsilon n) \cdot \mathsf{polylog}(t) + k \cdot \varepsilon n \cdot (1/\varepsilon)^{o(1)}$. The verifier runs in time $(k+\varepsilon n)^2 \cdot \mathsf{polylog}(t) + k \cdot (1/\varepsilon + \varepsilon n)^{1+o(1)}$ and has query complexity (to the main input) $k \cdot (1/\varepsilon)^{1+o(1)}$. Each of the (honest) provers runs in time $\mathsf{poly}(t, k)$.

Assume that there exists a (T, δ)-secure FHE. By Theorem 5.18 (and our setting of k, T and δ), we obtain that $\mathcal{L}$ has a 1-round argument of proximity with soundness $(T, 2^{-k})$.

The verifier runs in time $k \cdot (1/\varepsilon + \varepsilon n)^{1+o(1)} + \mathsf{poly}(\tau)$ and makes $k \cdot (1/\varepsilon)^{1+o(1)}$ oracle queries to the main input. The prover runs in time $\mathsf{poly}(t)$. The total communication is of length $\mathsf{poly}(\tau)$.

Appendix A

Works not included in this Thesis

In this section we provide a high-level description of results obtained during our doctoral studies and were not included above. See the links provided below for the full versions.

A.1 Efficient Multiparty Protocols via Log-Depth Threshold Formulae [CDI+13]

Secure multiparty computation (MPC) enables a set of parties to jointly accomplish some distributed computational task, while maintaining the secrecy of the inputs and the correctness of the outputs in the presence of coalitions of dishonest parties. Originating from the seminal works of [Yao82, GMW87, BGW88, CCD88], secure MPC has been the subject of an enormous body of work.

Together with Cohen, Damgård, Ishai, Kölker, Miltersen and Raz [CDI+13], we put forward a new approach for the design of efficient multiparty protocols:

1. Design a protocol π for a small number of parties (say, 3 or 4) that achieves security against a *single* corrupted party. Such protocols are typically easy to construct, because they may employ techniques that do not scale well with the number of corrupted parties.

2. Recursively compose π with itself to obtain an efficient n-party protocol that achieves security against a constant fraction of corrupted parties.

The second step of our approach builds on the "player emulation" technique of Hirt and Maurer [HM00], where the actual steps of the recursive composition are designing according to a logarithmic-depth formula that computes a threshold function using only constant fan-in threshold gates.

Using this approach, we simplified and improved upon previous results. In particular:

- We provided conceptually simple constructions of efficient protocols for MPC in the presence of an honest majority, as well as broadcast protocols from point-to-point channels and a 2-cast primitive.

- We obtained new results regarding MPC over blackbox groups[1] and other algebraic structures.

The above results rely on the following complexity-theoretic contributions, which may be of independent interest. In the following we say that a function f is an ϵ-approximation of the n/m-out-of-n threshold function, if for every string x, if the relative Hamming weight of x is greater than $1/m + \epsilon$, then $f(x) = 1$, and if the weight is smaller than $1/m - \epsilon$, then $f(x) = 0$.

- We showed that for every $j, k \in \mathbb{N}$ such that $m \stackrel{\text{def}}{=} \frac{k-1}{j-1}$ is an integer, there is an explicit ($\mathsf{poly}(n)$-time) construction of a logarithmic-depth formula that computes an $\Omega\left(\frac{1}{\sqrt{\log n}}\right)$ approximation of an (n/m)-out-of-n threshold function using only j-out-of-k threshold gates and no constants.

- For the special case of n-bit majority from 3-bit majority gates, a non-explicit construction follows from the work of Valiant [Val84]. For this special case, we provided an explicit construction with a better approximation than for the general threshold case (specifically, a $2^{-O(\sqrt{\log n})}$ approximation of the majority function), and also an *exact* explicit construction based on standard complexity-theoretic or cryptographic assumptions.

A.2 Circular Security of Bit-Encryption [Rot13]

A public-key encryption scheme is said to be circular secure if security is not violated even if the adversary is given a "circular" encryption in which the secret key is encrypted using the corresponding public-key.

Motivated by recent developments in fully homomorphic encryption, in [Rot13], we considered the folklore conjecture that every semantically-secure *bit-encryption* scheme is circular secure, or in other words, that every bit-encryption scheme remains secure even when the adversary is given encryptions of the individual bits of the private-key. We showed the following obstacles to proving this conjecture:

1. We constructed a public-key bit-encryption scheme that is plausibly semantically secure, but is not circular secure. The circular security attack manages to fully recover the private-key.

   The construction is based on an extension of the Symmetric External Diffie-Hellman assumption (SXDH) from bilinear groups, to ℓ-multilinear groups of order p where $\ell \geq c \cdot \log p$ for some $c > 1$.

[1] In an MPC protocol over a blackbox group, there is an underlying (possibly non-Abelian) group and the parties wish to securely compute a product of group elements held by the individual players. While it is possible in principle to emulate each algebraic operation by a sequence of Boolean operations, this is inefficient both in theory and in practice. This inefficiency can be avoided by designing protocols which make a *blackbox* (i.e., oracle) use of the underlying group.

While there do exist ℓ-multilinear groups (unconditionally), for $\ell \geq 3$ there were no known candidates for which the SXDH problem was believed to be hard.[2] Nevertheless, there is also no evidence that such groups do not exist. Our result shows that in order to prove the folklore conjecture, one must rule out the possibility that there exist ℓ-multilinear groups for which SXDH is hard.

2. We showed that the folklore conjecture cannot be proved using a black-box reduction. That is, there is no reduction of circular security of a bit-encryption scheme to semantic security of the very same scheme that uses both the encryption scheme and the adversary as black-boxes.

Both of our negative results extend also to the (seemingly) weaker conjecture that every CCA secure bit-encryption scheme is circular secure.

As a final contribution, we showed an equivalence between three seemingly distinct notions of circular security for public-key bit-encryption schemes. In particular, we give a general search to decision reduction that shows that an adversary that distinguishes between encryptions of the bits of the private-key and encryptions of zeros can be used to actually recover the private-key.

Follow-up Work: Realizing Multilinear Maps. Following the publication of [Rot13], Garg, Gentry and Halevi [GGH13] suggested the first concrete candidate for a multilinear map based on hard computational problems on lattices. Unfortunately, the parameters of the [GGH13] multilinear map do not suffice for an instantiation of our construction (i.e. $\ell = o(\log p)$). Additionally, in a very recent work Koppula *et al.* [KRW13] gave an alternate construction of a circular *insecure* bit-encryption scheme, based on indistinguishability obfuscation.

A.3 Enhancements of Trapdoor Permutations [GR13a]

Loosely speaking, the notion of trapdoor permutations refers to a collection of permutations that are easy to sample and have domains that are easy to sample from (when given the description of the permutation). The main requirements are that these permutations are easy to evaluate, easy to invert when given a suitable trapdoor, but hard to invert when only given the description of the permutation (but not the trapdoor).

Together with Goldreich [GR13a], we took a closer look at several enhancements of the notion of trapdoor permutations. Specifically, we considered the notions of *enhanced trapdoor permutation* [Gol04] and *doubly enhanced trapdoor permutation* [Gol09] as well as intermediate notions [Rot10]. These enhancements arose in the study of Oblivious Transfer and NIZK, but they address natural concerns that may arise also in other applications of trapdoor permutations. We clarified why these enhancements are needed in such applications, and showed that they actually suffice for these needs.

[2] This state of affairs has recently changed, see below.

A.4 Fast Pseudorandomness for Independence and Load Balancing [MRRR14]

Together with Meka, Reingold and Rothblum [MRRR14] we considered the question of constructing *computationally efficient* pseudorandom objects. We provided new constructions of several such fundamental pseudorandom objects. Loosely speaking, these constructions obtained exponential improvements in efficiency compared to previous constructions with comparable randomness complexity. Our measure of efficiency was the number of word operations, as captured by the well-established unit-cost word RAM model. Our main results were the following:

1. A family of $(1/n)$-almost $\log n$-wise independent Boolean hash functions with $O(\log n)$ description length (or seed length) and $O(\log \log n)$ operations per evaluation.

   Prior constructions with similar seed lengths required $\Theta(\log n)$ operations.

2. ε-biased sequences for $\varepsilon = 1/\mathsf{poly}(n)$ with seed length $O(\log n \log \log n)$ and $O((\log \log n)^2)$ operations (to evaluate an output bit or a block of up to $\log n$ consecutive bits).

   Prior constructions achieve $O(\log n)$ seed length, but require $\Theta(\log n)$ operations. This construction implies pseudorandom generators with similar efficiency that fool classes such as low-degree polynomials and read-once CNFs.

3. Hash functions for placing n balls in n bins such that with all but probability $1/n$ the maximal load is $O(\log n / \log \log n)$ (which is optimal), with seed-length $O(\log n \log \log n)$ and $O((\log \log n)^2)$ operations per evaluation.

   The previously known construction with similar seed length required $\Theta(\log n \log \log n)$ operations. Indeed, our construction is an efficient instantiation of that construction, due to Celis, Reingold, Segev and Wieder [CRSW13].

These constructions are all simultaneously within $\log \log n$ factors of the optimal seed length, and within $(\log \log n)^2$ factors of the optimal computational efficiency.

A.5 Pseudorandom Graphs in Data Structures [RRW14]

Together with Reingold and Wieder [RRW14], we proved that the hash functions required for several data structure applications could be instantiated using the hash functions of Celis *et al.*[CRSW13]. These functions simultaneously enjoy short description length as well as fast evaluation time. The applications we considered are: (1) Cuckoo Hashing, (2) Cuckoo Hashing with Stash and (3) the Power of Two Choices paradigm for load balancing. Our analysis relied on a notion of sparse pseudorandom graphs that are similar to random graphs in having no large connected component and no dense subgraph. Such graphs may be of independent interest. Relating pseudorandom graphs to the two-choice paradigm relies on a very simple new proof we give (at the price of somewhat worse parameters).

Bibliography

[ABOR00] William Aiello, Sandeep Bhatt, Rafail Ostrovsky, and S. Raj. Rajagopalan. Fast verification of any remote procedure call: Short witness-indistinguishable one-round proofs for NP. In *ICALP: Annual International Colloquium on Automata, Languages and Programming*, 2000.

[AFNS06] Noga Alon, Eldar Fischer, Ilan Newman, and Asaf Shapira. A combinatorial characterization of the testable graph properties: it's all about regularity. In *STOC*, pages 251–260, 2006.

[AII06] David Avis, Hiroshi Imai, and Tsuyoshi Ito. On the relationship between convex bodies related to correlation experiments with dichotomic observables. *Journal of Physics A: Mathematical and General, 39(36)*, 39(36):11283, 2006.

[AIK06] Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. Cryptography in NC^0. *SIAM J. Comput.*, 36(4):845–888, 2006.

[AIK10] Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. From secrecy to soundness: Efficient verification via secure computation. In *ICALP (1)*, pages 152–163, 2010.

[AKNS00] Noga Alon, Michael Krivelevich, Ilan Newman, and Mario Szegedy. Regular languages are testable with a constant number of queries. *SIAM J. Comput.*, 30(6):1842–1862, 2000.

[AS03] Sanjeev Arora and Madhu Sudan. Improved low-degree testing and its applications. *Combinatorica*, 23(3):365–426, 2003.

[AW09] Scott Aaronson and Avi Wigderson. Algebrization: A new barrier in complexity theory. *ACM Trans. Comput. Theory*, 1:2:1–2:54, February 2009.

[Bab85] László Babai. Trading group theory for randomness. In *Proceedings of the seventeenth annual ACM symposium on Theory of computing*, pages 421–429. ACM, 1985.

[BBM11] Eric Blais, Joshua Brody, and Kevin Matulef. Property testing lower bounds via communication complexity. In *IEEE Conference on Computational Complexity*, pages 210–220, 2011.

[BCC88] Gilles Brassard, David Chaum, and Claude Crépeau. Minimum disclosure proofs of knowledge. *J. Comput. Syst. Sci.*, 37(2):156–189, 1988.

[BCCT12a] Nir Bitansky, Ran Canetti, Alessandro Chiesa, and Eran Tromer. From extractable collision resistance to succinct non-interactive arguments of knowledge, and back again. In *ITCS*, pages 326–349, 2012.

[BCCT12b] Nir Bitansky, Ran Canetti, Alessandro Chiesa, and Eran Tromer. Recursive composition and bootstrapping for snarks and proof-carrying data. *IACR Cryptology ePrint Archive*, 2012:95, 2012.

[BdW02] Harry Buhrman and Ronald de Wolf. Complexity measures and decision tree complexity: a survey. *Theor. Comput. Sci.*, 288(1):21–43, 2002.

[BFLS91] László Babai, Lance Fortnow, Leonid A. Levin, and Mario Szegedy. Checking computations in polylogarithmic time. In *STOC*, pages 21–31, 1991.

[BFS86] Laszlo Babai, Peter Frankl, and Janos Simon. Complexity classes in communication complexity theory. In *Proceedings of the 27th Annual Symposium on Foundations of Computer Science*, pages 337–347, Washington, DC, USA, 1986. IEEE Computer Society.

[BGW88] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In *STOC*, pages 1–10, 1988.

[Bla10] Eric Blais. Testing juntas: A brief survey. In Goldreich [Gol10a], pages 32–40.

[BLM+05] Jonathan Barrett, Noah Linden, Serge Massar, Stefano Pironio, Sandu Popescu, and David Roberts. Nonlocal correlations as an information-theoretic resource. *Physical Review A, 71(022101)*, 71(2):022101, 2005.

[Bol05] Beate Bollig. Property testing and the branching program size of boolean functions. In *Fundamentals of Computation Theory, 15th International Symposium, FCT 2005, Lübeck, Germany, August 17-20, 2005, Proceedings*, pages 258–269, 2005.

[BSGH+06] Eli Ben-Sasson, Oded Goldreich, Prahladh Harsha, Madhu Sudan, and Salil P. Vadhan. Robust PCPs of proximity, shorter PCPs, and applications to coding. *SIAM J. Comput.*, 36(4):889–974, 2006.

[BT04] Andrej Bogdanov and Luca Trevisan. Lower bounds for testing bipartiteness in dense graphs. In *IEEE Conference on Computational Complexity*, pages 75–81, 2004.

[BYKS01] Ziv Bar-Yossef, Ravi Kumar, and D. Sivakumar. Sampling algorithms: lower bounds and applications. In *STOC*, pages 266–275, 2001.

[CCD88] David Chaum, Claude Crépeau, and Ivan Damgård. Multiparty unconditionally secure protocols (extended abstract). In *STOC*, pages 11–19, 1988.

[CCGT13] Amit Chakrabarti, Graham Cormode, Navin Goyal, and Justin Thaler. Annotations for sparse data streams. *arXiv preprint arXiv:1304.3816*, 2013.

[CCM09] Amit Chakrabarti, Graham Cormode, and Andrew Mcgregor. Annotations in data streams. In *Proceedings of the 36th International Colloquium on Automata, Languages and Programming: Part I*, ICALP '09, pages 222–234, Berlin, Heidelberg, 2009. Springer-Verlag.

[CDI+13] Gil Cohen, Ivan Bjerre Damgård, Yuval Ishai, Jonas Kölker, Peter Bro Miltersen, Ran Raz, and Ron D. Rothblum. Efficient multiparty protocols via log-depth threshold formulae - (extended abstract). In *Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part II*, pages 185–202, 2013.

[CEG95] Ran Canetti, Guy Even, and Oded Goldreich. Lower bounds for sampling algorithms for estimating the average. *Inf. Process. Lett.*, 53(1):17–25, 1995.

[CGR+12] Artur Czumaj, Oded Goldreich, Dana Ron, C Seshadhri, Asaf Shapira, and Christian Sohler. Finding cycles and trees in sublinear time. *Random Structures & Algorithms*, 2012.

[CKLR11] Kai-Min Chung, Yael Tauman Kalai, Feng-Hao Liu, and Ran Raz. Memory delegation. In *CRYPTO*, pages 151–168, 2011.

[CKV10] Kai-Min Chung, Yael Tauman Kalai, and Salil P. Vadhan. Improved delegation of computation using fully homomorphic encryption. In *CRYPTO*, pages 483–501, 2010.

[CMT12] Graham Cormode, Michael Mitzenmacher, and Justin Thaler. Practical verified computation with streaming interactive proofs. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, pages 90–112. ACM, 2012.

[CMT13] Graham Cormode, Michael Mitzenmacher, and Justin Thaler. Streaming graph computations with a helpful advisor. *Algorithmica*, 65(2):409–442, 2013.

[CR11] Amit Chakrabarti and Oded Regev. An optimal lower bound on the communication complexity of gap-hamming-distance. In *Proceedings of the 43rd annual ACM symposium on Theory of computing*, STOC '11, pages 51–60, New York, NY, USA, 2011. ACM.

[CRSW13] L. Elisa Celis, Omer Reingold, Gil Segev, and Udi Wieder. Balls and bins: Smaller hash families and faster evaluation. *SIAM J. Comput.*, 42(3):1030–1050, 2013.

A. BIBLIOGRAPHY

[DFH12] Ivan Damgård, Sebastian Faust, and Carmit Hazay. Secure two-party computation with low communication. In *TCC*, pages 54–74, 2012.

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

[DR06] Irit Dinur and Omer Reingold. Assignment testers: Towards a combinatorial proof of the PCP theorem. *SIAM J. Comput.*, 36(4):975–1024, 2006.

[EKR04] Funda Ergün, Ravi Kumar, and Ronitt Rubinfeld. Fast approximate probabilistically checkable proofs. *Inf. Comput.*, 189(2):135–159, 2004.

[FGL14] Eldar Fischer, Yonatan Goldhirsh, and Oded Lachish. Partial tests, universal tests and decomposability. In *Innovations in Theoretical Computer Science, ITCS'14, Princeton, NJ, USA, January 12-14, 2014*, pages 483–500, 2014.

[FLM+12] Eldar Fischer, Oded Lachish, Arie Matsliah, Ilan Newman, and Orly Yahalom. On the query complexity of testing orientations for being eulerian. *ACM Transactions on Algorithms*, 8(2):15, 2012.

[FS86] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *CRYPTO*, pages 186–194, 1986.

[Gen09] Craig Gentry. Fully homomorphic encryption using ideal lattices. In *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009*, pages 169–178. ACM, 2009.

[GGH13] Sanjam Garg, Craig Gentry, and Shai Halevi. Candidate multilinear maps from ideal lattices. In *EUROCRYPT*, pages 1–17, 2013.

[GGK14] Oded Goldreich, Tom Gur, and Ilan Komargodski. Strong locally testable codes with relaxed local decoders. *Electronic Colloquium on Computational Complexity (ECCC)*, 21:25, 2014.

[GGP10] Rosario Gennaro, Craig Gentry, and Bryan Parno. Non-interactive verifiable computing: Outsourcing computation to untrusted workers. In *CRYPTO*, pages 465–482, 2010.

[GGPR12] Rosario Gennaro, Craig Gentry, Bryan Parno, and Mariana Raykova. Quadratic span programs and succinct NIZKs without PCPs. *IACR Cryptology ePrint Archive*, 2012:215, 2012.

[GGR98] Oded Goldreich, Shafi Goldwasser, and Dana Ron. Property testing and its connection to learning and approximation. *Journal of the ACM (JACM)*, 45(4):653–750, 1998.

[GGR15] Oded Goldreich, Tom Gur, and Ron D. Rothblum. Proofs of proximity for context-free languages and read-once branching programs. *Electronic Colloquium on Computational Complexity (ECCC)*, 22:24, 2015.

[GK92] Oded Goldreich and Hugo Krawczyk. Sparse pseudorandom distributions. *Random Struct. Algorithms*, 3(2):163–174, 1992.

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

[GLR11] Shafi Goldwasser, Huijia Lin, and Aviad Rubinstein. Delegation of computation without rejection problem from designated verifier cs-proofs. *IACR Cryptology ePrint Archive*, 2011:456, 2011.

[GMR89] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. *SIAM J. Comput.*, 18(1):186–208, 1989.

[GMW87] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In *STOC*, pages 218–229. ACM, 1987.

[Gol99] Oded Goldreich. *Modern cryptography, probabilistic proofs and pseudorandomness*, volume 17 of *Algorithms and Combinatorics*. Springer-Verlag, 1999.

[Gol04] Oded Goldreich. *Foundations of Cryptography: Volume 2: Basic Applications*. Cambridge University Press, 2004.

[Gol08] Oded Goldreich. *Computational complexity - a conceptual perspective*. Cambridge University Press, 2008.

[Gol09] Oded Goldreich. Basing non-interactive zero-knowledge on (enhanced) trapdoor permutations: The state of the art. *http://www.wisdom.weizmann.ac.il/~oded/PSBookFrag/nizk-tdp.ps*, November 2008 (revised October 2009).

[Gol10a] Oded Goldreich, editor. *Property Testing - Current Research and Surveys*, volume 6390 of *Lecture Notes in Computer Science*. Springer, 2010.

[Gol10b] Oded Goldreich. Short locally testable codes and proofs: A survey in two parts. In *Property Testing* [Gol10a], pages 65–104.

[Gol11] Oded Goldreich. Introduction to testing graph properties. In *Studies in Complexity and Cryptography. Miscellanea on the Interplay between Randomness and Computation*, pages 470–506. Springer, 2011.

[Gol14] Oded Goldreich. On multiple input problems in property testing. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2014, September 4-6, 2014, Barcelona, Spain*, pages 704–720, 2014.

[GR62] Seymour Ginsburg and H Gordon Rice. Two families of languages related to ALGOL. *Journal of the ACM (JACM)*, 9(3):350–371, 1962.

[GR99] Oded Goldreich and Dana Ron. A sublinear bipartiteness tester for bounded degree graphs. *Combinatorica*, 19(3):335–373, 1999.

[GR02] Oded Goldreich and Dana Ron. Property testing in bounded degree graphs. *Algorithmica*, 32(2):302–343, 2002.

[GR11] Oded Goldreich and Dana Ron. On proximity-oblivious testing. *SIAM Journal on Computing*, 40(2):534–566, 2011.

[GR13a] Oded Goldreich and Ron D. Rothblum. Enhancements of trapdoor permutations. *J. Cryptology*, 26(3):484–512, 2013.

[GR13b] Tom Gur and Ran Raz. Arthur-Merlin streaming complexity. In *Proceedings of the 40th International Colloquium on Automata, Languages and Programming (ICALP)*, 2013.

[GR13c] Tom Gur and Ron Rothblum. Non-interactive proofs of proximity. *Electronic Colloquium on Computational Complexity (ECCC)*, 20:78, 2013.

[GR14] Oded Goldreich and Dana Ron. On learning and testing dynamic environments. In *55th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2014, Philadelphia, PA, USA, October 18-21, 2014*, pages 336–343, 2014.

[GR15a] Oded Goldreich and Dana Ron. On sample-based testers. In *Proceedings of the 2015 Conference on Innovations in Theoretical Computer Science, ITCS 2015, Rehovot, Israel, January 11-13, 2015*, pages 337–345, 2015.

[GR15b] Tom Gur and Ron D. Rothblum. Non-interactive proofs of proximity. In *Proceedings of the 2015 Conference on Innovations in Theoretical Computer Science, ITCS 2015, Rehovot, Israel, January 11-13, 2015*, pages 133–142. ACM, 2015.

[Gro10] Jens Groth. Short pairing-based non-interactive zero-knowledge arguments. In *ASIACRYPT*, pages 321–340, 2010.

[GS86] Shafi Goldwasser and Michael Sipser. Private coins versus public coins in interactive proof systems. In *STOC*, pages 59–68, 1986.

[GS92] Peter Gemmell and Madhu Sudan. Highly resilient correctors for polynomials. *Inf. Process. Lett.*, 43(4):169–174, 1992.

[GS06] Oded Goldreich and Madhu Sudan. Locally testable codes and PCPs of almost-linear length. *J. ACM*, 53(4):558–655, 2006.

[GS10a] Dmitry Gavinsky and Alexander A Sherstov. A separation of NP and coNP in multiparty communication complexity. *arXiv preprint arXiv:1004.0817*, 2010.

[GS10b] Oded Goldreich and Or Sheffet. On the randomness complexity of property testing. *Computational Complexity*, 19(1):99–133, 2010.

[GS12] Oded Goldreich and Igor Shinkar. Two-sided error proximity oblivious testing - (extended abstract). In *APPROX-RANDOM*, pages 565–578, 2012.

[GS13] Lior Gishboliner and Asaf Shapira. Deterministic vs non-deterministic graph property testing. *Electronic Colloquium on Computational Complexity (ECCC)*, 20:59, 2013.

[GW11] Craig Gentry and Daniel Wichs. Separating succinct non-interactive arguments from all falsifiable assumptions. In *STOC*, pages 99–108, 2011.

[HLNT05] Shirley Halevy, Oded Lachish, Ilan Newman, and Dekel Tsur. Testing orientation properties. *Electronic Colloquium on Computational Complexity (ECCC)*, 2005.

[HLNT07] Shirley Halevy, Oded Lachish, Ilan Newman, and Dekel Tsur. Testing properties of constraint-graphs. In *IEEE Conference on Computational Complexity*, pages 264–277, 2007.

[HM00] Martin Hirt and Ueli M. Maurer. Player simulation and general adversary structures in perfect multiparty computation. *J. Cryptology*, 13(1):31–60, 2000.

[HMU06] John E. Hopcroft, Rajeev Motwani, and Jeffrey D. Ullman. *Introduction to Automata Theory, Languages, and Computation (3rd Edition)*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2006.

[Hol09] Thomas Holenstein. Parallel repetition: Simplification and the no-signaling case. *Theory of Computing*, 5(1):141–172, 2009.

[IKM09] Tsuyoshi Ito, Hirotada Kobayashi, and Keiji Matsumoto. Oracularization and two-prover one-round interactive proofs against nonlocal strategies. In *IEEE Conference on Computational Complexity*, pages 217–228, 2009.

[Ito10] Tsuyoshi Ito. Polynomial-space approximation of no-signaling provers. In *ICALP (1)*, pages 140–151, 2010.

[IV12] Tsuyoshi Ito and Thomas Vidick. A multi-prover interactive proof for NEXP sound against entangled provers. *CoRR*, abs/1207.0550, 2012.

[Kil92] Joe Kilian. A note on efficient zero-knowledge proofs and arguments (extended abstract). In *STOC*, pages 723–732, 1992.

[KKM+08] Julia Kempe, Hirotada Kobayashi, Keiji Matsumoto, Ben Toner, and Thomas Vidick. Entangled games are hard to approximate. In *FOCS*, pages 447–456, 2008.

[Kla03] Hartmut Klauck. Rectangle size bounds and threshold covers in communication complexity. In *Computational Complexity, 2003. Proceedings. 18th IEEE Annual Conference on*, pages 118–134. IEEE, 2003.

[Kla11] Hartmut Klauck. On Arthur Merlin games in communication complexity. In *Computational Complexity (CCC), 2011 IEEE 26th Annual Conference on*, pages 189–199. IEEE, 2011.

[KN97] Eyal Kushilevitz and Noam Nisan. *Communication complexity*. Cambridge University Press, 1997.

[KR09] Yael Tauman Kalai and Ran Raz. Probabilistically checkable arguments. In *CRYPTO*, pages 143–159, 2009.

[KR14] Yael Tauman Kalai and Ron D. Rothblum. Arguments of proximity. Manuscript, 2014.

[KRR13a] Yael Tauman Kalai, Ran Raz, and Ron D. Rothblum. Delegation for bounded space. In *STOC*, pages 565–574, 2013.

[KRR13b] Yael Tauman Kalai, Ran Raz, and Ron D. Rothblum. How to delegate computations: The power of no-signaling proofs. *Electronic Colloquium on Computational Complexity (ECCC)*, 20:183, 2013.

[KRR14] Yael Tauman Kalai, Ran Raz, and Ron D. Rothblum. How to delegate computations: the power of no-signaling proofs. In *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014*, pages 485–494, 2014.

[KRW13] Venkata Koppula, Kim Ramchen, and Brent Waters. Separations in circular security for arbitrary length key cycles. *IACR Cryptology ePrint Archive*, 2013:683, 2013.

[KS92] Bala Kalyanasundaram and Georg Schintger. The probabilistic communication complexity of set intersection. *SIAM Journal on Discrete Mathematics*, 5(4):545–557, 1992.

[KS08] Tali Kaufman and Madhu Sudan. Algebraic property testing: the role of invariance. In *Proceedings of the 40th annual ACM Symposium on Theory of Computing (STOC)*, pages 403–412. ACM, 2008.

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

[KT00] Jonathan Katz and Luca Trevisan. On the efficiency of local decoding procedures for error-correcting codes. In *STOC*, pages 80–86, 2000.

[KW88] Klaus Kriegel and Stephan Waack. Lower bounds on the complexity of real-time branching programs. *ITA*, 22(4):447–459, 1988.

[Lau83] Clemens Lautemann. BPP and the polynomial hierarchy. *Inf. Process. Lett.*, 17(4):215–217, 1983.

[Lev85] Leonid A. Levin. One-way functions and pseudorandom generators. In *STOC*, pages 363–365, 1985.

[LFKN92] Carsten Lund, Lance Fortnow, Howard J. Karloff, and Noam Nisan. Algebraic methods for interactive proof systems. *J. ACM*, 39(4):859–868, 1992.

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

[LSH65] Philip M. Lewis, Richard Edwin Stearns, and Juris Hartmanis. Memory bounds for recognition of context-free and context-sensitive languages. In *SWCT (FOCS)*, pages 191–202, 1965.

[LV12] László Lovász and Katalin Vesztergombi. Nondeterministic graph property testing. *arXiv preprint arXiv:1202.5337*, 2012.

[Mic94] Silvio Micali. CS proofs (extended abstracts). In *FOCS*, pages 436–453, 1994.

[MRRR14] Raghu Meka, Omer Reingold, Guy N. Rothblum, and Ron D. Rothblum. Fast pseudorandomness for independence and load balancing - (extended abstract). In *Automata, Languages, and Programming - 41st International Colloquium, ICALP 2014, Copenhagen, Denmark, July 8-11, 2014, Proceedings, Part I*, pages 859–870, 2014.

[Nao03] Moni Naor. On cryptographic assumptions and challenges. In *CRYPTO*, pages 96–109, 2003.

[New91] Ilan Newman. Private vs. common random bits in communication complexity. *Information processing letters*, 39(2):67–71, 1991.

[New02] Ilan Newman. Testing membership in languages that have small width branching programs. *SIAM Journal on Computing*, 31(5):1557–1570, 2002.

[New10] Ilan Newman. Property testing of massively parametrized problems - a survey. In *Property Testing*, pages 142–157, 2010.

[PR94] Sandu Popescu and Daniel Rohrlich. Quantum nonlocality as an axiom. *Foundations of Physics*, 24(3):379–385, 1994.

[PRR01] Michal Parnas, Dana Ron, and Ronitt Rubinfeld. Testing parenthesis languages. In *RANDOM-APPROX*, pages 261–272, 2001.

[PRV12] Bryan Parno, Mariana Raykova, and Vinod Vaikuntanathan. How to delegate and verify in public: Verifiable computation from attribute-based encryption. In *TCC*, pages 422–439, 2012.

[RAD78] Ronald L. Rivest, Leonard Adleman, and Michael L. Dertouzos. On data banks and privacy homomorphisms. In *Foundations of Secure Computation*, pages 169–180. Academic Press, 1978.

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

[Ron08] Dana Ron. Property testing: A learning theory perspective. *Foundations and Trends in Machine Learning*, 1(3):307–402, 2008.

[Ron09] Dana Ron. Algorithmic and analysis techniques in property testing. *Foundations and Trends in Theoretical Computer Science*, 5(2):73–205, 2009.

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

[Rot10] Ron Rothblum. A taxonomy of enhanced trapdoor permutations. *Electronic Colloquium on Computational Complexity (ECCC)*, 17:145, 2010.

[Rot13] Ron Rothblum. On the circular security of bit-encryption. In *TCC*, pages 579–598, 2013.

[RRW14] Omer Reingold, Ron D. Rothblum, and Udi Wieder. Pseudorandom graphs in data structures. In *Automata, Languages, and Programming - 41st International Colloquium, ICALP 2014, Copenhagen, Denmark, July 8-11, 2014, Proceedings, Part I*, pages 943–954, 2014.

[RS96] Ronitt Rubinfeld and Madhu Sudan. Robust characterizations of polynomials with applications to program testing. *SIAM J. Comput.*, 25(2):252–271, 1996.

[Ruz81] Walter L. Ruzzo. On uniform circuit complexity. *J. Comput. Syst. Sci.*, 22(3):365–383, 1981.

[RVW13] Guy N. Rothblum, Salil Vadhan, and Avi Wigderson. Interactive proofs of proximity: Delegating computation in sublinear time. In *Proceedings of the 45th annual ACM Symposium on Theory of Computing (STOC)*, 2013.

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

[She11] Alexander A. Sherstov. The communication complexity of gap hamming distance. *Electronic Colloquium on Computational Complexity (ECCC)*, 18:63, 2011.

[She12] Alexander A Sherstov. The multiparty communication complexity of set disjointness. In *Proceedings of the 44th symposium on Theory of Computing*, pages 525–548. ACM, 2012.

[Spi96] Daniel A. Spielman. Linear-time encodable and decodable error-correcting codes. *IEEE Transactions on Information Theory*, 42(6):1723–1731, 1996.

[Sud95] Madhu Sudan. *Efficient Checking of Polynomials and Proofs anf the Hardness of Approximation Problems*, volume 1001 of *Lecture Notes in Computer Science*. Springer, 1995.

[Sud00] Madhu Sudan. Probabilistically checkable proofs - lecture notes, 2000. Available at http://people.csail.mit.edu/madhu/pcp/pcp.ps.

[Tha13] Justin Thaler. Time-optimal interactive proofs for circuit evaluation. In *CRYPTO (2)*, pages 71–89, 2013.

[Ton09] Ben Toner. Monogamy of non-local quantum correlations. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Science*, 465(2101):59–69, 2009.

[Val84] Leslie G. Valiant. Short monotone formulae for the majority function. *J. Algorithms*, 5(3):363–366, 1984.

[Vid11] Thomas Vidick. A concentration inequality for the overlap of a vector on a large set, with application to the communication complexity of the gap-hamming-distance problem. *Electronic Colloquium on Computational Complexity (ECCC)*, 18:51, 2011.

[Vid13] Thomas Vidick. Three-player entangled xor games are np-hard to approximate. In *FOCS*, 2013.

[VSBW13] Victor Vu, Srinath T. V. Setty, Andrew J. Blumberg, and Michael Walfish. A hybrid architecture for interactive verifiable computation. In *IEEE Symposium on Security and Privacy*, pages 223–237, 2013.

[Yao82] Andrew Chi-Chih Yao. Protocols for secure computations (extended abstract). In *FOCS*, pages 160–164, 1982.