Electronic Colloquium on Computational Complexity, Report No. 78 (2013)

Non-Interactive Proofs of Proximity

Tom Gur Ron D. Rothblum

May 28, 2013

Abstract

We initiate 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 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 $\mathcal{NP}$ (or more accurately $\mathcal{MA}$) analogue of *property testing*.

We explore both the power and limitations of non-interactive proofs of proximity. We show that such proof-systems can be exponentially stronger than property testers, but are exponentially weaker than the *interactive* proofs of proximity studied by Rothblum, Vadhan and Wigderson (STOC 2013). In addition, we show 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.

ISSN 1433-8092

Contents

1 Introduction

Understanding the power and limitations of sublinear algorithms is a central question in complexity theory. 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 using proofs, we can indeed overcome inherent limitations of property testing.

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* ($\mathcal{IPP}$s).

1.1 The Notion of $\mathcal{MAP}$

In this work, we also consider proofs of proximity, but we 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 $\mathcal{NP}$ proof-system (whereas $\mathcal{IPP}$ is the property testing analogue of $\mathcal{IP}$). However, in contrast to polynomial-time algorithms, sublinear time algorithms inherently rely on *randomization*.[1] Since an $\mathcal{NP}$ proof-system in which the verifier is randomized is known as a *Merlin-Arthur* ($\mathcal{MA}$) proof-system, we call these sublinear non-interactive proof-systems *Merlin-Arthur proofs of proximity* or simply $\mathcal{MAP}$s.

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 solutions 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 test 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 $\mathcal{MAP}$ by the total amount of information available to the tester, namely, the sum of the $\mathcal{MAP}$s 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 we study the complexity of $\mathcal{MAP}$s in comparison to property testers and to the recently introduced $\mathcal{IPP}$s.

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 $\mathcal{MAP}$ for f can be used to verify the correctness of the computation

[1] It is not difficult to see that the sublinear time *deterministic* computation or even verification is limited to trivial properties (cf. [GS10b]).

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 $\mathcal{MAP}$ 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 (i.e., in polynomial-time).

Comparison with $\mathcal{PCP}$s of Proximity. $\mathcal{PCP}$s of proximity ($\mathcal{PCPP}$s), studied by Ben-Sasson *et al.* [BSGH+06] and by Dinur and Reingold [DR06] (called assignment testers therein) 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, $\mathcal{PCPP}$s differ from $\mathcal{MAP}$s in that the verifier is only given *implicit* (i.e., oracle) access to the proof, whereas in $\mathcal{MAP}$s, the verifier has free (*explicit*) access to the proof. Indeed, in contrast to $\mathcal{MAP}$s, the proof string in $\mathcal{PCPP}$s is typically of super-linear length (but only a small fraction of it is actually read). Thus, $\mathcal{PCPP}$s may be thought of as the $\mathcal{PCP}$ analogue of property testing, whereas $\mathcal{MAP}$s are the $\mathcal{NP}$ 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, leads to the taxonomy depicted in Table 1.1. Interestingly, the three other variants, corresponding to $\mathcal{NP}, \mathcal{PCP}$ and $\mathcal{PCPP}$, have all been well studied. Thus, we view the notion of $\mathcal{MAP}$s as completing this taxonomy of non-interactive proof-systems.

	Access to Proof	
Access to Main Input	**Explicit**	**Implicit (Oracle)**
Explicit	$\mathcal{NP}$ or $\mathcal{MA}$	$\mathcal{PCP}$
Implicit (Oracle)	$\mathcal{MAP}$ (**this work**)	$\mathcal{PCPP}$

Table 1: Taxonomy of non-interactive proof-systems.

1.2 The Power of $\mathcal{MAP}$

The first question that one might ask about the model of $\mathcal{MAP}$s 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 a 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 indeed possible:

Informal Theorem 1 (see Theorem 3.1)**.** *There exists a property that has an $\mathcal{MAP}$ 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 $\mathcal{MAP}$s we continue our study of $\mathcal{MAP}$s by asking how many queries can be saved by slightly increasing the length of the proof. The following result shows a property for which an (almost) tight full *multiplicative* trade-off exists between the number of queries and length of the proof:

Informal Theorem 2 (see Theorem 3.16)**.** *There exists a property Π such that for every $p \geq 1$, there is an $\mathcal{MAP}$ for Π that uses a proof of length p and makes $\frac{n^{0.999}}{p}$ queries.*

We remark that the trade-off in Informal Theorem 2 is (almost) tight, see Informal Theorem 5.

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 $\mathcal{MAP}$s.*

Informal Theorem 3 (see Theorem 4.3)**.** *Any two-sided error $\mathcal{MAP}$ 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 $\mathcal{MAP}$ that uses an empty proof, as an immediate corollary, we obtain a transformation from every two-sided error *property tester* into a one sided $\mathcal{MAP}$ 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 implies an exponential separation between $\mathcal{MAP}$s (with poly-logarithmically long proofs) and *one-sided error* property testing. We note that Informal Theorem 1 shows such a separation for the more general case of two-sided error.

Properties with/without distance. Many of the results discussed so far are based on 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 $\mathcal{MAP}$s. To show that the power of $\mathcal{MAP}$s is not limited to such problems, we also show an efficient $\mathcal{MAP}$ for approximating the Hamming weight of a given string — a property "without distance". We also give a lower bound on the $\mathcal{MAP}$ complexity of the problem. For more details, see Section 5.

$\mathcal{MAP}$s for graph properties. To see that $\mathcal{MAP}$s are also useful for testing graph properties, we note that testing bipartiteness (or more generally k-colorability) in the *dense* graph model can be tested using only $O(1/\varepsilon)$ queries (where ε represents the desired proximity to the object) provided 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 $\mathcal{PCPP}$) was suggested in [EKR04, BSGH+06].

$\mathcal{MAP}$s 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]).

1.3 The Limitations of $\mathcal{MAP}$

In the previous section, we described results that exhibit the power of $\mathcal{MAP}$s. But what are the limitations of $\mathcal{MAP}$s? We already described how long proofs (of size linear in the input-size) make $\mathcal{MAP}$s all powerful. Furthermore, Informal Theorem 1 shows that even a logarithmically long proof can be extremely useful. One might ask whether shorter, even constant-size proofs can be of use. Unfortunately, the answer is negative since an $\mathcal{MAP}$ 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 $\mathcal{MAP}$. In contrast, consider *proof-oblivious queries* $\mathcal{MAP}$s, which are $\mathcal{MAP}$s in which the distribution of queries is independent of the tested object. Such $\mathcal{MAP}$s can be viewed as a two step process in which the tester first (adaptively) queries the

[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.

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 $\mathcal{MAP}$s have proof oblivious queries.

Interestingly, the following result shows that $\mathcal{MAP}$s with *proof-oblivious queries* can provide at most a *quadratic* improvement over standard property testers.

Informal Theorem 4 (see Theorem 4.2)**.** *If a property* Π *has an* $\mathcal{MAP}$ *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 1, the restriction to *proof oblivious queries* is a necessary precondition for Informal Theorem 4 (and indeed, the $\mathcal{MAP}$ verifier of Informal Theorem 1 must make proof-dependent queries).

Recall that, on the one hand, by Informal Theorem 1 we know that $\mathcal{MAP}$s can be *exponentially* stronger than property testers, and, on the other hand, Informal Theorem 4 tells us that $\mathcal{MAP}$s that make *proof oblivious queries* can be at most *quadratically* stronger than property testers — we may be left with the hope that using $\mathcal{MAP}$s that make proof *dependent* queries it is *always* possible to obtain a super-quadratic (possibly even an exponential) improvement over property testers. Unfortunately, this is not the case: The following theorem (which complements Informal Theorem 2) shows a property that has a (roughly) quadratic lower bound on its $\mathcal{MAP}$ complexity, even if the queries are proof *dependent*.

Informal Theorem 5 (see Theorem 3.16)**.** *Any* $\mathcal{MAP}$ *for the property* Π *of Informal Theorem 2, which uses a proof of length* $p \geq 1$*, must make* $\tilde{\Omega}(n^{0.999}/p)$ *queries. (In particular, every property tester for* Π *must make* $\tilde{\Omega}(n^{0.99})$ *queries.)*

Having inspected the relationship between $\mathcal{MAP}$s and property testing, we proceed to consider the relationship between $\mathcal{MAP}$s and $\mathcal{IPP}$s. Recall that $\mathcal{MAP}$s are actually a special case of $\mathcal{IPP}$s in which the interaction is limited to a single message sent from the prover to the verifier. When comparing $\mathcal{MAP}$s and $\mathcal{IPP}$s it is natural to compare both the query complexity and the total amount of communication with the prover (which in the case of $\mathcal{MAP}$s is simply the length of the proof).

The following theorem shows that $\mathcal{IPP}$s are stronger than $\mathcal{MAP}$s not only syntactically but also in essence. We show that even 3-message $\mathcal{IPP}$s may have exponentially better query complexity than $\mathcal{MAP}$s (while using the same amount of communication). Moreover, we show that $\mathcal{IPP}$s with *poly-logarithmically* many messages (of poly-logarithmic length) can also have exponentially better communication.

Informal Theorem 6 (see Theorem 3.23 and Theorem 3.25)**.** *There exists a property that has* $\Theta\left(n^{0.499 \pm o(1)}\right)$ $\mathcal{MAP}$ *complexity but has:*

1. *A 3-message* $\mathcal{IPP}$ *that makes* $\mathsf{polylog}(n)$ *queries while using a total of* $n^{0.499+o(1)}$ *communication.*
2. *An* $\mathcal{IPP}$ *with only* $\mathsf{polylog}(n)$ *query and communication complexities but using a* poly-logarithmic *number of messages.*

1.4 Techniques

Many of our results (in particular Informal Theorems 2, 5 and 6) are based on a specific algebraic property, which we call *Sub-Tensor Sum* and denote by TensorSum. Let $\mathbb{F}$ be a finite field and let $H \subset \mathbb{F}$. 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.[3] That is, TensorSum contains all polynomials $P : \mathbb{F}^m \to \mathbb{F}$ of individual degree d such that

$$\sum_{x \in H^m} P(x) = 0.$$

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. $\boldsymbol{\mathcal{PT}}$: The query complexity of testing TensorSum (without a proof) is $\Theta(n^{0.999 \pm o(1)})$ queries.
2. $\boldsymbol{\mathcal{MAP}}$: The $\mathcal{MAP}$ complexity of TensorSum is $\Theta\left(n^{0.499 \pm o(1)}\right)$. Moreover, for every $p \geq 1$, the $\mathcal{MAP}$ query complexity of TensorSum with respect to proofs of length p is $\Theta\left(\frac{n^{0.999 \pm o(1)}}{p}\right)$.
3. $\boldsymbol{\mathcal{IPP}}$[3]: TensorSum has a 3-*message* $\mathcal{IPP}$ with query complexity $\mathsf{polylog}(n)$ and communication complexity $O\left(n^{0.499 + o(1)}\right)$.
4. $\boldsymbol{\mathcal{IPP}}$: TensorSum has an $\mathcal{IPP}$ with query and communication complexities $\mathsf{polylog}(n)$. However, in contrast to Item 3, this $\mathcal{IPP}$ 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 $\mathcal{MAP}$ 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.

[3] The choice of the constant 0 is arbitrary.

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 A.8).

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 $\mathcal{MAP}$ with proof and query complexity $O(\sqrt{n})$ (since $n = |\mathbb{F}|^2$). Using more sophisticated techniques in the same spirit, we obtain both $\mathcal{MAP}$ and $\mathcal{IPP}$ upper bounds for the TensorSum problem.[4]

In order to show an efficient $\mathcal{MAP}$ for properties that do not rely on such robust structure (which allows self-correction), we study the problem of approximating the Hamming weight of a given string. We design an $\mathcal{MAP}$ that allows the verifier to reduce the Hamming weight approximation problem to a concatenation-problem of properties that are easier to test (see [Gol13] for an extensive study of such problems). Then, we use a technique known as *precision sampling* (originally due to Levin [Lev85], see also [Gol13, Appendix A.2]), in order to solve the problem efficiently.

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 $\mathcal{MAP}$ lower bounds, we extend the [BBM11] framework. We show that lower bounds on the $\mathcal{MA}$ *communication complexity* of a communication complexity problem related to a property Π can be used to derive lower bounds on the $\mathcal{MAP}$ *complexity* of Π.

$\mathcal{MA}$ 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 $\mathcal{MAP}$ we are thus left with the task of showing lower bounds for related $\mathcal{MA}$ communication complexity problems. Fortunately, Klauck [Kla03] showed a strong lower bound for the set-disjointness problem, which we use in our reductions.

[4] We use TensorSum rather than MatrixSum because we do not know how to obtain an $\mathcal{IPP}$ nor a *full* trade-off between proof and query complexities for MatrixSum.

Additionally, in order to obtain another lower bound (specifically for the Hamming weight property), we extend a recent result of Gur and Raz [GR13] who give an $\mathcal{MA}$ communication complexity lower bound on the classical problem of *Gap Hamming Distance*.

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 relevant to our work is the recent work of Rothblum, Vadhan and Wigderson [RVW13] who initiated a systematic study of the power of this notion. Their main result is that all languages in $\mathcal{NC}$ 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 $\mathcal{NC}^1$ 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 $\mathcal{MA}$ is a special case, was initiated in the seminal works of Goldwasser, Micali and Rackoff [GMR89] and Babai [Bab85]. In the last decade, $\mathcal{MA}$ proof-systems were introduced for various computational models. There is a rich body of work in the literature addressing $\mathcal{MA}$ communication complexity protocols (e.g., [Kla03, GS10a, Kla11, She12]). Aaronson and Wigderson [AW09] used $\mathcal{MA}$ 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, GR13, 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.

Last, we note that Lovász and Vesztergombi [LV12] recently introduced a model of *non-deterministic property testing of graphs*. Their model is a form of $\mathcal{PCP}$ of proximity in which both the proof and verification procedure are restricted to be of a particular form.

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 2 we formally define $\mathcal{MAP}$s and property testers (which are essentially $\mathcal{MAP}$s with an empty string). In Section 3 we formally state and prove all of our separation results, whereas in Section 4 we prove our general transformation results. Finally, in Section 5 we show $\mathcal{MAP}$ results for some properties without distance. Important background material is provided in Appendix A.

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.

2.1 Merlin-Arthur Proofs of Proximity

We are now ready to define Merlin-Arthur proofs of proximity.

Definition 2.1. *A* Merlin-Arthur proof of proximity ($\mathcal{MAP}$) *for a property* $\Pi = \cup_{i\in\mathbb{N}}\Pi_n$ *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 $\mathcal{MAP}$ has a* one-sided error *and otherwise we say that it has* two-sided error*.*

We note that $\mathcal{MAP}$s can be viewed as a restricted form of the *interactive* proofs of proximity, studied by [RVW13] (see Appendix A.1 for the definition of $\mathcal{IPP}$).

An $\mathcal{MAP}$ 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 $\mathcal{MAP}$ 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.[5] If the $\mathcal{MAP}$ has query complexity q and proof complexity p, we say that it has complexity $t(n, \varepsilon) \stackrel{\text{def}}{=} q(n, \varepsilon) + p(n)$.

For every pair of functions $q : \mathbb{N} \times \mathbb{R}^+ \to \mathbb{N}$ and $p : \mathbb{N} \to \mathbb{N}$, we denote by $\mathcal{MAP}_2(p, q)$ (resp., $\mathcal{MAP}_1(p, q)$) the complexity class of all properties that have an $\mathcal{MAP}$ with proof complexity $O(p)$, query complexity $O(q)$ and two-sided error (resp., one-sided error). We also use $\mathcal{MAP}$ as a shorthand for the class $\mathcal{MAP}_2$.

Note that we defined $\mathcal{MAP}$s such that the proofs do not depend on the proximity parameter ε. Since our focus is on demonstrating the power of $\mathcal{MAP}$s (and our lower bounds refer to fixed valued of the proximity parameter), this makes our results stronger. Nevertheless, see Appendix C for a discussion of the alternate notion, in which the proof *may* depend on the proximity parameter.

Proof oblivious queries. An aspect of $\mathcal{MAP}$ proof-systems, which turns out to be very important, is whether the queries that the verifier makes depend on the proof. An $\mathcal{MAP}$ in which the queries *do not depend on the proof* may be thought of as the following two step process:

[5] 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 $p(n)$.

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 2.2. *An $\mathcal{MAP}$ 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 $\mathcal{MAP}$ 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.*

$\mathcal{MA}$ proximity-oblivious testing. We also present an $\mathcal{MA}$ 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* $\mathcal{MA}$ proximity-oblivious testers, and note that a *two-sided error* variant of $\mathcal{MA}$ proximity-oblivious testers can be defined similarly to [GS12].

Definition 2.3. *Let $\rho : (0,1] \to (0,1]$ be some increasing function. A (one-sided error)* $\mathcal{MA}$ 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 $\mathcal{MAP}$s presented in this work are based on corresponding $\mathcal{MA}$ proximity-oblivious testers. The most notable example is the $\mathcal{MAP}$ in Theorem 3.3.

2.2 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 $\mathcal{MAP}$s 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 $\mathcal{MAP}$s 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 $T^f(n, \varepsilon)$).

2.3 Property Testing

The standard definition of property testing may be derived from Definition 2.1 by restricting both the completeness and soundness conditions to hold when the proof length is fixed to 0. Hence, $\mathcal{MAP}$s are a strict syntactic generalization of property testers. We will always refer to a tester that uses a proof as an "$\mathcal{MAP}$ 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 $\mathcal{PT}_2(q)$ (resp., $\mathcal{PT}_1(q)$) the class $\mathcal{MAP}_2(0, q)$ (resp., $\mathcal{MAP}_1(0, q)$). We also use $\mathcal{PT}$ as a shorthand for the class $\mathcal{PT}_2$.

For a detailed introduction to property testing, see the surveys [Ron08, Ron09] and the collection [Gol10a].

3 Separation Results

In this section we explore the power of $\mathcal{MAP}$ verifiers in comparison to other types of testers, such as property testers and $\mathcal{IPP}$ verifiers and present properties that exhibit a separation between these different types of testers.

In Section 3.1 we show an exponential gap between the complexity of $\mathcal{PT}$ and $\mathcal{MAP}$. In Section 3.2 we show a problem that has an $\mathcal{MAP}$ with an (almost) tight multiplicative trade-off between the proof length and number of queries. In Section 3.3 we consider 3-*message* $\mathcal{IPP}$ verifiers and show that they may have exponentially smaller *query* complexity than $\mathcal{MAP}$ verifiers (when using a proof of similar length). Finally, in Section 3.4 we also show

an exponential gap between the total complexity (i.e., query plus proof/communication complexities) of $\mathcal{MAP}$ and general $\mathcal{IPP}$ (which uses a poly-logarithmic number of messages).

3.1 Exponential Separation between $\mathcal{PT}$ and $\mathcal{MAP}$

In this section we show an *exponential* separation between the power of property testing and $\mathcal{MAP}$. Roughly speaking, we show a property that requires roughly $n^{0.999}$ queries for every property tester but has an $\mathcal{MAP}$ that, while using a proof of only *logarithmic* length, requires only a *constant* number of queries. We prove three incomparable variants of this result.

Theorem 3.1. *For every constant $\alpha > 0$, there exists a property Π_α that has an $\mathcal{MAP}$ 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 $\mathcal{MAP}$ 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 3.1 that let us handle general values of ε. In Theorem 3.2 we do so but at the cost of increasing the $\mathcal{MAP}$ query complexity to depend poly-logarithmically on n. In Theorem 3.3 we maintain the $\mathsf{poly}(1/\varepsilon)$ query complexity, at the cost of having a smaller (yet still exponential) gap between the power of property testers and $\mathcal{MAP}$s.

Theorem 3.2. *For every constant $\alpha > 0$, there exists a property Π_α that has an $\mathcal{MAP}$ 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 $\mathcal{MAP}$ has one-sided error.*

Theorem 3.3. *There exists a universal constant $c \in (0,1)$ and a property Π that has an $\mathcal{MAP}$ 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 $\mathcal{MAP}$ has one-sided error.*[6]

Note that all of the above separation results refer to the general (i.e., two-sided error) classes $\mathcal{PT}_2$ and $\mathcal{MAP}_2$. As noted in the introduction, a more restricted separation between the *one-sided error* classes (i.e., between $\mathcal{PT}_1$ and $\mathcal{MAP}_1$) can be obtained by using Theorem 4.3.

3.1.1 Our Approach

The proof of Theorem 3.1 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

[6] We remark that the proof of Theorem 3.3 can be adapted to yield an $\mathcal{MA}$ *proximity-oblivious tester* (see Definition 2.3) for Π.

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 Appendix A.4.

As discussed in the introduction, the complexities of property testers and $\mathcal{MAP}$ verifiers with *proof oblivious queries* are polynomially related (see Theorem 4.2). Thus, in order to show an *exponential* separation between $\mathcal{PT}$ and $\mathcal{MAP}$, one has to use an $\mathcal{MAP}$ 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 $\mathcal{MAP}$ in which the queries are "strongly affected" by the proof;
2. Π is hard for property testers (and hence for $\mathcal{MAP}$s 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 Appendix A.4), it is easy to test this property using an $\mathcal{MAP}$ 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 $\mathcal{MAP}$ that we construct is proportional to the number of queries required by the LTC and LDC procedures.

It is well known that (a proper 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 3.2; namely, a property that has an $\mathcal{MAP}$ 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* $\mathcal{MAP}$ query complexity (as in Theorem 3.1), 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 A.5 for the formal definition.

Ben-Sasson *et al.* [BSGH+06] used $\mathcal{PCPP}$s 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 3.1. 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 3.3.

Organization. In Section 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 $\mathsf{EIM}_C \in \mathcal{MAP}\big(\log k, t_1(n/2) + t_2(n/2, \varepsilon/2)\big)$. In Section 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 3.1.4 we state the result of [BSGH+06] and derive Theorem 3.1, and in Section 3.1.5 we prove Theorem 3.2 using an appropriate instantiation of the Reed-Muller code. Lastly, in Section 3.1.6 we construct a $\mathsf{poly}(1/\varepsilon)$-LTC, which is also an $O(1)$-relaxed-LDC (with polynomial stretch), and prove Theorem 3.3.

3.1.2 An $\mathcal{MAP}$ Upper Bound for EIM

Lemma 3.4. *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, $\mathsf{EIM}_C \in \mathcal{MAP}_1\big(\log k, t_1(n/2) + t_2(n/2, \varepsilon/2)\big)$.*

Proof. We prove Lemma 3.4 by showing an $\mathcal{MAP}$ 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 Figure 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).

$\mathcal{MAP}$ 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 1: $\mathcal{MAP}$ for EIM_C

Since the code is a t_1-relaxed-LDC and a ${}_2t$-LTC, the query complexity of the $\mathcal{MAP}$ 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 $\mathcal{MAP}$ 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 $\mathcal{MAP}$ 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 $1/2$. □

3.1.3 A $\mathcal{PT}$ Lower Bound for EIM

Next, we show a that the query complexity of property testing the EIM property must be linear in k.

Lemma 3.5. *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 3.5 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 Appendix A.2 (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 3.6 (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,[7] and their goal is to decide whether $\psi(f,g) \in \Pi$ or $\psi(f,g)$ is ε-far from Π.[8] Next, we state the main lemma from [BBM11].

Lemma 3.7. *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 Appendix A.2).

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 Appendix A.2) 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 3.8 ([KS92])**.** *For every $n \in \mathbb{N}$,*

$$\mathsf{CC}(\mathsf{DISJ}_n) = \Omega(n).$$

Using the aforementioned results, we are ready to prove Lemma 3.5.

Proof of Lemma 3.5. 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

[7] More formally, the parties get as input strings that represent the truth table of the functions.

[8] 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 12.

(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 3.7) 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 3.8 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, $(\alpha, \beta) \in \mathsf{EIM}_c$. 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 3.8, we conclude that for every $\varepsilon > 0$,

$$\mathsf{PT}_\varepsilon(\mathsf{EIM}_c) \geq \frac{1}{2\log|\Sigma|}\mathsf{CC}\left(\mathcal{C}^{\mathsf{EIM}_C}_{\mathsf{Pair},\varepsilon}\right) \geq \frac{1}{2\log|\Sigma|}\mathsf{CC}(\mathsf{DISJ}_k) = \Omega(k).$$

□

3.1.4 Proof of Theorem 3.1

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 3.9 ([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 3.1 follows by combining Theorem 3.9 with Lemma 3.4 and Lemma 3.5.

3.1.5 Proof of Theorem 3.2

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 3.4 and Lemma 3.5, we prove Theorem 3.2.

Lemma 3.10. *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 A.9), and locally decoded using $O(m|H|)$ queries (see Theorem A.7). 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

$$\Pi_{\mathbb{F},d,m,H} = \left\{ (P,Q) : P, Q : \mathbb{F}^m \to \mathbb{F} \text{ have individual degree } d \text{ and } \sum_{x \in H^m} (P \cdot Q)(x) \neq 0 \right\}.$$

3.1.6 Proof of Theorem 3.3

In this section we prove Theorem 3.3 by showing an $O(1)$-relaxed-LDC that is also a $\mathsf{poly}(1/\varepsilon)$-LTC, for general values of ε, but with a polynomial stretch. We believe that the following theorem may be of independent interest.

Theorem 3.11. *There exists a binary code that is an $O(1)$-relaxed-LDC and a $\mathsf{poly}(1/\varepsilon)$-LTC with constant relative distance and stretch $n = \mathsf{poly}(k)$.*

Indeed, Theorem 3.3 follows by combining Theorem 3.11 with Lemma 3.4 and Lemma 3.5.

Proof of Theorem 3.11. Our approach is to show that the [BSGH+06] relaxed-LDC can actually be modified to also be an LTC. Since we only show a code with polynomial (rather than "almost linear") stretch, it suffices to consider the simpler quadratic stretch relaxed-LDC of [BSGH+06]. Recall that the construction in [BSGH+06] has the form $C'(x) = \big(x^t, C(x)^{t'}, \pi(x)\big)$, where C is a code with constant relative distance and constant rate, whereas t and t' are integers set such that the three parts of $C'(x)$ have equal length, and $\pi(x) = \pi_1(x), \ldots, \pi_k(x)$ is a sequence of k $\mathcal{PCPP}$s. The i^{th} $\mathcal{PCPP}$ (i.e., $\pi_i(x)$) refers to a statement of the form $(z_1, z_2) \in \{0,1\}^{|C(x)|+|C(x)|}$, and asserts that there exists $x \in \{0,1\}^k$ such that $z_1 = x_i^{|C(x)|}$ and $z_2 = C(x)$ (see [BSGH+06, Section 4.2] for further details).

Ben-Sasson *et al.* showed that the code C' is indeed a relaxed-LDC. The main problem that we encounter in turning C' into an LTC is that there is no apparent way for an LTC tester to detect errors (i.e., flipped bits) in the $\mathcal{PCPP}$ part of the codeword. Indeed, merely verifying that the statement holds using the $\mathcal{PCPP}$ proof string will not do, since in the case that all the errors are on the $\mathcal{PCPP}$ part, the statement is in fact *correct* and only the proof string contains errors. In such case there is no guarantee on the output of the $\mathcal{PCPP}$ verifier.

One possible way to overcome this difficulty is to use the *strong* $\mathcal{PCPP}$*s* of Goldreich and Sudan [GS06, Definition 5.7 (see revision dated May 2013)]. Recall that strong $\mathcal{PCPP}$s are $\mathcal{PCPP}$s in which every statement has a unique canonical proof such that a statement-proof pair is rejected with probability that is proportional to its distance from a true statement and corresponding canonical proof. Thus, even a true statement may be rejected when given a non-canonical proof.

Unfortunately, there is no known construction of general strong $\mathcal{PCPP}$s (e.g., for a $\mathcal{P}$-complete language). However, [GS06] constructed a special-purpose strong $\mathcal{PCPP}$ for a particular task. Using these special-purpose $\mathcal{PCPP}$s (and additional ideas from [BSGH+06] and [GS06]), we construct a strong $\mathcal{PCPP}$ for the particular statements in the [BSGH+06] relaxed-LDC. We remark that the strong $\mathcal{PCPP}$ that we use has polynomial length, which results in our code having polynomial stretch.

While the previous description provides the high-level view of our construction, we caution that the actual details are slightly more complex. The result stated in following lemma forms the core of our construction.

Lemma 3.12. *There exists constants $c_1, c_2, c_3 > 1$, a polynomial $n = \mathsf{poly}(k)$, and a code $C : \{0,1\}^k \to \{0,1\}^{c_2 n}$ with constant relative distance such that for every $i \in [k]$, there exists a function $\pi_i : \{0,1\}^k \to \{0,1\}^{c_3 n}$ such that the code $C_i : \{0,1\}^k \to \{0,1\}^{c_1 n + c_2 n + c_3 n}$ defined as $C_i(x) = (x_i^{c_1 n}, C(x), \pi_i(x))$ is a $\mathsf{poly}(1/\varepsilon)$-LTC.*

Lemma 3.12 is proved in Appendix B.2. The proof uses an extension of [GS06, Theorem 5.20] that was communicated to us by Oded Goldreich and appears in Appendix B.1.

Intuitively, Lemma 3.12 provides an LTC in which the i^{th} bit can be recovered (in the relaxed-LDC sense).[9] Using the particular form of the codes $\{C_i\}_{i\in[k]}$ of Lemma 3.12, we show that that the code $C'(x) \stackrel{\text{def}}{=} C_1(x), \ldots, C_k(x)$ (which has constant relative distance) is an LTC in which each one of the bits can be recovered (again, in the relaxed-LDC sense).

Proposition 3.13. *C' is an $O(1)$-relaxed-LDC.*

Proof. Observe that, up to a permutation of the indices, C' has the form

$$C'(x) = \left(x^{c_1 n}, C(x)^k, \pi(x)\right)$$

where $\pi(x) = \pi_1(x), \ldots, \pi_k(x)$. We proceed to show that the code C' has the same form as the quadratic relaxed-LDC code in [BSGH+06, Section 4.2], and therefore, is an $O(1)$-relaxed-LDC.

We first note that in contrast to the [BSGH+06] construction, the three parts of C' do not have the exact same length. However, their lengths are proportional (by a constant factor) and that the proof of [BSGH+06] can be easily extended to this case.

The key observation is that each $\pi_i(x)$ is actually a $\mathcal{PCPP}$ proof of membership to the set

$$S_i = \left\{(z_1, z_2) \in \{0,1\}^{c_1 n + c_2 n} \;:\; z_1 = x_i^{c_1 n} \text{ and } z_2 = C(x)\right\}$$

(as in [BSGH+06]). To see that the latter holds, consider the following $\mathcal{PCPP}$ verifier for membership in S_i. The $\mathcal{PCPP}$ verifier is given oracle access to a pair $(z_1, z_2) \in \{0,1\}^{c_1 n + c_2 n}$ and a proof string π. The verifier simply invokes the LTC procedure of C_i on the string (z_1, z_2, π), and outputs its result.

If $(z_1, z_2) \in S_i$, then there exists $x \in \{0,1\}^k$ such that $z_1 = x_i^{c_1 n}$ and $z_2 = C(x)$. Letting the proof string π equal $\pi_i(x)$, the string (z_1, z_2, π) is identical to $C'(x)$, and so, the LTC tester accepts with probability 1.

On the other hand, if (z_1, z_2) has constant distance from S_i, then for every string π, the string (z_1, z_2, π) has constant distance from the code C', and the LTC tester rejects with probability at least $1/2$. This ends the proof of Proposition 3.13. □

Proposition 3.14. *C' is a $\mathsf{poly}(1/\varepsilon)$-LTC.*

Here we provide a sketch of the proof. The full proof, which follows ideas from [GS06], is deferred to Appendix B.3.

Proof Sketch for Proposition 3.14. Consider the following codeword tester for C'. The tester is given oracle access to a string $\left((w_1, y_1, z_1), \ldots, (w_k, y_k, z_k)\right)$, where each $(w_i, y_i, z_i) \in \{0,1\}^{c_1 n + c_2 n + c_3 n}$, and operates as follows:

1. Selects at random $i \in_R [k]$ and runs the LTC tester of C_i on (w_i, y_i, z_i).

[9] Recall that a formal definition of relaxed-LDCs can be found in Appendix A.4.

2. Selects at random $i_1, i_2 \in_R [k]$ and $j \in_R [c_2 n]$ and checks that the j^{th} bit of y_{i_1} and y_{i_2} agree.

Intuitively, the first part of the test guarantees that most of the triplets (w_i, y_i, z_i) have the form $\left(x_i^{(i)}, C(x^{(i)}), \pi_i(x^{(i)})\right)$ for some $x^{(i)} \in \{0,1\}^k$, and the second part guarantees that all of the $x^{(i)}$'s are the same. For the detailed proof, see Appendix B.3. □

This completes the proof of Theorem 3.11. □

Remark 3.15. *The construction of Theorem 3.11 can be adapted to be an $\mathcal{MA}$* proximity-oblivious tester *(see Definition 2.3) by observing that the* LTC *tester can be adapted to be a* strong-LTC[10] *tester.*

3.2 Tradeoff between Query and Proof Complexity

In this section we show a property that has a multiplicative trade-off between proof and query complexities for $\mathcal{MAP}$ testing. We show a property that can be tested with a nearly smooth tradeoff between the proof and query complexities.

Theorem 3.16. *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 $\mathcal{MAP}$ 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 Appendix A.5 for the necessary background (e.g., the Schwartz-Zippel lemma and low degree testing). In fact, the proof of Theorem 3.16 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 and Section 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 3.17. *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 $\mathbb{F}$.

[10] A strong-LTC tester is not given the proximity parameter as input and is only required to reject strings that are ε-far from the code with probability $\Omega(1/\varepsilon)$ (see [GS06, Definition 2.2]).

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 3.2.1 we prove the upper bound and in Section 3.2.2 we prove the lower bound. Finally, in Section 3.2.3 we set the parameters for proving Theorem 3.16.

3.2.1 $\mathcal{MAP}$ Upper Bound for $\mathsf{TensorSum}$

We start by proving the following upper bound.

Lemma 3.18. *If $dm < |\mathbb{F}|/10$, then, for every $\ell \in \{0, \ldots, m\}$, the $\mathsf{TensorSum}_{\mathbb{F},m,d,H}$ property has an $\mathcal{MAP}$ 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 $\mathcal{MAP}$ 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 3.18. We prove the lemma by showing an $\mathcal{MAP}$ 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 $\mathcal{MAP}$ 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 Figure 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.

$\mathcal{MAP}$ 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 A.9) 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 A.7) 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 2: $\mathcal{MAP}$ 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 A.9), 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 $\sum_{x_{\ell+1},\ldots,x_m \in H} z_{r_1,\ldots,r_\ell,x_{\ell+1},\ldots,x_m}$. □

3.2.2 $\mathcal{MAP}$ Lower Bound for TensorSum

Next, we give an (almost) matching lower bound on the $\mathcal{MAP}$ complexity of *Sub-Tensor Sum*. Formally, we show

Lemma 3.19. *For every $\varepsilon \in (0, 1 - \frac{dm}{|\mathbb{F}|})$, if $d \geq 2(|H|-1)$, then every $\mathcal{MAP}$ 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 3.19 we obtain the following:[11]

Corollary 3.20. *For every $\varepsilon \in (0, 1 - \frac{dm}{|\mathbb{F}|})$, if $d \geq 2(|H|-1)$,*

$$\mathcal{PT}_\varepsilon(\mathsf{TensorSum}) = \Omega\left(\frac{|H|^m}{\log(|\mathbb{F}|)}\right).$$

In order to prove Lemma 3.19, we first extend the framework of [BBM11] from the property testing model to the $\mathcal{MAP}$ model. More specifically, we show a methodology for proving lower bounds on $\mathcal{MAP}$s via $\mathcal{MA}$ communication complexity lower bounds. We refer the reader to Appendix A.3 for background on $\mathcal{MA}$ communication complexity.

Let Π be a property and let ψ be a simple combining operator (see Definition 3.6). For every proximity parameter $\varepsilon > 0$, denote by $\mathcal{C}^{\Pi}_{\psi,\varepsilon}$ the communication complexity problem in

[11] The corollary can be derived by setting $p = 1$, and the fact that any property tester is an $\mathcal{MAP}$.

which Alice gets as input a function f and Bob gets as input a function g and they 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.[12] We prove the following lemma.

Lemma 3.21 ($\mathcal{MAP}$ lower bounds via $\mathcal{MA}$ communication complexity)**.** *For any* simple *combining operator ψ, any property Π and any proximity parameter $\varepsilon > 0$, if $\Pi \in \mathcal{MAP}(p,q)$, then $\mathcal{C}^{\Pi}_{\psi,\varepsilon}$ has an $\mathcal{MA}$ communication complexity protocol with a proof of length p and total communication $2q|\psi|$.*

Proof. Let V be an $\mathcal{MAP}$ verifier for Π with proof complexity p and query complexity q. We construct an $\mathcal{MA}$ 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 $\mathcal{MAP}$ with respect to $h \stackrel{\text{def}}{=} \psi(f,g)$. As their first step, Alice and Bob emulate the execution of the $\mathcal{MAP}$ protocol with respect to the proof string w using their common random string as the source of randomness (for the emulated verifier). Whenever the $\mathcal{MAP}$ 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 $\mathcal{MAP}$ 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 $\mathcal{MAP}$ 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 $\mathcal{MA}$ communication complexity of set-disjointness [Kla03], and use Lemma 3.21 to show a lower bound on the $\mathcal{MAP}$ complexity of the *Sub-Tensor Sum* property.

Theorem 3.22 ([Kla03])**.** *Every $\mathcal{MA}$ communication complexity protocol for DISJ_n with proof complexity p and communication complexity c satisfies $p \cdot c = \Omega(n)$.*

[12] When proving property testing lower bounds via standard (i.e., non-$\mathcal{MA}$) 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 3.5). In contrast, the definition of $\mathcal{MA}$ communication complexity is *asymmetrical*; therefore when using our extension of the framework to $\mathcal{MA}$ one must map YES-instances to YES-instances, and NO-instances to NO-instances.

Proof of Lemma 3.19. Denote $k = |H|^m$ and by $f \cdot g$ the function $h(x) \stackrel{\text{def}}{=} f(x) \cdot g(x)$. Let $\mathcal{C}_{\cdot,\varepsilon}^{\mathsf{TensorSum}}$ be the communication complexity problem wherein Alice gets a function $f : \mathbb{F}^m \to \mathbb{F}$, 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 3.22 we know that every $\mathcal{MA}$ 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 3.21 we know that if $\mathsf{TensorSum} \in \mathcal{MAP}(p, q)$, then $\mathsf{CC}(\mathcal{C}_{\cdot,\varepsilon}^{\mathsf{TensorSum}})$ has an $\mathcal{MA}$ 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 $\mathcal{MA}$ communication complexity). Toward this end, suppose that π is an $\mathcal{MA}$ communication complexity protocol for $\mathcal{C}_{\cdot,\varepsilon}^{\mathsf{TensorSum}}$ with proof complexity p and communication complexity c. We use π to construct an $\mathcal{MA}$ 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 $\mathcal{MA}$ 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 Appendix A.5), and since $\varepsilon < 1 - \frac{dm}{|\mathbb{F}|}$, the function $\hat{a} \cdot \hat{b}$ is at least ε-far from $\mathsf{TensorSum}$.

We conclude that every $\mathcal{MAP}$ verifier for $\mathsf{TensorSum}$ with q queries and p proof length must satisfy $q \cdot p \geq \Omega\left(\frac{k}{\log(|\mathbb{F}|)}\right)$. □

3.2.3 Proof of Theorem 3.16

In this section we complete the proof of Theorem 3.16, 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 $\mathcal{MAP}$ 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 $\mathsf{TensorSum}$ 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 3.19). 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 3.18 guarantees the existence of an $\mathcal{MAP}$ 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 3.18. We obtain an $\mathcal{MAP}$ 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 $\mathcal{MAP}$ is $\frac{n^{1-\alpha+o(1)}}{p} \cdot \mathsf{poly}(1/\varepsilon)$.

On the other hand, by Lemma 3.19, for every $\mathcal{MAP}$ for $\mathsf{TensorSum}$ 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 $\mathcal{MAP}$ vs. $\mathcal{IPP}[O(1)]$

In this section and the following one, we consider the power of $\mathcal{MAP}$ in comparison to the more general notion of $\mathcal{IPP}$ (for a formal definition of $\mathcal{IPP}$, see Appendix A.1.) Roughly speaking, in this section we show a property that requires $\sqrt{n}$ queries by an $\mathcal{MAP}$ verifier that uses a proof of length $\sqrt{n}$ but requires only $\mathsf{polylog}(n)$ queries by an $\mathcal{IPP}[3]$ verifier (i.e., an $\mathcal{IPP}$ with only 3-messages) that also uses a proof of length $\sqrt{n}$.

Theorem 3.23. *For every $\alpha > 0$, there exists a property Π_α such that:*

1. *The $\mathcal{MAP}$ complexity of Π_α is $\tilde{\Omega}\left(n^{1/2-\alpha}\right)$; and*
2. *There is an $\mathcal{IPP}[3]$ 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 $\mathsf{TensorSum}$ property (introduced in Section 3.2). Note that the first part of Theorem 3.23 was already shown in Theorem 3.16, and so, to prove Theorem 3.23, what remains to be shown is that $\mathsf{TensorSum}$ can be tested by a 3-message $\mathcal{IPP}$ verifier that uses roughly $\sqrt{n}$ communication and $\mathsf{polylog}(n)$ queries.

Lemma 3.24. *If $dm < |\mathbb{F}|/10$, then there is a 3-message $\mathcal{IPP}$ 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 3.23 follows from Lemma 3.24 (and Lemma 3.19) by setting the parameters $\mathbb{F}, m, d, H$ as in Section 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 3.24

Proof of Lemma 3.24. The first part of the protocol closely resembles the $\mathcal{MAP}$ that was presented in Lemma 3.18. 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 3.18, 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 3.2, 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 $\mathcal{IPP}[3]$ protocol, in which the prover is denoted by $\mathcal{P}$ and the verifier is denoted by $\mathcal{V}$, is described in Figure 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.

Completeness. If $P \in \mathsf{TensorSum}$, 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 $\mathsf{TensorSum}$. 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 3.24.1. *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 $\mathsf{TensorSum}$. 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 3.24.2. *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

$$\tilde{W}(s_{\ell+1}, \ldots, s_m) \neq P'(r_1, \ldots, r_\ell, s_{\ell+1}, \ldots, s_m).$$

$\mathcal{IPP}[3]$ 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 A.9) 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 A.7) with soundness error $1/10$ and rejects if $z_{r_1, \ldots, r_\ell, s_{\ell+1}, \ldots, m} \neq W(s_{\ell+1}, \ldots, s_m)$.
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: $\mathcal{IPP}[3]$ for TensorSum

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 3.24.2, 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 3.24.1 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 3.24. □

3.4 Exponential Separation between $\mathcal{MAP}$ and $\mathcal{IPP}$

In this section we show an exponential separation between $\mathcal{MAP}$ and general $\mathcal{IPP}$. Namely, we show a property that has $\mathcal{MAP}$ complexity roughly $\sqrt{n}$ but has $\mathcal{IPP}$ complexity $\mathsf{polylog}(n)$. In contrast to the $\mathcal{IPP}$ of Section 3.3 (which used $O(1)$ messages) here we use an $\mathcal{IPP}$ with *poly-logarithmically* many messages.

Theorem 3.25. *For every $\alpha > 0$, there exists a property Π_α such that:*

1. *The $\mathcal{MAP}$ complexity of Π_α is $\tilde{\Omega}\left(n^{1/2-\alpha} \cdot \mathsf{poly}(1/\varepsilon)\right)$; and*
2. *Π_α has an $\mathcal{IPP}$ with query complexity $\mathsf{polylog}(n) \cdot \mathsf{poly}(1/\varepsilon)$ and communication complexity $\mathsf{polylog}(n)$.*

Moreover, the $\mathcal{PT}$ complexity of Π_α is $\tilde{\Theta}(n^{1-\alpha})$.

To prove Theorem 3.25, we yet again use the $\mathsf{TensorSum}$ problem. The first part of the theorem follows directly from Theorem 3.16 and the query complexity of property testers (without a proof) is implied by Corollary 3.20.[13] Thus, to prove the theorem, all that remains is to show an $\mathcal{IPP}$ protocol for $\mathsf{TensorSum}$.

Lemma 3.26. *If $d \cdot m < \mathbb{F}/10$, then there exists an m-round $\mathcal{IPP}$ 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 3.26 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

[13] 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.

and direct access to P.[14] 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 $\mathcal{IPP}$ protocol is described in Figure 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 Appendix A.6.)

$\mathcal{IPP}$ 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 A.9) 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 A.7) with soundness error 0.1.
5. $\mathcal{V}$ verifies that $z^* = \nu_m$.

Figure 4: $\mathcal{IPP}$ for $\mathsf{TensorSum}_{m,d,\mathbb{F},S,c}$

We note that during the run of the $\mathcal{IPP}$ the prover sends m degree d univariate polynomial, and the verifier sends m elements in $\mathbb{F}$. Thus, the total communication complexity of the $\mathcal{IPP}$ 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] An additional minor difference is that in the [LFKN92] protocol the set H is fixed to $\{0, 1\}$, but this is common in the $\mathcal{PCP}$ literature (most notably in [BFLS91]).

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 3.26.1. *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 3.26.2. *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 3.26.3 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 3.26.1, 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 3.26.3. *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 3.26. □

4 General Transformations

In this section we show general transformations on $\mathcal{MAP}$ proof-systems. In Section 4.1 we show general transformations from $\mathcal{MAP}$s with restricted proofs into $\mathcal{PT}$. In Section 4.2 we show a general transformation from $\mathcal{MAP}$s that have two-sided error into $\mathcal{MAP}$s that have one-sided error.

4.1 From $\mathcal{MAP}$ to $\mathcal{PT}$

In this section we show that $\mathcal{MAP}$s with restricted proofs can be emulated by property testers. We show two such results. Theorem 4.1 shows that every $\mathcal{MAP}$ that uses a *very short* proof can be emulated by a property tester, and Theorem 4.2 shows that even $\mathcal{MAP}$s with long proofs *in which the verifier's queries are proof oblivious* (see Definition 2.2) can also be emulated. We note that in both constructions the tester may be inefficient in terms of *computational* complexity (even if the original $\mathcal{MAP}$ tester can be implemented efficiently).

Theorem 4.1. *If the property Π has an $\mathcal{MAP}$ tester 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 $\mathcal{MAP}$ tester has one-sided error, then the resulting property tester has one-sided error.*

Proof. Let V be an $\mathcal{MAP}$ 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 $\mathcal{MAP}$ 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 4.1, follows by noting that both the parallel repetition and proof enumeration steps preserve one-sided error. □

The tester of Theorem 4.1 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 $\mathcal{MAP}$ verifier generates (heavily) depends on the proof, then the only guarantee that we have is that the $\mathcal{MAP}$ verifier will be correct when emulated with a distribution of queries that matches the specific good proof.[15] Hence, we may indeed have to generate a different sequence of queries for every possible proof string.

However, as proved in the following theorem, if the tester makes *proof oblivious queries* (see Definition 2.2), 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 4.2. *If the property* Π *has an* $\mathcal{MAP}$ *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* $\mathcal{MAP}$ *verifier has one-sided error, then the resulting property tester has one-sided error.*

Proof. Let V be an $\mathcal{MAP}$ verifier for Π with query complexity q and proof complexity p, and let V' be exactly as in the proof of Theorem 4.1 (i.e., an $\mathcal{MAP}$ 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 4.1. 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

[15] For an example of such $\mathcal{MAP}$s, see Theorem 3.1 and Theorem 4.3.

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 4.1, 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 $2^p \cdot 2^{-(p+2)} < 1/3$. □

4.2 From Two-Sided Error $\mathcal{MAP}$ to One-Sided Error $\mathcal{MAP}$

In this section we show a general result transforming any $\mathcal{MAP}$ (which may have two-sided error) into an $\mathcal{MAP}$ 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 $\mathcal{BPP}$ is contained the polynomial hierarchy coupled with the observation that $\mathcal{MAP}$s 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 4.3. *Let Π be a property of functions $f_n : D_n \to R_n$, where $|R_n| \leq \exp\big(\mathsf{poly}(n)\big)$. If Π has a* two-sided error *$\mathcal{MAP}$ with q queries and a proof of length p, then Π has a* one-sided error *$\mathcal{MAP}$ 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 $|R_n| > \exp(\mathsf{poly}(n))$ seem quite pathological. Before proceeding to the proof of Theorem 4.3, 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 $\mathcal{MAP}$ (denoted $\mathcal{MAP}_1$).

Corollary 4.4. *For every function $q : \mathbb{N} \times \mathbb{R}^+ \to \mathbb{N}$ it holds that:*

$$\mathcal{PT}(q) \subseteq \mathcal{MAP}_1(\mathsf{polylog}(n), q \cdot \mathsf{polylog}(n)).$$

The proof of Theorem 4.3 is based on two lemmas. The first, Lemma 4.5, shows that a two-sided error $\mathcal{MAP}$ verifier that has low *randomness complexity*, can be transformed into a one-sided error $\mathcal{MAP}$. The proof of this lemma is based on the technique of Lautemann [Lau83]. The second lemma (Lemma 4.6) shows that the Goldreich-Sheffet [GS10b] technique for reducing the randomness of *property testers* can also be used to reduce the randomness of $\mathcal{MAP}$ verifiers.

Lemma 4.5. *If the property Π has a* two-sided *$\mathcal{MAP}$ verifier that makes q queries, uses a proof of length p, and has randomness complexity r, then Π has a* one-sided *$\mathcal{MAP}$ 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 $\mathcal{MAP}$. 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 $\mathcal{MAP}$ 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* $\mathcal{MAP}$ verifier $V_{(1)}$ for Π.

Let $V_{(2)'}$ be the two-sided error $\mathcal{MAP}$ 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 $\mathcal{MAP}$ 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,

$$\Pr_{s_1,\ldots,s_t}\left[\exists s \text{ s.t. } \forall i \in [t],\, V^f_{(2)'}(w; s \oplus s_i) = 0\right] \leq \sum_s \Pr_{s_1,\ldots,s_t}\left[\forall i \in [t],\, V^f_{(2)'}(w; s \oplus s_i) = 0\right], \tag{4.1}$$

but since the s_i's are independent, for every $s \in \{0,1\}^{r'}$,

$$\Pr_{s_1,\ldots,s_t}\left[\forall i \in [t],\, V^f_{(2)'}(w; s \oplus s_i) = 0\right] = \prod_{i=1}^{t} \Pr_{s_i}\left[V^f_{(2)'}(w; s \oplus s_i) = 0\right] \leq \delta^t. \tag{4.2}$$

Combining Equations (4.1) and (4.2) 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 4.6. *Let Π be a property of functions $f_n : D_n \to R_n$, where $|R_n| \leq \exp\left(\mathsf{poly}(n)\right)$. If Π has an $\mathcal{MAP}$ verifier that makes q queries, uses a proof of length p, and has randomness complexity r, then Π has an $\mathcal{MAP}$ 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 $|R_n| = \exp\left(\mathsf{poly}(n)\right)$ (and where $D_n = [n]$, cf. Section 2), and let V be the $\mathcal{MAP}$ 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 $\mathcal{MAP}$ 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 1.2), that the domain size is n and that $|R| \leq \exp(\mathsf{poly}(n))$ (by the hypothesis). □

Theorem 4.3 follows by applying the randomness reducing transformation of Lemma 4.6, and then applying Lemma 4.5 to the resulting $\mathcal{MAP}$ verifier.

5 $\mathcal{MAP}$s for Properties without Distance

All of the properties studied in this work so far are properties of low-degree polynomials and error-correcting codes. The $\mathcal{MAP}$s that we have shown crucially relied on the fact that these properties have *distance* (i.e., properties wherein each two objects that have the property are far from each other), and moreover, they allow for a local form of self-correction.[16]

To show that the usefulness of $\mathcal{MAP}$s is not limited to properties with distance, we consider a property *without* distance (and therefore also "without self-correctability"), and show that one can design an efficient $\mathcal{MAP}$ for it, which surpasses the limitations of standard property testers. Specifically, we consider the problem of approximating the Hamming weight of a given string. This property is without distance since, for example, there are pairs of strings at distance 2 that have the exact same Hamming weight. We complement our $\mathcal{MAP}$ for the approximate Hamming weight problem with a (non-tight) *lower bound* on its $\mathcal{MAP}$ complexity. We leave the question of resolving the gap between the upper and lower bounds to future work.

5.1 Approximate Hamming Weight

We consider the problem of deciding whether a given string $x \in \{0,1\}^n$ has Hamming weight approximately w. That is, for a function $w : \mathbb{N} \to \mathbb{N}$, we are interested in testing proximity to the property $\mathsf{Hamming}_w$ that contains all n-bit strings x of Hamming weight $w(n)$. Thus, given $x \in \{0,1\}^n$ and a proximity parameter $\varepsilon > 0$, the tester should accept if $\mathsf{wt}(x) = w$ and reject if $\mathsf{wt}(x) \notin [w - \varepsilon n, w + \varepsilon n]$.

[16] 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.

We first note that by well-known sampling lower bounds (see, e.g., [Gol11, Theorem 2.1] or [BYKS01, Theorem 15]), the query complexity of any property tester (which does not use a proof) is $\Omega\left(\min\left(n, \varepsilon^{-2}\right)\right)$. Hence, our goal is to use $\mathcal{MAP}$s in order to bypass this lower bound. We remark that $\mathsf{Hamming}_w$ was already studied by [RVW13] who showed a multiple-message $\mathcal{IPP}$ with complexity $\tilde{O}\left(\varepsilon^{-1}\right)$ and a 2-message $\mathcal{IPP}$ with complexity $\tilde{O}\left(n^{\frac{1}{3}} \cdot \varepsilon^{-\frac{2}{3}}\right)$. For $\varepsilon = 1/\sqrt{n}$, this gives sublinear complexity of $\tilde{O}\left(n^{2/3}\right)$, whereas property testing requires $\Omega(n)$ queries. Using a technique known as precision sampling (originating in Levin [Lev85, last paragraph of Section 9], see also [Gol13, Appendix A.2]), we show that the [RVW13] 2-message $\mathcal{IPP}$ can be transformed into an $\mathcal{MAP}$ (i.e., a 1-message $\mathcal{IPP}$), while essentially preserving its complexity.[17] 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 $\mathcal{MAP}$ for $\mathsf{Hamming}$ 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 $\mathcal{MAP}$ is more efficient than the best possible property tester (which does not use a proof) for $\mathsf{Hamming}$. A comparison of the efficiency of our $\mathcal{MAP}$ versus standard property testers, for different values of α, is provided in Table 2.

		$\mathcal{MAP}$	
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 2: The complexity of testing $\mathsf{Hamming}$ for different values of α.

Before we proceed, we note that we actually prove a stronger result. Namely, we show that for every $k \in \mathbb{N}$ there is an $\mathcal{MAP}$ for $\mathsf{Hamming}$ 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

[17] We note that a similar $\mathcal{MAP}$ protocol for approximating the Hamming distance was also discovered independently by (Guy) Rothblum *et al.* following the initial publication of [RVW13].

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 $\mathcal{MAP}$, we consider a slight relaxation of our definition of $\mathcal{MAP}$ (Definition 2.1) by allowing the proof of proximity to depend on the proximity parameter. Thus, we define an $\mathcal{MAP}'$ as an $\mathcal{MAP}$ wherein both the contents of the proof of proximity, *and its length* may depend on the proximity parameter. With this relaxation, we can set $k = n^{\frac{1}{3}} \cdot \varepsilon^{-\frac{2}{3}}$ to obtain an $\mathcal{MAP}'$ with (total) complexity $\tilde{O}\left(n^{\frac{1}{3}} \cdot \varepsilon^{-\frac{2}{3}}\right)$. See Appendix C for further discussion of $\mathcal{MAP}'$.

We complement the foregoing upper bound by showing a *lower bound* on the $\mathcal{MAP}$ complexity of Hamming. Specifically, we show that every $\mathcal{MAP}$ for Hamming that uses a proof of length $p \geq 1$ must use $\Omega\left(\frac{\min(n,\varepsilon^{-2})}{p}\right)$ queries. As noted above, 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 closing this gap for future work.

Relation to TensorSum. The Hamming problem is loosely related to the *Sub-Tensor Sum problem* (see Section 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.

5.1.1 Upper Bound

The following $\mathcal{MAP}$ for Hamming depends on a parameter k that offers a trade-off between the proof and the query complexities.

Theorem 5.1. *For every $k \stackrel{\text{def}}{=} k(n,\varepsilon)$ and proximity parameter $\varepsilon \stackrel{\text{def}}{=} \varepsilon(n) > 0$, the property $\mathsf{Hamming}_w$ has a (two-sided error) $\mathcal{MAP}'$ verifier that uses a proof of length $k \cdot \log n$ and $\tilde{O}\left(\min\left(\sqrt{n/k} \cdot \varepsilon^{-1}, n\right)\right)$ queries. Furthermore, if k does not depend on ε, then the protocol is an $\mathcal{MAP}$ (rather than an $\mathcal{MAP}'$).*

We remark that by applying Theorem 4.3 we can (somewhat surprisingly) construct a *one-sided error* $\mathcal{MAP}$ with proof complexity $O(k \log n + \mathsf{polylog} n)$ and query complexity $\tilde{O}\left(\min\left(\sqrt{n/k} \cdot \varepsilon^{-1}, n\right)\right)$. In contrast, the query complexity of every one-sided error property tester for $\mathsf{Hamming}_w$ (without a proof) is linear in the input size.

Proof of Theorem 5.1. The key idea is to use the proof in order to reduce the problem of verifying the Hamming weight of a string x, to the concatenation problem of verifying the Hamming weight of numerous "short" substrings of x. The proof will (allegedly) provide the

Hamming weight of each of the k substrings, and the verification will amount to checking that x is close to a string having substrings that fit the provided weights.

More specifically, we view the string x as a $k \times \ell$ matrix (where $\ell \stackrel{\text{def}}{=} n/k$), and set the proof to be the Hamming weight of each one of the k rows. The verifier, given this alleged proof, checks that indeed the sum of the provided weights equals w, and is then left with the task of ascertaining that the claimed Hamming weights of the rows is (approximately) consistent with x.

Toward this end, we note that given an input that is far from having the claimed Hamming weight, the difference in the weight can be either "spread" between all of the rows, or "concentrated" on a few rows (or anything in between). The main idea is that if the excess/deficiency of weight is "concentrated", then the deviation in these rows must be large, and so, we can detect the deviation in a particular row by sampling only relatively few bits from that row. Since we only read a few bits for this test, we can afford to run it on many rows (thereby increasing our chance of catching a "heavy" row). On the other hand, if the excess/deficiency of weight is "spread" among the rows, then it suffices to examine only a few rows, but for each such row to look at many of its bits. In the latter case, it is actually beneficial to read each such row entirely, rather than to use naive sampling. Indeed, if the excess/deficiency is well-spread, the accuracy that we require of every row test is below $1/\sqrt{\ell}$, in which case it is cheaper to simply read the entire row.

Since the tester does not know whether it is in one of the extreme situations or anywhere in between, ideally we would have liked to consider all of the possible distributions of the excess/deficiency. However, since the former is too costly, we use the precision sampling technique, which allows us to deal with all of the possible distributions of the excess/deficiency economically (specifically, by considering only a logarithmic number of representative distributions). The protocol is formally described in Figure 5.

For every iteration $i \in [\log n]$, the number of queries made in the i^{th} iteration is at most:

$$\frac{k}{2^i} \cdot \min\left(\ell, \left(\frac{2^{i+1}\log(\varepsilon n)}{\varepsilon k}\right)^2 \cdot \log n\right) = \min\left(\frac{n}{2^i}, O\left(\frac{2^i \log^3 n}{\varepsilon^2 k}\right)\right).$$

Thus, by a straightforward calculation, if $k = \Omega\left(\frac{1}{\varepsilon^2 n} \cdot \log^3 n\right)$, then the total number of queries is at most $O\left(\sqrt{n/k} \cdot \varepsilon^{-1} \cdot \log^{5/2} n\right)$. Otherwise (i.e., if k is very small), the total number of queries is exactly n (since the verifier simply reads the entire string).

Completeness. Suppose that $\mathsf{wt}(x) = w$. Then, the tests that reads the entire row (i.e., step 2(b)) passes with probability 1, whereas for each one of the second type of tests (i.e., step 2(c)), the test passes with probability at least $1 - 1/\mathsf{poly}(n)$. Thus, by the union bound, the verifier accepts with probability at least $2/3$.

$\mathcal{MAP}$ for $\mathsf{Hamming}_w$ parameterized by k

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 $k \times \ell$ matrix where $\ell \stackrel{\text{def}}{=} \frac{n}{k}$. We denote by $x_{i,j}$ the $(i,j)^{\text{th}}$ entry of x, viewed as such a matrix.
- The proof consists of the sums of the weights of each of the rows of the matrix; namely, the values $(w_1, \ldots, w_k)$, where $w_i \stackrel{\text{def}}{=} \sum_{j=1}^{\ell} x_{i,j}$ for every $i \in [k]$.

The Verifier:

1. If $\sum_{i=1}^{k} w_i \neq w$, then reject.
2. For each $i \in \{1, \ldots, \log n\}$, repeat the following test $O(k/2^i)$ times:
   (a) Let $\delta_i \stackrel{\text{def}}{=} \frac{\varepsilon k}{2^{i+1} \log \varepsilon n}$.
   (b) If $\delta_i < 1/\sqrt{\ell}$, then read the entire i^{th} row and reject if $w_i \neq \sum_{j=1}^{\ell} x_{i,j}$;
   (c) Otherwise, read $O\left(\frac{1}{\delta_i^2} \cdot \log n\right)$ random points from the j^{th} row and reject if their average is not in the range $(\frac{w_j}{\ell} - \frac{\delta_i}{2}, \frac{w_j}{\ell} + \frac{\delta_i}{2})$.
3. If all the previous tests passed, then accept.

Figure 5: $\mathcal{MAP}$ for $\mathsf{Hamming}_w$

Soundness. Let $\varepsilon > 0$ and suppose that $\mathsf{wt}(x) \notin [w - \varepsilon n, w + \varepsilon n]$. Let $w_1, \ldots, w_k$ be an alleged proof for the false statement $\mathsf{wt}(x) = w$. Recall that $w_1, \ldots, w_k$ refer to the weights of the rows of x when interpreted as a $k \times \ell$ matrix. We assume that $\sum_{i=1}^{k} w_i = w$ since otherwise the verifier rejects with probability 1.

Claim 5.1.1 (Precision Sampling (cf. [Lev85, last paragraph of Section 9] or [Gol13, Appendix A.2]))**.** *There exists $i \in [\log \varepsilon n]$ such that 2^i of the rows j of x have weight at least $w_j + \frac{\varepsilon n}{2^{i+1} \log \varepsilon n}$ or at most $w_j - \frac{\varepsilon n}{2^{i+1} \log(\varepsilon n)}$.*

Proof. Let $d \stackrel{\text{def}}{=} \varepsilon n$. For $i \in [\log d]$, let

$$A_i \stackrel{\text{def}}{=} \left\{ j \in [k] : \ |\mathsf{wt}(x_j) - w_j| \geq \frac{d}{2^{i+1} \log d} \right\}$$

where x_j denotes the j^{th} row of x, and let $B = [k] \backslash (\cup_{i \in [\log d]} A_i)$. Notice that the sets $B, A_1, A_2 \backslash A_1, \ldots, A_d \backslash A_{d-1}$ form a partition of the k rows. Also note that for $j \in B$, it holds that $\mathsf{wt}(x_j) = w_j$ (since $|\mathsf{wt}(x_j) - w_j| < 2/\log d < 1$).

Suppose toward a contradiction that for every $i \in [\log d]$ it holds that $|A_i| < 2^i$. Using the fact for $j \in A_i \backslash A_{i-1}$ it holds that $|\mathsf{wt}(x_j) - w_j| < \frac{d}{2^i \log d}$, we get

$$\begin{aligned}
|\mathsf{wt}(x) - w| &\leq \sum_{j=1}^{k} |\mathsf{wt}(x_j) - w_j| \\
&= \sum_{i \in [\log d]} \sum_{j \in A_i \backslash A_{i-1}} |\mathsf{wt}(x_j) - w_j| \\
&\leq \sum_{i \in [\log d]} |A_i \backslash A_{i-1}| \cdot \frac{d}{2^i \log d} \\
&< \sum_{i \in [\log d]} 2^i \cdot \frac{d}{2^i \log d} \\
&= \varepsilon n,
\end{aligned}$$

in contradiction to our assumption that $\mathsf{wt}(x) \notin [w - \varepsilon n, w + \varepsilon n]$. □

Consider the execution of iteration i, where i is the index guaranteed by Claim 5.1.1. In this iteration, since the verifier selects $O(k/2^i)$ rows at random, with probability at least 0.9 it selects at least one row $j \in [k]$ such that $|\mathsf{wt}(x_j) - w_j| \geq \frac{\varepsilon n}{2^{i+1} \log(\varepsilon n)}$. Suppose that such a row j is indeed selected. We consider two cases:

1. If $\ell < \frac{1}{\delta_i^2}$, then the verifier reads the entire row and rejects, since the row's sum does not equal w_j.
2. If $\ell \geq \frac{1}{\delta_i^2}$, then, since the row j has an average of at least $w_j/\ell + \delta_i$ (or at most $w_j/\ell - \delta_i$), when the verifier samples from this row, the average of the samples will be greater than $w_i/\ell + \frac{\delta_i}{2}$ (resp., smaller than $w_i/\ell - \frac{\delta_i}{2}$) with probability at least $1 - 1/\mathsf{poly}(n)$, in which case the verifier rejects.

Thus, with probability at least 2/3, the verifier rejects and soundness follows. □

As corollaries of Theorem 5.1, we obtain

1. an $\mathcal{MAP}$ for every selection of $k = k(n)$, and
2. an $\mathcal{MAP}'$ (in which we allow k to depend on ε, and set it to $k = n^{\frac{1}{3}} \cdot \varepsilon^{-\frac{2}{3}}$) with total complexity (i.e., the sum of the query complexity and the proof complexity) $\tilde{O}\left(n^{\frac{1}{3}} \cdot \varepsilon^{-\frac{2}{3}}\right)$.

5.1.2 Lower Bound

In this section we show a lower bound on the $\mathcal{MAP}$ complexity of the property $\mathsf{Hamming}_{n/2}$ (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 $\mathsf{Hamming}_w$ for more general functions w by reducing to $\mathsf{Hamming}_{n/2}$ 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 $\mathcal{MAP}$ model that was established in Section 3.2.2. Recall that this extension allows us to prove lower bounds on the complexity of $\mathcal{MAP}$s via $\mathcal{MA}$ communication complexity lower bounds. We note that since an $\mathcal{MAP}$ lower bound refers to a particular value of ε, it immediately implies a lower bound also on $\mathcal{MAP}'$.

One natural candidate for a communication complexity problem on which we can base our $\mathsf{Hamming}$ 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 $\mathcal{MAP}$ lower bounds that we have shown before (e.g., for $\mathsf{TensorSum}$, and EIM), $\mathsf{Hamming}$ 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 $\mathsf{Hamming}$, we wish to reduce $\mathsf{Hamming}$ to an $\mathcal{MA}$ 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 $\mathsf{Gap\ Hamming\ Distance}$ 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 Appendix D for formal definitions and background. By extending a recent result of Gur and Raz [GR13], we show

Lemma 5.2. *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 $\mathcal{MA}$ 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 5.2, which is by a reduction to the result of [GR13], is presented in Appendix D (see Corollary D.3). Equipped with Lemma 5.2, we proceed to prove the lower bound for $\mathsf{Hamming}_w$.

Theorem 5.3. *For every $n \in \mathbb{N}$ and $\varepsilon \stackrel{\text{def}}{=} \varepsilon(n) \in (0, 1/2)$, if $\mathsf{Hamming}_{n/2}$ has an $\mathcal{MAP}$ 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 5.3. Throughout the proof we fix the function w as $w(m) \stackrel{\text{def}}{=} m/2$. By Lemma 3.21, if $\mathsf{Hamming}_w \in \mathcal{MAP}(p,q)$, then the communication complexity (promise) problem $\mathcal{C}_{\oplus,\varepsilon}^{\mathsf{Hamming}_w}$ has an $\mathcal{MA}$ 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 5.2, the theorem follows by reducing $\mathsf{GHD}_{n,n/2-\varepsilon n,\varepsilon n}$ to $\mathcal{C}_{\oplus,\varepsilon}^{\mathsf{Hamming}_w}$, which we proceed to do. (We stress that this reduction takes place entirely in the context of $\mathcal{MA}$ 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 $\mathcal{MA}$ 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 $\mathcal{C}_{\oplus,\varepsilon}^{\mathsf{Hamming}_w}$ $\mathcal{MA}$ 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 $\mathsf{Hamming}_w \in \mathcal{MAP}(p, q)$ where p and q are as in the hypothesis. Then, by Lemma 3.21, the $\mathcal{C}_{\oplus,\varepsilon}^{\mathsf{Hamming}_w}$ problem has an $\mathcal{MA}$ 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 $\mathcal{MA}$ 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 5.2, it holds that $p \cdot q = \Omega\left(\min(n, \varepsilon^{-2})\right)$.

□

Acknowledgments

We thank our advisor, Oded Goldreich, for his encouragement and guidance. We also thank Oded for multiple technical and conceptual suggestions that greatly improved both the results and presentation of this work, and for allowing us to include his generalization of [GS06, Theorem 5.20] in Appendix B.1.

References

[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.

[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.

[Bla10] Eric Blais. Testing juntas: A brief survey. In Goldreich [Gol10a], pages 32–40.

[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.

[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.

[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.

[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.

[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.

[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.

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

[GMR89] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. *SIAM J. Comput.*, 18(1):186–208, 1989.

[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. A sample of samplers: A computational perspective on sampling. In Oded Goldreich, editor, *Studies in Complexity and Cryptography*, volume 6650 of *Lecture Notes in Computer Science*, pages 302–332. Springer, 2011.

[Gol13] Oded Goldreich. On multiple input problems in property testing. *Electronic Colloquium on Computational Complexity (ECCC)*, 20:67, 2013.

[GR11] Oded Goldreich and Dana Ron. On proximity-oblivious testing. *SIAM Journal on Computing*, 40(2):534–566, 2011.

[GR13] Tom Gur and Ran Raz. Arthur-Merlin streaming complexity. In *Proceedings of the 40th International Colloquium on Automata, Languages and Programming (ICALP)*, 2013.

[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.

[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.

[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.

[KT00] Jonathan Katz and Luca Trevisan. On the efficiency of local decoding procedures for error-correcting codes. In *STOC*, pages 80–86, 2000.

[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.

[LV12] László Lovász and Katalin Vesztergombi. Nondeterministic graph property testing. *arXiv preprint arXiv:1202.5337*, 2012.

[New91] Ilan Newman. Private vs. common random bits in communication complexity. *Information processing letters*, 39(2):67–71, 1991.

[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.

[RS96] Ronitt Rubinfeld and Madhu Sudan. Robust characterizations of polynomials with applications to program testing. *SIAM J. Comput.*, 25(2):252–271, 1996.

[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.

[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.

[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.

[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.

A Background

A.1 Interactive Proofs of Proximity

In this section we define *interactive proofs of proximity*, following Rothblum *et al.* [RVW13].[18] 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.

Definition A.1. *An* interactive proof of proximity *system ($\mathcal{IPP}$) 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 $\mathcal{V}$.

[18] Our definition of $\mathcal{IPP}$ 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.)

If the completeness condition holds with probability 1, then we say that the $\mathcal{IPP}$ has a one-sided error *and otherwise the $\mathcal{IPP}$ is said to have a* two-sided error.

An $\mathcal{IPP}$ 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 $\mathcal{IPP}$ 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 $\mathcal{IPP}$ has query complexity q and communication complexity c, we say that it has $\mathcal{IPP}$ complexity $q + c$.

For every pair of functions $c, q : \mathbb{N} \times \mathbb{R}^+ \to \mathbb{N}$, we denote by $\mathcal{IPP}_2(c, q)$ (resp., $\mathcal{IPP}_1(c, q)$) the complexity class of all properties that have an $\mathcal{IPP}$ with communication complexity $O(c)$, query complexity $O(q)$ and two-sided error (resp., one-sided error). We also use $\mathcal{IPP}$ as a shorthand for the class $\mathcal{IPP}_2$.

An important parameter of an $\mathcal{IPP}$ is the number of messages m sent between the two parties. We denote by $\mathcal{IPP}[m](c, q)$ the set of properties that have m-message $\mathcal{IPP}$ protocols in which the verifier uses at most $O(c)$ bits of communication, and makes at most $O(q)$ oracles queries.

A.2 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.[19] 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 complex-

[19] In the case of a partial function, we consider only relevant x and y's.

ity 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,

Definition A.2. *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.

A.3 $\mathcal{MA}$ Communication Complexity

In $\mathcal{MA}$ *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 $\mathcal{MA}$ *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 $\mathcal{MA}$ *communication complexity protocol* as follows.

Definition A.3. *An $\mathcal{MA}(c,p)$-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*

$$\Pr\big[\pi\big((x,y),w\big) = 1\big] \geq 2/3$$

(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 $\mathcal{MA}$ 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 3.1.3 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 $\mathcal{MA}$ communication complexity of set-disjointness (later proved to be tight, by Aaronson and Wigderson [AW09]).

Theorem A.4. *Every $\mathcal{MA}$ communication complexity protocol for DISJ_n with proof complexity p and communication complexity c satisfies $p \cdot c = \Omega(n)$.*

A.4 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 A.5 (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.

A.5 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 A.6 (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 A.7 (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 A.7, 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 A.8 (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 A.9 (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 $0.9^2 > 1/2$. □

A.6 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 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 $\mathcal{IPP}$, 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 A.8) and self-correction (Theorem A.7).

B Missing Proofs from Section 3.1.6

In this section we provide proofs that were omitted from Section 3.1.6. These proofs are used to establish the existence of an $O(1)$-relaxed-LDC that is also a $\mathsf{poly}(1/\varepsilon)$-LTC (with polynomial stretch).

B.1 A Generalization of [GS06, Theorem 5.20]

In order to obtain the separation result of Theorem 3.3, we require a generalization of a theorem from [GS06]. The following was communicated to us by Oded Goldreich. In this subsection, we fix $F = \mathbb{F}_2$ (i.e., $\{0,1\}$, when viewed as a field).

Theorem B.1 (Generalization of [GS06, Theorem 5.20])**.** *Let $\Sigma = F^b$. There exists $n = \mathsf{poly}(k)$ and a linear code $E : \Sigma \to F^n$ such that the following holds. Suppose that $C : \Sigma^K \to \Sigma^N$ is an F-linear code having a strong-LTC [20] tester of randomness complexity r such that (w.l.o.g.) 2^r is a multiple of N. Furthermore, suppose that this tester is non-adaptive and queries each location with probability $\Theta(1/N)$. Then, there exists $\ell = \mathsf{poly}(k)$ such that ℓ is a multiple of n, and a linear LTC $C'' : F^{bk} \to F^{2^{r+1}\cdot\ell}$ such that the $2^r \cdot \ell$-bit long prefix of $C''(x)$ equals $\big(E(C(x)_1), \ldots, E(C(x)_N)\big)^{2^r \ell/Nn}$, where $C(x)_i$ is the ith symbol of $C(x)$.*

Alternatively, we can have this prefix take the form $\big(E(C(x)_1)^{2^r\ell/Nn}, ..., E(C(x)_N)^{2^r\ell/Nn}\big)$.

[20] A strong-LTC tester is not given the proximity parameter as input and is only required to reject strings that are ε-far from the code with probability $O(1/\varepsilon)$ (see [GS06, Definition 2.2]).

Corollary B.2 ([GS06, Theorem 5.20]). *For infinitely many k, there exists a locally-testable binary code of constant relative distance mapping k bits to $n \stackrel{\text{def}}{=} \exp\left(\tilde{O}\left(\sqrt{\log k}\right)\right) \cdot k$ bits. Furthermore, the code is linear.*

Proof. The claim follows by instantiating Theorem B.1 using the code of Part 1 of [GS06, Theorem 2.1], taking $b = \exp\left(\tilde{O}\left(\sqrt{K}\right)\right)$, $N = \exp\left(\tilde{O}\left(\sqrt{K}\right)\right) \cdot K$, and $r = \tilde{O}\left(\sqrt{K}\right) + \log_2 K$. □

Corollary B.3. *Let $C : F^k \to F^{ck}$ be a code of constant relative distance and constant rate $1/c$. Then, for some $m, n = \mathsf{poly}(k)$, there exists a strong LTC $C' : F^k \to F^{2m}$ and a function $E : F^k \to F^n$ such that the m-bit long prefix of $C'(x)$ equals $\big(E(y_1), ..., E(y_c)\big)^{m/cn}$ where y_i is the ith block of length k in $C(x)$.*

Proof. The claim follows by instantiating Theorem B.1 using the code $C : F^k \to F^{ck}$, but viewing it as a mapping $\Sigma = F^k$ to Σ^c, which is (trivially) checked by reading all c symbols. That is, take $b = k$, $K = 1$, $N = c = O(1)$, and $r = 0$. □

Proof of Theorem B.1. We follow the proof of [GS06, Theorem 5.20], while using C instead of the third ingredient. In the following all references refer to [GS06].

Recall some basics regarding the terminology used in [GS06]. By Definitions 5.8-5.9, a $\big(F, (q, b) \to (p, a), \delta, \gamma\big)$-LIPS refers to input oracles $X_1, ..., X_q : [n] \to F^a$ and a proof oracle $X_{q+1} : [\ell] \to F^a$, where the input oracles provide an n-long encoding (over F^a) of a single symbol in the (much) bigger alphabet F^b (i.e., this encoding is denoted $E : F^b \to (F^a)^n$). (In addition δ is the relative distance of the encoding used, and γ is the detection ratio in strong soundness. In the following, both parameters will be small constants.)

The proof of Theorem 5.20 starts with an overview (p. 79), and then lists three ingredients (p. 80) that will be used. We shall use the very same first two ingredients, but use C in place of the third. Specifically, the second paragraph following the ingredients-list asserts, for any desired p'' and k'', an $\big(F, (p'', k'') \to (p'' + 13, 1), \Omega(1), \Omega(1/p'')^2\big)$-LIPS with randomness $O(p'' \log k'')$ and input/proof lengths that are $\mathsf{poly}(p''k'')$. We shall use $p'' = O(1)$ and $k'' = b$, where the $O(1)$ stands for the query complexity of the codeword tester for C, so the above simplifies to asserting an $\big(F, (O(1), b) \to (O(1), 1), \Omega(1), \Omega(1)\big)$-LIPS with randomness $O(\log b)$ and input/proof lengths (i.e., n and ℓ) that are $\mathsf{poly}(b)$. Without loss of generality, we may assume that ℓ is a multiple of n.

Now we want to compose C with the above LIPS (via Theorem 5.13). It follows that in Item 1 of Theorem 5.13 we use K, N and r as provided by the hypothesis and $q = O(1)$. For Item 2, we use b as provided by the hypothesis, ($q = O(1)$ as above), $p = O(1)$ and $a = 1$, and $n, \ell = \mathsf{poly}(b)$ (all fitting the LIPS above). So we have $\Gamma = F$, and get a (strong) LTC mapping F^{bK} to $F^{2^{r+1} \cdot \ell}$. In particular, for $t = 2^r \ell / Nn$ (i.e., $tNn = 2^r \ell$), as shown on top of p. 56 (see Eq (32)), the first half of the codewords of the resulting code have the form

$\big(E(C(x)_1)),...,E(C(x)_N)\big)^t$, where $x \in F^{bK}$ is viewed as an element of Σ^K. The theorem follows. □

B.2 Proof of Lemma 3.12

Let $C' : \{0,1\}^k \to \{0,1\}^{ck}$ be a code with constant relative distance and constant rate $1/c$ for some constant $c > 1$. For every $j \in [c]$, we denote by $(C'(x))[j]$ the j^{th} block of length k of $C'(x)$. For every $i \in [k]$, define the code $C'_i : \{0,1\}^k \to \{0,1\}^{(c+1)k}$ as $C'_i(x) \stackrel{\text{def}}{=} \big(x_i^k, C'(x)\big) = \big(x_i^k, (C'(x))[1], \ldots, (C'(x))[c]\big)$. Note that C'_i has rate $1/(c+1)$ and constant relative distance.

For every $i \in [k]$, we apply Corollary B.3 to C'_i to obtain an LTC $C''_i : \{0,1\}^k \to \{0,1\}^{2m}$ of the form

$$C''_i(x) = \Big(E(x_i^k), E\Big((C(x))[1]\Big), \ldots, E\Big((C(x))[c]\Big), \pi_i(x)\Big)$$

where $n, m = \mathsf{poly}(k)$, the function $E : \{0,1\}^k \to \{0,1\}^n$ is a linear code, and $\pi_i(x) \in \{0,1\}^m$ is some string. Note that $C(x) \stackrel{\text{def}}{=} E\left(\Big((C(x))[1]\Big), \ldots, E\Big((C(x))[c]\Big)\right)$ is a code with constant relative distance.

Since E is a linear code with constant relative distance, $E(0^k) = 0^n$ and $\mathsf{wt}(E(1^k)) \geq \alpha n$ for some constant $\alpha \in (0,1)$. Consider the code $C'''_i(x) = \big(x_i^{\alpha n}, C'(x), \pi_i(x)\big)$ which is obtained from C'_i by simply removing coordinates on which $E(0^k)$ and $E(1^k)$ agree. Thus, $C'''_i(x)$ is a an LTC of the desired form.

B.3 Proof of Proposition 3.14

We show a procedure that, given oracle access to a binary string $\big((w_1, y_1, z_1), \ldots, (w_k, y_k, z_k)\big)$, where $(w_i, y_i, z_i) \in \{0,1\}^{c_1 n + c_2 n + c_3 n}$ for every $i \in [k]$, and a proximity parameter $\varepsilon > 0$, accepts every codeword of C' and rejects strings that are ε-far from C', with probability $1/2$. The LTC procedure for C' is described in Fig. 6.

Since for every $i \in [k]$ the code C_i is a $\mathsf{poly}(1/\varepsilon)$-$\mathsf{LTC}$, the tester indeed makes at most $\mathsf{poly}(1/\varepsilon)$ queries.

Completeness. If $\big((w_1, y_1, z_1), \ldots, (w_k, y_k, z_k)\big)$ is equal to $C'(x)$ for some $x \in \{0,1\}^k$, then for every $i \in [k]$ it holds that (1) $y_i = C(x)$, and so the codeword repetition test (Item 1a) passes with probability 1 and (2) (w_i, y_i, z_i) is equal to $C_i(x)$, and so, by the zero-error completeness of the LTC test for C_i, the second test (i.e., Item 1b) also passes with probability 1.

Soundness. Let $\big((w_1, y_1, z_1), \ldots, (w_k, y_k, z_k)\big)$ be ε-far from the code C', where $(w_i, y_i, z_i) \in \{0,1\}^{c_1 n + c_2 n + c_3 n}$ for every $i \in [k]$. Let $u \in \{0,1\}^{c_2 n}$ be a string that minimizes the value

The LTC Procedure for C'

Input: a string $\big((w_1, y_1, z_1), \ldots, (w_k, y_k, z_k)\big) \in \{0,1\}^{(c_1+c_2+c_3)nk}$ and a proximity parameter $\varepsilon \in (0, c)$ for some universal constant $c \in (0,1)$ specified below.

1. Repeat $O(1/\varepsilon)$ times:
   (a) **The codeword repetition test:** Select at random $i_1, i_2 \in [k]$ and $j \in [c_2 n]$, and reject if the j^{th} bit of y_{i_1} and y_{i_2} differs.
   (b) **The LTC test:** Select at random $i \in [k]$, and run the LTC tester of C_i on (w_i, y_i, z_i) with respect to proximity parameter $\varepsilon' = \Theta(\varepsilon)$.

Figure 6: Local tester for C'

of $\Delta\Big((y_1, \ldots, y_k), u^k\Big)$ (recall that Δ refers to the *absolute distance* between the strings). Let $\delta_0 \in (0,1)$ be the relative distance of the code C (recall that δ_0 is a constant). We proceed with a (somewhat tedious) case analysis that uses parameters $\gamma_1, \ldots, \gamma_5 = \Theta(\varepsilon)$. More specifically, we set:

$$\gamma_1 = \frac{\delta_0}{12} \cdot \varepsilon, \qquad \gamma_2 = 3\gamma_1, \qquad \gamma_3 = \varepsilon/3, \qquad \gamma_4 = \varepsilon/3, \qquad \gamma_5 = 2(\gamma_1 + \gamma_2)/\delta_0.$$

- Suppose that $(y_1, \ldots, y_k)$ is γ_1-far from u^k. In this case, each iteration of the codeword repetition test (i.e., Item 1a) rejects with probability

$$\begin{aligned}
\mathbf{E}_{i_1,i_2}\left[\Delta\left(y_{i_1}, y_{i_2}\right)/c_2 n\right] &= \mathbf{E}_{i_1}\left[\mathbf{E}_{i_2}\left[\Delta\left(y_{i_1}, y_{i_2}\right)\right)/c_2 n\right]\Big] \\
&\geq \mathbf{E}_{i_1}\left[\Delta\left(y_{i_1}, u\right)/c_2 n\right] \\
&= \Delta\left((y_1, \ldots, y_k), u^k\right)/c_2 nk \\
&> \gamma_1
\end{aligned}$$

  and so, since the test is repeated $O(1/\varepsilon)$ times, the tester rejects with probability at least $1/2$.

  Thus, in the sequel, we assume that $(y_1, \ldots, y_k)$ is γ_1-close to u^k.

- Suppose that u is γ_2-far from the code C. Since $(y_1, \ldots, y_k)$ is γ_1-close to u^k, at least half of the y_i's must be $2\gamma_1$-close to u. Thus, by the triangle inequality, in each invocation of the LTC test of C_i (i.e., Item 1b), with probability $1/2$, the test is invoked on a string (w_i, y_i, z_i) such that y_i is $(\gamma_2 - 2\gamma_1)$-far from C, and in particular, (w_i, y_i, z_i) is $\Theta(\varepsilon)$-far from C_i. Hence, in each one of the $O(1/\varepsilon)$ iterations, the LTC test rejects with probability at least $1/2 \cdot 1/2$.

  Thus, in the sequel, we assume that u is γ_2-close to a codeword $C(x)$.

- Suppose that for more than a γ_3 fraction of $i \in [k]$ it holds that (w_i, y_i, z_i) is γ_4-far from C_i. Then, for each invocation of the LTC test (Item 1b), with probability at least γ_3, the LTC test will be run on a string (w_i, y_i, z_i) that is γ_4-far from $C_i(x)$. In such a case, the LTC tester of C_i rejects with probability $1/2$.

  Thus, in each iteration the tester rejects with probability $\gamma_3/2$, and since there are $O(1/\varepsilon)$ iterations, the tester rejects with probability at least $1/2$. Hence, in the sequel we assume that for at most a γ_3 fraction of $i \in [k]$ it holds that (w_i, y_i, z_i) is γ_4-far from C_i.

By the triangle inequality, the string $(y_1, \ldots, y_k)$ is $(\gamma_1 + \gamma_2)$-close to the string $C(x)^k$. Hence, for at most a γ_5 fraction of $i \in [k]$, it holds that y_i is $(\gamma_1+\gamma_2)/\gamma_5$-far from $C(x)$. Thus, by the union bound, for at least $1 - \gamma_3 - \gamma_5$ fraction of $i \in [k]$ it holds that (1) (w_i, y_i, z_i) is γ_4-close to $C_i\left(x^{(i)}\right)$ for some $x^{(i)} \in \{0,1\}^k$ and (2) y_i is $(\gamma_1 + \gamma_2)/\gamma_5$-close to $C(x)$.

Suppose that for one of the i's above it holds that $x^{(i)} \neq x$. Then, y_i is $\Theta(\gamma_4)$-close to $C\left(x^{(i)}\right)$ and $(\gamma_1 + \gamma_2)/\gamma_5$-close to $C(x)$. Thus, by the triangle inequality and our setting or parameters, $C(x)$ is $(\Theta(\varepsilon) + \delta_0/2)$-close to $C\left(x^{(i)}\right)$. Recall that $\varepsilon < c$ for some constant $c > 0$ of our choosing and so we set c so that the two codewords $C(x)$ and $C(x_i)$ have relative distance less than δ_0, which is a contradiction.

Hence, at least a $1 - \gamma_3 - \gamma_5$ fraction of $i \in [k]$ are such that (w_i, y_i, z_i) is γ_4-close to $C_i(x)$. We conclude that $\big((w_1, y_1, z_1), \ldots, (w_k, y_k, z_k)\big)$ is at least δ-close to $C'(x)$, where

$$\delta \leq \gamma_3 + \gamma_5 + (1 - \gamma_3 - \gamma_5) \cdot \gamma_4 < \varepsilon,$$

in contradiction to our assumption. This ends the proof of Proposition 3.14.

C More on $\mathcal{MAP}$s with Proximity-Dependent Proofs

Recall that we defined the notion of $\mathcal{MAP}$s such that the proof of proximity is *oblivious* of the proximity parameter ε. However, it is also natural to consider a relaxation of $\mathcal{MAP}$s 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 $\mathcal{MAP}$, 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 $\mathcal{MAP}$ with only a logarithmic overhead to the proof complexity. As for the second possibility, we refer to it as an $\mathcal{MAP}'$ and note that we do not know of any general transformation from an $\mathcal{MAP}'$ to an $\mathcal{MAP}$.

We note that the distinction does not make much sense in the context of $\mathcal{IPP}$. Namely, in the $\mathcal{IPP}$ model we do give the prover ε as part of its input since otherwise the verifier

could simply send the value of ε to the prover (at the cost of (at most) a single message and additional $O\big(\log(1/\varepsilon)\big)$ communication).

D Lower Bound on the $\mathcal{MA}$ 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 D.1. *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 [GR13] showed the following tight lower bound on the $\mathcal{MA}$ communication complexity of GHD.

Theorem D.2 ([GR13])**.** *Every $\mathcal{MA}$ 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 $\mathcal{MA}$) to obtain the following corollary.

Corollary D.3. *Let $g, n \in \mathbb{N}$ such that $g \leq n$, let $\alpha \in (0,1)$ and $t = \alpha n$. Then, every $\mathcal{MA}$ communication complexity protocol for $\mathsf{GHD}_{n,t,g}$, with proof complexity $p \geq 1$, has communication complexity at least $\Omega\left(\frac{\min\left(n,(n/g)^2\right)}{p}\right)$.*

ECCC ISSN 1433-8092
http://eccc.hpi-web.de