Non-Interactive Proofs of Proximity*

Tom Gur
University of Warwick
tom.gur@warwick.ac.uk

Ron D. Rothblum*
Technion
ron.rothblum@technion.ac.il

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. On the negative side, we also show that there exist properties for which even a linearly-long (non-interactive) proof of proximity cannot significantly reduce the query complexity.

*This paper is published in the journal Computational Complexity, March 2018, Volume 27, Issue 1, pp 99-207.

Contents

1 Introduction

Understanding the power and limitations of sublinear-time algorithms is a central question in the theory of computation. The study of *property testing*, initiated by Rubinfeld and Sudan [RS96] and Goldreich, Goldwasser and Ron [GGR98], aims to address this question by considering highly-efficient randomized algorithms that solve approximate decision problems, while only inspecting a small fraction of the input. Such algorithms, commonly referred to as *property testers*, are given oracle access to some object, and are required to determine whether the object has some predetermined property, or is far (say, in Hamming distance) from every object that has the property. Remarkably, it turns out that many natural properties can be tested by making very few queries to the object.

Once a model of computation has been established, a fundamental question that arises is to understand the power of *proof systems* in this model. Recall that a proof system consists of a powerful prover that wishes to convince a weak verifier, which does not trust the prover, of the validity of some statement. Since verifying is usually easier than computing, using the power of proofs, it is often possible to overcome limitations of the basic model of computation. In this paper we study proof systems in the context of property testing, with the hope that by augmenting testers with proofs we can indeed overcome inherent limitations of property testers.

Thus, we are interested in proof systems in which the verifier reads only a small fraction of the input. Of course we cannot hope for such a verifier to reject *every* false statement. Instead, as is the case in property testing, we relax the soundness condition and only require that it be impossible to convince the verifier to accept statements that are *far* from true statements. Such proof systems were first introduced by Ergün, Kumar and Rubinfeld [EKR04] and were recently further studied by Rothblum, Vadhan and Wigderson [RVW13] who were motivated by applications to *delegation of computation* in sublinear time. Rothblum *et al.* [RVW13] showed that by allowing a property tester to interact with an untrusted prover (who can read the *entire* input), sublinear-time verification is indeed possible for a wide class of properties. As in the property testing framework, the tester is only assured of the proximity of the input to the property and hence such protocols are called *interactive proofs of proximity* ($\mathcal{IPP}$s).

1.1 The Notion of $\mathcal{MAP}$

In this work, we also consider proofs of proximity, but restrict the verification process to be *non-interactive*. In other words, we augment the property testing framework by allowing the tester full and explicit access to an (alleged) proof. Such a proof-aided tester for a property Π, is given oracle access to an input x and explicit 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

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

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 proof systems rely fundamentally on two-way interaction.)

Given the (widely believed) power of proofs in the context of *polynomial-time* computation, one would hope that proofs can help decrease the number of queries that is needed to test various properties. This is indeed the case. In fact, for every property Π, consider a proof system for the statement $x \in \Pi$, wherein the proof w is simply equal to x. In order to verify the statement, the tester need only verify that indeed $w \in \Pi$ and that w is close to x (i.e., that the relative Hamming distance between w and x is a small constant). The former check can be carried out without any queries to x, whereas for the latter a constant number of queries suffice.[2] 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*. Our main results are outlined in Sections 1.2 and 1.3.

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 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

[2] Note that for objects that are not binary strings (e.g., functions over finite fields), each query returns an element of a set Σ that may require $\omega(1)$ bits to represent.

to also consider the computational complexity of generating and verifying the proof. However, in this work our main focus is on the query and proof complexities. Still, we note that unless stated otherwise, our protocols can be implemented efficiently; that is, the proof can be generated in *polynomial-time* and verified in *sublinear-time*.

Comparison with $\mathcal{PCP}$s of Proximity. $\mathcal{PCP}$s of proximity ($\mathcal{PCPP}$s), first studied by Ben-Sasson *et al.* [BGH+06] and by Dinur and Reingold [DR06] (where they are called assignment testers) are also non-interactive proof systems in which the verifier has oracle access to an object, and needs to decide whether the object is close to having a predetermined property. However, $\mathcal{PCPP}$s differ from $\mathcal{MAP}$s in that the verifier is only given *query* (i.e., oracle) access to the proof, whereas in $\mathcal{MAP}$s, the verifier has *explicit* access to the proof. Indeed, the proof string in $\mathcal{PCPP}$s is typically of super-linear length (but only a small fraction of it is actually read at random), and in contrast, in $\mathcal{MAP}$s (similarly to limited-nondeterminsm complexity [PY96], bounded-communication interactive proofs [GH98], and laconic-prover interactive proofs [GVW02]), the proof is short, and in particular, sublinear. 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 (i.e., via query access or explicit access), leads to the taxonomy depicted in Table 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	**No Proof**	**Explicit Access**	**Oracle Access**
Explicit Access	$\mathcal{P}$	$\mathcal{NP}$ or $\mathcal{MA}$	$\mathcal{PCP}$
Oracle Access	Property Testers	$\mathcal{MAP}$ (**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 savings in the query complexity while using only a relatively short proof. Our first result shows that indeed there exists a property for which a dramatic saving is possible:

Theorem 1 (seperating $\mathcal{MAP}$ from testers (informally stated, see Theorem 3.1))**.** *There exists a (natural) 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 a smooth *multiplicative* trade-off, which is (almost) tight, between the number of queries and length of the proof holds:[3]

Theorem 2 (proof-query tradeoff (informally stated, see Theorem 3.12))**.** *There exists a (natural) property* Π *such that for every* $p \geq 1$*: (1) there is an* $\mathcal{MAP}$ *for* Π *that uses a proof of length* p *and makes* $O\left(\frac{n^{0.999}}{p}\right)$ *queries, and (2) every* $\mathcal{MAP}$ *for* Π *with proof length* p *must have query complexity* $\Omega\left(\frac{n^{0.998}}{p}\right)$*.*

Next, 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.*

Theorem 3 (one-sided error $\mathcal{MAP}$ (informally stated, 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, Theorem 3 implies an exponential separation between $\mathcal{MAP}$s (with poly-logarithmically long proofs) and *one-sided error* property testing. We note that Theorem 1 shows such a separation for the more general case of two-sided error.

We note that all of the explicit properties that were discussed thus far are properties "with distance"; that is, properties for which every two objects that have the property are far apart. In other words, the set of objects forms an error-correcting code. This distance, along with a form of local *self-correction*, is a crucial ingredient of the foregoing $\mathcal{MAP}$s. In contrast, all of the properties described next are properties "without distance". Hence, the power of $\mathcal{MAP}$s is not limited to properties with distance.

$\mathcal{MAP}$s for parameterized concatenation problems. We identify a family of natural properties, for which it is possible to construct efficient $\mathcal{MAP}$s, by using a generic scheme. Specifically, for every problem that can be expressed as a parameterized concatenation problem, we show how to construct an efficient $\mathcal{MAP}$ that allows a trade-off between the query and proof complexity. Loosely speaking, a property Π is a parameterized concatenation problem if $\Pi = \Pi_{\alpha_1} \times \cdots \times \Pi_{\alpha_k}$, for some integer k, where each property Π_{α_i} is a property parameterized by α_i (represented, say, by a string). For example, the property of all n-bit strings with Hamming weight w, denoted

[3] We remark that the relation $p \cdot q \approx \Theta(n)$ is not the best possible for either upper bounds (e.g., the empty language has an $\mathcal{MAP}$ with $p = q = 0$) or lower bounds (see Theorem 4). Theorem 2 shows that there exists a property for which a smooth tradeoff is possible.

$\mathsf{Hamming}_n^w$, can be written as $\mathsf{Hamming}_{n/k}^{w_1} \times \cdots \times \mathsf{Hamming}_{n/k}^{w_k}$, where $w_1 + \ldots + w_k = w$. Using the aforementioned generic scheme, we obtain $\mathcal{MAP}$s for a couple of natural problems, including: (1) approximating the Hamming weight of a string, and (2) graph orientation problems. (For more details, see Section 6).

$\mathcal{MAP}$s for graph properties. To see that $\mathcal{MAP}$s are also useful for testing graph properties, we consider the problem of testing bipartiteness in the *bounded-degree* graph model. We construct an $\mathcal{MAP}$ protocol for verifying bipartiteness of *rapidly-mixing graphs*, with proof complexity p and query complexity q, *for every* p and q such that $p \cdot q \geq N$ (where N is the number of vertices in the graph). In particular, we obtain an $\mathcal{MAP}$ verifier that uses a proof of length $N^{2/3}$ and makes only $N^{1/3}$ queries. This stands in contrast to the $\Omega(\sqrt{N})$ lower bound on the query complexity of property testers (which do not use a proof), shown by Goldreich and Ron [GR02], which also holds for *rapidly-mixing graphs*. We remark that in [RVW13] a (multi-round) $\mathcal{IPP}$ was given for the same problem (see Section 7).

We note that in the *dense* graph model, testing bipartiteness (or more generally k-colorability) can be easily done using only $O(1/\varepsilon)$ queries (where ε represents the desired proximity to the object) when given a proof that is simply the k-coloring of the graph (which can be represented by $N \log_2 k$ bits where N is the number of vertices and k is the number of colors).[4] In contrast, for standard property testers such query complexity is impossible (see [BT04]). We note that a similar protocol (described as a $\mathcal{PCPP}$) for testing bipartiteness in the dense graph model was suggested in [EKR04] and in [BGH+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? As discussed above, a proof of linear length suffices to reduce the query complexity to $O(1/\varepsilon)$. Moreover, Theorem 1 shows that even a logarithmically long proof can be extremely useful for a specific property. Thus, it is natural to ask whether a *sublinear* proof can reduce the query complexity for *every* property. The following result shows that for *almost all* properties, even a proof of length $n/100$ cannot improve the query complexity by more than a constant factor.

Theorem 4 (a hard property for $\mathcal{MAP}$ (informally stated, see Theorem 5.1))**.** *For almost all properties, every $\mathcal{MAP}$ verifier that uses a proof of length $n/100$ must make $\Omega(n)$ queries.*[5]

[4] 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.

[5] In fact, we show a general additive tradeoff between proof and query complexities, that is, every $\mathcal{MAP}$ verifier that uses a proof of length p must make $\tilde{\Omega}(n-p)$ queries.

Although Theorem 5.1 holds for most properties, finding an *explicit* property for which a similar statement holds remains an interesting open question. We note that Theorem 4 improves upon a result of Fischer *et al.* [FGL14] (see discussion in Section 1.5).

Since Theorem 4 shows that even a relatively long proof cannot help in general for *every* property, one might ask whether there are specific properties for which short proofs do suffice. As was shown in Theorem 1, this is indeed the case and a logarithmically long proof allows for an exponential improvement in the query complexity for a specific property. But can an even shorter, say constant-size proof, help? Unfortunately, the answer is negative since an $\mathcal{MAP}$ 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 verifer's queries are independent of the provided proof. Such $\mathcal{MAP}$s can be viewed as a two step process in which the verifier first (adaptively) queries the object and only then it receives the proof and decides whether to accept or reject based on both the answers and the proof. We say that such $\mathcal{MAP}$s have proof oblivious queries. The following result shows that $\mathcal{MAP}$s with *proof-oblivious queries* can provide at most a *quadratic* improvement over standard property testers.

Theorem 5 (emulating proof-oblivious $\mathcal{MAP}$ by testers (informally stated, 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 Theorem 1, the restriction to *proof oblivious queries* is a necessary precondition for Theorem 5 (and indeed, the $\mathcal{MAP}$ verifier of Informal Theorem 1 must make proof-dependent 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 complexity.

Theorem 6 (seperating $\mathcal{IPP}$ from $\mathcal{MAP}$ (informally stated, see Theorem 3.19 and Theorem 3.21))**.** *There exists a property* Π *such that on the one hand, any* $\mathcal{MAP}$ *for* Π *with proof of length at most* $n^{0.499+o(1)}$ *has query complexity at least* $n^{0.499-o(1)}$*, and on the other hand,* Π *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

Several of our results (in particular Informal Theorems 2 and 6) are based on a specific algebraic property, which we call *Sub-Tensor Sum* and denote by TensorSum (c.f. [LFKN92]). Let $\mathbb{F}$ be a finite field and let $H \subset \mathbb{F}$ be an arbitrary subset. We consider m-variate polynomials over $\mathbb{F}$ that have individual degree d. The TensorSum property contains all such polynomials whose sum on H^m equals 0.[6] 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.
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.7).
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

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

a slightly more sophisticated technique, 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.[7]

Parameterized Concatenation Problems. Our techniques for showing $\mathcal{MAP}$s for properties that do not have distance (and a structure that allows for self-correction) differ from the above. One class of problems that we consider is that of *parameterized concatenation problems*. Such properties consists of strings that are a concatenation of substrings, where each substring satisfies a particular parameterized property. The actual parameterization is not known a priori to the tester, and so an $\mathcal{MAP}$ proof that simply provides this parameterization turns out to be quite useful. Given this parameterization, the $\mathcal{MAP}$ verifier can simply test each substring individually (or a random subset of these substrings). Actually, in order to solve the problem more efficiently, the different substrings are tested with respect to different values of the proximity parameter by using a technique known as *precision sampling* (see survey [Gol13, Appendix A]).

Verifying Bipartiteness of Well-Mixing Graphs. Our $\mathcal{MAP}$ protocol for proving bipartiteness of a given *well-mixing* graph $G = (V, E)$ of size $N = |V|$ proceeds as follows. The proof consists of a subset $W \subseteq V$ of vertices that are allegedly on the same side of the graph. The verifier selects a random vertex $s \in V$ and takes roughly $N/|W|$ random walks of length $\Theta(\log n)$, starting at s. The verifier rejects if two of the walks pass through vertices of the set W, where the lengths of the paths from s to these vertices of W have opposite parities. Indeed, such walks cannot occur in bipartite graphs, assuming that all vertices in S are on the same side.

We show that if the graph is rapidly mixing and far from bipartite, then, for a $O(1/\log(N))$ fraction of vertices $s \in W$, the probability that a random walk starting in s will end in W with odd (respectively, even) parity is roughly $|W|/N$. Since the verifier takes $N/|W|$ random walks starting in s, with constant probability, it will detect a violation and reject. The analysis of our protocol is inspired by [GR02]. Interestingly, in contrast to the analysis of the rapidly-mixing case in [GR02], our analysis crucially relies on the random selection of the starting vertex.

Lower Bounds via $\mathcal{MA}$ Communication Complexity. As for our property testing *lower bounds*, we base these on the recently introduced technique of Blais, Brody and Matulef [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

[7] 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.

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. Additionally, we extend a recent result of Gur and Raz [GR13b] who give an $\mathcal{MA}$ communication complexity lower bound on the classical problem of *Gap Hamming Distance*.

We note that nearly all of the lower bounds shown in [BBM11] are proved via reductions from the communication complexity problems of *set-disjointness* and *gap Hamming distance*. Since these communication complexity problems have known $\mathcal{MA}$ communication complexity lower bounds (cf. [Kla03, GR13b]), these reductions, together with our extension of the [BBM11] framework to $\mathcal{MAP}$s, gives $\mathcal{MAP}$ *lower bounds for the problems studied in [BBM11]* (e.g., testing juntas, Fourier degree, sparse polynomials, monotonicity, etc.).

Lower Bounds via the Probabilistic Method. Lastly, to prove Theorem 4, which shows a property that requires $\Omega(n)$ queries even from an $\mathcal{MAP}$ that has access to a proof of length $n/100$, we use a technique that is inspired by [GGR98], and also uses ideas from [RVW13]. In more detail, we note that $\mathcal{MAP}$s can be represented by a relatively small class of functions. Since this class of functions is small, using the probabilistic method, we argue that a "random property" (chosen from an adequate distribution) fools every $\mathcal{MAP}$ verifier in the sense that the verifier cannot distinguish between a random input that has the property and a totally random input (which will be far from the property).

1.5 Related Works

The notion of interactive proofs of proximity was first considered by Ergün, Kumar and Rubinfeld [EKR04] (where it was called approximate interactive proofs). More recently, Rothblum, Vadhan and Wigderson [RVW13] initiated a systematic study of the power of this notion. Their main result is that all languages in $\mathcal{NC}$ 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. We remark that a straightforward application of the techniques in [RVW13] implies an $\mathcal{MAP}$ lower bound of $\Omega(\sqrt{n})$ for a non-explicit property and a lower bound of $\Omega(n^{1/4})$ for an explicit property, whereas Theorem 4 and Theorem 2 show an $\mathcal{MAP}$ lower bound of $\Omega(n)$ for a non-explicit property and $\Omega(\sqrt{n})$ for an explicit property (respectively).

The study of interactive proof 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.

Relation to Annotated Data Steams. 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 [CCMT14] and extended in [CMT12, CMT13, GR13b, CCGT14, Tha14, CCM+15, DTV15]. We remark that there are several related notions between $\mathcal{MAP}$s and annotated data streams. For example, $\mathcal{MAP}$s that make proof oblivious queries can be thought of as analogous to online annotated data streams, and general $\mathcal{MAP}$s can be thought of as analogous to prescient annotated data streams (see [CCMT14] for definitions).

Relation to Partial Testing [FGL14]. Independently of this work, Fischer, Goldhirsh and Lachish [FGL14] introduced the notion of *partial testing*, which is closely related to $\mathcal{MAP}$s. A property Π is a said to be Π'-partially testable, for $\Pi' \subseteq \Pi$, if inputs in Π' can be distinguished from inputs that are far from Π by a tester that makes only few queries. As pointed out by [FGL14], an $\mathcal{MAP}(p,q)$ for a property Π is equivalent to the existence of sub-properties $\Pi_1, \ldots, \Pi_{2^p} \subseteq \Pi$ such that $\cup_{i \in [2^p]} \Pi_i = \Pi$ and for every $i \in [2^p]$, the property Π is Π_i-partially testable using q queries.

In our terminology, the main result of [FGL14] is that there exists a (natural) property Π such that every $\mathcal{MAP}(p,q)$ for Π must satisfy that $p \cdot q = \Omega(n)$. In contrast, Theorem 2 shows a different property Π' for which $p \cdot q = \Omega(n^{0.999})$. However, we also show an (almost) matching *upper* bound for our property Π' (see Theorem 2). We also note that Theorem 4 (see Theorem 5.1), which was discovered following the publication of [FGL14], shows a property for which every $\mathcal{MAP}(p,q)$ must satisfy $p + q = \Omega(n)$; that is, if $p = n/100$, then $q = \Omega(n)$. We note that the latter result also resolves (a natural interpretation of) a question asked by [FGL14, Open Question 1.4].[8]

Applications of our Work and Follow-Up Works. Our work has also found applications in unrelated studies. For example, in the study of *sample-based testers*, Goldreich and Ron [GR13a] used the separation between the power of $\mathcal{MAP}$s and property testers (see Theorem 3.1) in order to show that proximity-oblivious testers do not necessarily imply *fair* proximity-oblivious testers (where fair proximity-oblivious testers are testers in which every query is almost uniformly distributed). Another example is an application for *testing dynamic environments*. Specifically, the separation between the power of standard $\mathcal{MAP}$s and $\mathcal{MAP}$s with *proof-oblivious queries* (see Lemma 3.6 and Theorem 4.2) was used to show that time-conforming testers can be exponentially weaker than their non-time-conforming counterparts (see [GR14] for details). In addition, following the publication of this work, Goldreich, Gur, and Komargodski [GGK14] improved on Theorem 1 by tightening the separation between $\mathcal{MAP}$s and testers (see Section 3.1 for more details).

Non-Deterministic Testing of Graphs Last, we note that Alon *et al.* [AFNS09] discussed the notion of *non-deterministic property testing of graphs*, which was formally stated recently by Lovász and Vesztergombi [LV12], and further studied by Gishboliner and Shapira *et al.* [GS13]. This model is a form of $\mathcal{PCP}$ of proximity in which both the proof and verification procedure are restricted to be of a particular form.

[8] Loosely speaking, in the terminology of [FGL14], Theorem 5.1 implies that for every r there exists a property Π that can be tested with r queries, but every partition of Π into k properties $\Pi_1, \ldots, \Pi_k$, such that Π is P_i-partially testable with $O(1)$ queries, must satisfy that $k = 2^{\Omega(r)}$.

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. In Section 5 we show a property that is hard for $\mathcal{MAP}$s even given a (relatively) long proof. In Section 6 we consider $\mathcal{MAP}$s for concatenation problems and in Section 7 we show our $\mathcal{MAP}$ for verifying bipartiteness of rapidly-mixing graphs in the bounded degree model. 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 an 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* $\Pi = \cup_{n\in\mathbb{N}}\Pi_n$ *consists of a probabilistic algorithm* V*, called the verifier, that is given as explicit inputs an integer* $n \in \mathbb{N}$*, a proximity parameter* $\varepsilon > 0$*, and a proof string* $w \in \{0,1\}^*$*; in addition, it is given oracle access to a function* $f \in \mathcal{F}_n$*. The verifier satisfies the following two conditions:*

1. Completeness: *For every* $n \in \mathbb{N}$ *and* $f \in \Pi_n$*, there exists a string* w *(referred to as a* proof *or* witness*) such that for every proximity parameter* $\varepsilon > 0$*:*

$$\Pr\left[V^f(n,\varepsilon,w) = 1\right] \geq 2/3.$$

   where the probability is over the random coin tosses of the verifier V*.*

2. Soundness: *For every* $n \in \mathbb{N}$*, function* $f \in F_n$*, string* w*, and proximity parameter* $\varepsilon > 0$*, if* f *is* ε*-far from* Π_n*, then:*

$$\Pr\left[V^f(n,\varepsilon,w) = 1\right] \leq 1/3.$$

   where the probability is over the random coin tosses of the verifier V*.*

If the completeness condition holds with probability 1*, then we say that the* $\mathcal{MAP}$ *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 Section 2.2 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.[9] 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 Section 2.1 for a discussion of the alternate notion, in which the proof *may* depend on the proximity parameter.

Proof oblivious queries. An aspect of $\mathcal{MAP}$ 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:

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).

[9] 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 $p(n)$.

2. After getting answers to all of its queries, the verifier is given explicit and explicit 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 non-decreasing 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.

$\mathcal{MAP}$s with Proximity-Dependent Proofs 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.

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].

2.2 Interactive Proofs of Proximity

In this section we define *interactive proofs of proximity*, following Rothblum *et al.* [RVW13].[10] 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 2.4. *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 $\mathcal{V}$.

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 $q + c$.

[10] 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.)

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.

2.3 Useful Conventions

The proximity parameter. We view the proximity parameter as a function $\varepsilon = \varepsilon(n)$. For simplicity we assume that $\varepsilon(n)$ is a non-increasing function.

Our definition of $\mathcal{MAP}$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 $T^f(n, \varepsilon)$).

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 tradeoff 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 the following 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.

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.*

The above separation results refer to the general (i.e., two-sided error) class $\mathcal{PT}_2$. 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. We remark that the preliminary technical report [GR13c] also contained a proof of the following (incomparable) variant, which can handle all values of the proximity parameter while using $\mathsf{poly}(1/\varepsilon)$ query complexity, at the cost of having a smaller (yet still exponential) gap between the power of property testers and $\mathcal{MAP}$s.

Theorem 3.3 ([GR13c])**.** *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.*[11]

A different proof of Theorem 3.3 is sketched in [FGL14] who, using a result of Alon *et al.* [AKNS00], showed a property that requires $\Omega(\sqrt{n})$ queries (without a proof) but can be tested using only $O(1/\varepsilon)$ queries and a proof of length $O(\log n)$.

Follow-Up Work. Following the publication of this work, Goldreich, Gur, and Komargodski [GGK14] improved the separation between $\mathcal{MAP}$s and testers, achieving the best of Theorems 3.1 to 3.3 simultaneously; that is, they obtain a separation for all values of the proximity parameter, with constant query complexity for the $\mathcal{MAP}$s, and nearly-linear query complexity for testers.

Theorem 3.4 ([GGK14])**.** *For every constant $\alpha > 0$, there 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 (without limitation on ε), but for which every* property tester *must make $n^{1-\alpha}$ queries. Furthermore, the $\mathcal{MAP}$ has one-sided error.*

In the next subsections we will show two lemmas (Lemmas 3.5 and 3.6) that allow us to reduce the problem of separating the power of $\mathcal{MAP}$s and testers to the problem of designing error-correcting codes that are both locally testable and locally decodable. Theorems 3.1 to 3.4 are then obtained by instantiating Lemmas 3.5 and 3.6 with such codes. Since the codes of [GGK14] improve upon the codes that are used to obtain Theorems 3.1 to 3.3, we omit the more involved proof of Theorem 3.3, which consists of a construction of a code with the desired properties (see technical report [GR13c] for details and proof). We provide the proofs of Theorems 3.1 and 3.2, which are instantiations of Lemmas 3.5 and 3.6 for known codes.

[11] 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.

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 is the minimal relative distance between every two (distinct) codewords (i.e., the fraction of locations in which the codewords differ), and the length of the code is n when viewed as a function of k. Further necessary background is provided in Appendix A.3.

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.3), 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 length 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 suitable instantiation of) the Reed-Muller code is both an LTC and LDC with $\mathsf{polylog}(n)$ query complexities, and almost linear length. 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) are known, constructing LDCs with constant query complexity (and polynomial length) 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.* [BGH+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.4 for the formal definition.

Ben-Sasson *et al.* [BGH+06] used $\mathcal{PCPP}$s to construct an $O(1)$-relaxed-LDC with almost linear length. Furthermore, [BGH+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 [BGH+06] code, we (only) obtain Theorem 3.1. In addition, by combining ideas and results of [BGH+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) length. 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 [BGH+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.

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

Lemma 3.5. *Let $C : \Sigma^k \to \Sigma^n$ be a code with constant relative distance that is a t_1-relaxed-LDC and also a t_2-LTC. Then, $\mathsf{EIM}_C \in \mathcal{MAP}_1\big(\log k, t_1(n/2) + t_2(n/2,\varepsilon/2)\big)$.*

Proof. We prove Lemma 3.5 by showing an $\mathcal{MAP}$ 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).

Since the code is a t_1-relaxed-LDC and a t_2-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

$\mathcal{MAP}$ 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

$x_i, y_i \neq 0$, 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 3.6. *Let $C : \Sigma^k \to \Sigma^n$ be an error-correcting code with relative distance at least $\delta_0 \in (0, 1)$. Then, for any $\varepsilon \in (0, \delta_0/2)$ it holds that:*

$$\mathsf{PT}_\varepsilon\big(\mathsf{EIM}_C\big) = \Omega(k/\log|\Sigma|)$$

The proof of Lemma 3.6 uses the framework of [BBM11] for showing property testing lower bounds via communication complexity lower bounds. The necessary background on communication complexity is provided in Appendix A.1 (for a comprehensive introduction to communication complexity, see [KN97]).

The basic approach of [BBM11] is to reduce a hard communication complexity problem to the property testing problem for which we want to show a lower bound. We follow [BBM11] by showing a reduction from the well-known communication complexity problem of *set-disjointness*. The aforementioned framework allows us to obtain a lower bound on the query complexity of testing the *encoded intersecting messages* property.

For sake of self containment, we state the relevant definitions and lemmas that we need from [BBM11].

Definition 3.7 (Combining operators)**.** *A* combining operator *is an operator ψ that takes as input two functions $f, g : D \to R$ (where D and R are some finite sets) and returns a function $h_{f,g}$. We denote by $|\psi| \stackrel{\text{def}}{=} \log_2 |R|$. The combining operator is called* simple *if $h_{f,g}(x)$ can be computed from $x, f(x)$ and $g(x)$ (i.e., without requiring access to f and g).*

Let Π be a property, and let ψ be a combining operator. For every integer $n \in \mathbb{N}$ and proximity parameter $\varepsilon > 0$, we denote by $\mathcal{C}^{\Pi}_{\psi,\varepsilon}$ the communication complexity problem wherein Alice gets a function f, and Bob gets a function g,[12] and their goal is to decide whether $\psi(f, g) \in \Pi$ or $\psi(f, g)$ is ε-far from Π.[13] Next, we state the main lemma from [BBM11].

Lemma 3.8. *For any* simple *combining operator ψ, any property Π and any proximity parameter $\varepsilon > 0$, we have that:*

$$\mathsf{PT}_{\varepsilon}(\Pi) \geq \frac{\mathsf{CC}(\mathcal{C}^{\Pi}_{\psi,\varepsilon})}{2|\psi|}$$

where $\mathsf{PT}_{\varepsilon}(\Pi)$ refers to the query complexity of the property Π with respect to proximity ε and $\mathsf{CC}(\mathcal{C})$ refers to the communication complexity of $\mathcal{C}$ (see Appendix A.1).

Recall that the set-disjointness problem is the communication complexity problem wherein Alice gets an n-bit string x, Bob gets an n-bit string y, and their goal is to decide whether there exists $i \in [n]$ such that $x_i = y_i = 1$. Equivalently, Alice and Bob's inputs can be viewed as indicator vectors of sets $A, B \subseteq [n]$. In this case, the goal of the players is to decide if the sets corresponding to their inputs intersect or not. Following many works in the literature we consider the promise problem (sometimes also called *unique* disjointness) in which the intersection is of size at most 1. That is, the two parties need to distinguish between the case that their intersection is empty, and the case that it is of size exactly 1. We denote the latter problem by DISJ_n.

It is well-known (see Appendix A.1) that the randomized communication complexity of the *set-disjointness* problem is linear in the size of the inputs, even under the promise that A and B intersect in at most one element.

Theorem 3.9 ([KS92])**.** *For every $n \in \mathbb{N}$,*

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

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

[13] 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 15.

Using the aforementioned results, we are ready to prove Lemma 3.6.

Proof of Lemma 3.6. Let $C : \Sigma^k \to \Sigma^n$ be an error-correcting code with relative distance $\delta_0 \in (0,1)$ where we assume without loss of generality that $\{0,1\} \subseteq \Sigma$. Denote by Pair the operator that takes two strings $x, y \in \Sigma^k$ and returns a function $z : [k] \to \Sigma$ that outputs (x_i, y_i) on input $i \in [k]$. Consider $\mathcal{C}^{\mathsf{EIM}_C}_{\mathsf{Pair},\varepsilon}$, the communication complexity problem wherein Alice gets a string $x \in \Sigma^k$, Bob gets a string $y \in \Sigma^k$, and their goal is to decide whether $(x,y) \in \mathsf{EIM}_C$ or (x,y) is ε-far from EIM_C. Using the results of [BBM11] (see Lemma 3.8) we have,

$$\mathsf{PT}_\varepsilon(\mathsf{EIM}_C) \geq \frac{1}{2\log|\Sigma|}\mathsf{CC}\left(\mathcal{C}^{\mathsf{EIM}_C}_{\mathsf{Pair},\varepsilon}\right). \tag{3.1}$$

Since by Theorem 3.9 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 3.9, 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.* [BGH+06].

Theorem 3.10 ([BGH+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 length $n = k^{1+\alpha}$, 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 3.10 with Lemma 3.5 and Lemma 3.6.

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 a $\mathsf{polylog}(n)$-LDC (and in particular a $\mathsf{polylog}(n)$-relaxed-LDC) and a $\mathsf{poly}(\log n, 1/\varepsilon)$-$\mathsf{LTC}$. Combining the latter with Lemma 3.5 and Lemma 3.6, we prove Theorem 3.2.

Lemma 3.11. *For every constant $\alpha > 0$, there exists a $\mathsf{polylog}(n)$-LDC that is also a $\mathsf{poly}(\log n, 1/\varepsilon)$-$\mathsf{LTC}$ with length $n = k^{1+\alpha}$ 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.8), and locally decoded using $O(m|H|)$ queries (see Theorem A.6). 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 (known as the product Reed-Solomon 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

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

3.2 Trade-off between Query and Proof Complexity

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

Theorem 3.12. *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.4 for the necessary background (e.g., the Schwartz-Zippel lemma and low degree testing). In fact, the proof of Theorem 3.12 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 3.13. *The* Sub-Tensor Sum *property, denoted* $\mathsf{TensorSum}_{\mathbb{F},m,d,H}$*, is parameterized by a field* $\mathbb{F}$*, a dimension* $m \in \mathbb{N}$*, a degree* $d \in \mathbb{N}$ *and a subset* $H \subset \mathbb{F}$*, and contains all polynomials* $P : \mathbb{F}^m \to \mathbb{F}$ *of individual degree* d*, such that*

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

where the arithmetic is over $\mathbb{F}$*.*

To obtain a tight trade-off, we shall be using some $d = \Theta(|H|)$. To simplify the notation, when the parameters are clear from the context, we shorthand $\mathsf{TensorSum}$ for $\mathsf{TensorSum}_{\mathbb{F},m,d,H}$. Next, we proceed to show the (almost) tight multiplicative trade-off for $\mathsf{TensorSum}$. In Section 3.2.1 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 3.12.

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

We start by proving the following upper bound.

Lemma 3.14. *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* $(d+1)^\ell \cdot \log(|\mathbb{F}|)$ *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 3.14. 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.8) 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.6) 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.8), 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 3.15. *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 3.15 we obtain the following:[14]

Corollary 3.16. *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 3.15, 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.2 for background on $\mathcal{MA}$ communication complexity.

Let Π be a property and let ψ be a simple combining operator (see Definition 3.7). For every proximity parameter $\varepsilon > 0$, denote by $\mathcal{C}^{\Pi}_{\psi,\varepsilon}$ the communication complexity problem in which Alice gets as input a function f and Bob gets as input a function g and they 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.[15] We prove the following lemma.

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

[15] 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 3.6). 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.

Lemma 3.17 ($\mathcal{MAP}$ 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 explicit 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 (unique) set-disjointness [Kla03], and use Lemma 3.17 to show a lower bound on the $\mathcal{MAP}$ complexity of the *Sub-Tensor Sum* property.

Theorem 3.18 ([Kla03])**.** *Every $\mathcal{MA}$ communication complexity protocol for DISJ_n with proof complexity p and communication complexity c satisfies $p \cdot c = \Omega(n)$.*

Proof of Lemma 3.15. Denote $k = |H|^m$ and by $f \cdot g$ the function $h(x) \stackrel{\text{def}}{=} f(x) \cdot g(x)$. Let $\mathcal{C}^{\mathsf{TensorSum}}_{\cdot,\varepsilon}$ be the communication complexity problem wherein Alice gets a function $f : \mathbb{F}^m \to \mathbb{F}$, 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 3.18 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 3.17 we know that if $\mathsf{TensorSum} \in \mathcal{MAP}(p, q)$, then $\mathsf{CC}(\mathcal{C}^{\mathsf{TensorSum}}_{\cdot,\varepsilon})$ 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 $\mathcal{C}^{\mathsf{TensorSum}}_{\cdot,\varepsilon}$ (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 $\mathcal{C}^{\mathsf{TensorSum}}_{\cdot,\varepsilon}$ 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.4), 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 3.12

In this section we complete the proof of Theorem 3.12, 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 3.15). 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 3.14 guarantees the existence of an $\mathcal{MAP}$ for $\mathsf{TensorSum}_{\mathbb{F},m,d,H}$ with proof complexity $(d+1)^\ell \cdot \log(|\mathbb{F}|)$ and query complexity $|H|^{m-\ell} \cdot dm^2 \log(|H|)$ for every $\ell \in [m]$. Thus, for every parameter $p \in \left\{(d+1)^i \cdot \log(|\mathbb{F}|) : i \in \mathbb{N}\right\}$ (which corresponds to the proof length), we set:

$$\ell = \frac{\log(p) - \log\log(|\mathbb{F}|)}{\log(d+1)}.$$

and apply Lemma 3.14. We obtain an $\mathcal{MAP}$ protocol for computing $\mathsf{TensorSum}_{\mathbb{F},m,d,H}$ with a proof of length

$$(d+1)^\ell \cdot \log(|\mathbb{F}|) = p$$

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 3.15, 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 Section 2.2.) 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 3.19. *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 3.19 was already shown in Theorem 3.12, and so, to prove Theorem 3.19, 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 3.20. *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 3.19 follows from Lemma 3.20 (and Lemma 3.15) 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 $|\mathbb{F}|^{1-\alpha}$ and set $d = 2(|H| - 1)$. We proceed to prove Lemma 3.20

Proof of Lemma 3.20. The first part of the protocol closely resembles the $\mathcal{MAP}$ that was presented in Lemma 3.14. 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 3.14, 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 3.20.1. *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 3.20.2. *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

$\mathcal{IPP}[3]$ 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.8) 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.6) 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

d and, by the Schwartz-Zippel Lemma, they can agree on at most a $\frac{d(m-\ell)}{|\mathbb{F}|} < 0.1$ fraction of their domain. Therefore, with probability 0.9 over the verifier's choice of $s_{\ell+1}, \ldots, s_m \in \mathbb{F}$, it holds that

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

Using the self-correction procedure, with probability at least 0.9, the verifier correctly obtains the value $z_{r_1, \ldots, r_\ell, s_{\ell+1}, \ldots, s_m} = P'(r_1, \ldots, r_\ell, s_{\ell+1}, \ldots, s_m)$. Hence, with probability at least $0.9^2 > 2/3$, the verifier rejects when testing whether $z_{r_1, \ldots, r_\ell, s_{\ell+1}, \ldots, s_m} = \tilde{W}(s_{\ell+1}, \ldots, s_m)$. □

By Claim 3.20.2, 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 3.20.1 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 3.20. □

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 3.21. *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 3.21, we yet again use the $\mathsf{TensorSum}$ problem. The first part of the theorem follows directly from Theorem 3.12 and the query complexity of property testers (which do not use a proof) is implied by Corollary 3.16.[16] Thus, to prove the theorem, all that remains is to show an $\mathcal{IPP}$ protocol for $\mathsf{TensorSum}$.

Lemma 3.22. *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 $O(dm \log |\mathbb{F}|)$, and query complexity $O(dm \cdot \mathsf{poly}(1/\varepsilon))$.*

Proof. The proof of Lemma 3.22 follows by adapting the well-known sum-check protocol of Lund *et al.* [LFKN92] to the settings of interactive proofs of proximity. Recall that the sum-check protocol is an interactive protocol that enables verification of the a claim of the form:

$$\sum_{x_1,\ldots,x_m \in H} P(x_1, \ldots, x_m) = 0.$$

where P is a low-degree polynomial. The difference between our setting and the classical setting of the sum-check protocol of [LFKN92] is that in the latter the verifier has explicit and direct access to P.[17] 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.5.)

We note that during the run of the $\mathcal{IPP}$ the prover sends m degree d univariate polynomials, and the verifier sends m elements in $\mathbb{F}$. Thus, the total communication complexity of the $\mathcal{IPP}$ is $O(dm \log |\mathbb{F}|)$. 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.

[16] 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.

[17] 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]).

$\mathcal{IPP}$ 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.8) 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.6) with soundness error 0.1.
5. $\mathcal{V}$ verifies that $z^* = \nu_m$.

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

Completeness. If $P \in$ 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 TensorSum. Let $\mathcal{P}^*$ be a cheating prover that attempts to convince the verifier of the false statement $P \in$ 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 3.22.1. *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 $\sum_{z \in H} P'_1(z) \neq 0$, 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 3.22.2. *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 3.22.2 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 3.22.1, 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 3.22.3. *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 3.22. □

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 $\mathcal{MAP}$ verifier can be implemented efficiently).

Theorem 4.1. *If the property Π has an $\mathcal{MAP}$ verifier that makes q queries and uses a proof of length p, then Π has a property tester that makes $\tilde{O}(2^p \cdot q)$ queries. Moreover, if the $\mathcal{MAP}$ 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 (and completeness) 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 error reduction 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.[18] 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

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

strings for V', and for each proof string $w \in \{0,1\}^p$ it emulates V' while feeding it the proof string w, the random string r, and the answer sequence $\bar{a}_r^f$. If for some string w the verifier accepts, then T accepts. Otherwise, it rejects.

If $f \in \Pi$, then there exists a proof string w that will make V' accept with probability at least $2/3$. The key point is that since the distribution of the queries does not depend on w. Hence, the queries actually made by T (using the dummy proof w_0) are identical to those V' would have made using the proof w (and the same randomness as T). Hence, T accepts in this case with probability at least $2/3$ (and in case V' has one-sided error, then T accepts with probability 1). On the other hand, similarly to the proof of Theorem 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 an amplification 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 $|R_n| \leq \exp\big(\mathsf{poly}(n)\big)$. 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 $|R_n| = \exp\big(\mathsf{poly}(n)\big)$ (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 An Extremely Hard Property for $\mathcal{MAP}$s

As noted in the introduction, every property has an $\mathcal{MAP}$ that uses a proof of length n and makes only $O(1/\varepsilon)$ queries (where the proof is simply the object itself). In contrast, in this section we show that for "almost all" properties Π, every $\mathcal{MAP}$ for Π that uses a proof that is even $n/100$ bits long, requires $\Omega(n)$ queries.

Our result is actually slightly stronger. Roughly speaking, we show that for every t, a random property of size 2^t can be tested (without a proof) using $O(t)$ queries, but any $\mathcal{MAP}$ that uses a proof of length even $t/100$ must make $\Omega(t)$ queries in order test this property.

In the following we consider properties that are sets of strings rather than functions. We note that a function formulation (as in Definition 2.1) can be easily obtained by mapping every string $x \in \{0,1\}^n$ to the function $f_x : [n] \to \{0,1\}$, defined as $f_x(i) = x_i$.

Theorem 5.1. *Let $t = t(n) < n/10$. Every property $\Pi = \cup_{n\in\mathbb{N}}\Pi_n$ (where $\Pi_n \subseteq \{0,1\}^n$) of size 2^t can be tested with $O(t/\varepsilon)$ queries (without using a proof), but for every $n \in \mathbb{N}$, for 99% of sets $\Pi_n \subseteq \{0,1\}^n$ of size 2^t, it holds that every $\mathcal{MAP}$ for testing $\varepsilon < 1/4$ proximity to Π_n that uses a proof of length p must make at least $t - p - O(\log n)$ queries.*

The rest of this section is devoted to the proof of Theorem 5.1, which is inspired by [GGR98, Section 4.1] and uses also ideas from [RVW13, Section 4]. We remark that while Theorem 5.1 holds for almost all properties, finding an *explicit* property for which a similar statement holds is an interesting open question.

The key idea in the proof of Theorem 5.1 is to show that $\mathcal{MAP}$s that use a relatively short proof and make relatively few queries can be represented by a small class of functions. Since this class of functions is small, we argue that a (small) *random* set $S \subseteq \{0,1\}^n$, viewed as a property, will fool every $\mathcal{MAP}$, in the sense that no $\mathcal{MAP}$ verifier can distinguish between a random element in S and a random element in $\{0,1\}^n$.

The foregoing intuition is formalized by the following lemma, which shows that there exists a set of *randomized decision trees* (see definition below) such that for every $\mathcal{MAP}$, there exists a subset of the decision trees such that the $\mathcal{MAP}$ accepts an input x (with probability at least 2/3) if and only if at least one of the randomized decision trees accepts x (with probability at least 2/3).

Lemma 5.2. *Let $\varepsilon \in (0,1/4)$. For every $n \in \mathbb{N}$ and for every $p, q \leq n$, there exists a class of functions $\mathcal{F}^{(n)}_{p,q}$ of size $2^{(\mathsf{poly}(n)\cdot 2^{p+q})}$ of functions from $\{0,1\}^n$ to $\{0,1\}$, such that the following holds. For every $\mathcal{MAP}$ verifier V for testing ε-proximity to $\Pi_n \subseteq \{0,1\}^n$ that uses a proof of length p and q queries, it holds that $I_V \in \mathcal{F}^{(n)}_{p,q}$, where $I_V(x)$ is defined as the indicator function for the event that there exists some $\pi \in \{0,1\}^p$ such that $\Pr[V^x(n,\varepsilon,\pi) = 1] \geq 2/3$.*

Note that the order of quantifiers in Lemma 5.2 is such that the class of functions is the same for *every* $\mathcal{MAP}$ verifier (and depends only on p and q). This will be crucial in showing that a random set fools *every* $\mathcal{MAP}$ verifier. Also note that if $p+q \ll n$, then the size of $\mathcal{F}$ is quite small relative to the class of all functions from $\{0,1\}^n$ to $\{0,1\}$ (which has size 2^{2^n}).

Proof of Lemma 5.2. To facilitate the proof of Lemma 5.2, it will be useful to describe standard testers (which do not use a proof) as *randomized decision trees*. Our main observation is that, roughly speaking, an $\mathcal{MAP}$ can be expressed as an OR of randomized decision trees.

Recall that a randomized decision tree is a model of computation for computing a randomized function $f : \{0,1\}^n \to \{0,1\}$. The randomized decision tree is a rooted ordered binary tree. Each

internal vertex of the tree is labeled with a value $i \in \{1,\ldots,n,*\}$ and the leaves of the tree are labeled with 0 or 1. (We think of a node that is labeled with $i \in [n]$ as representing the reading of the i^{th} bit, and of a node that is labeled with $*$ as representing a random coin toss.) Given an input $x \in \{0,1\}^n$, the decision tree is recursively evaluated as follows. If the root's label is $*$, then one of its two children is selected uniformly at random, and we recurse on that child. Otherwise (i.e., $i \in [n]$), if $x_i = 0$, then we recurse on the left subtree, and if $x_i = 1$, then we recurse on the right subtree. Once a leaf is reached, we output the label of that leaf and halt. If T is a randomized decision tree, we denote by $T(x)$ the (random variable that corresponds to) the output of T on input x.

The size of the decision tree is defined as the number of vertices in the tree, and the depth of the tree is defined as the longest path between the root of the tree and one of its leaves. (See [BdW02] for an extensive survey of decision tree complexity.) Let RDT_s be the set of all randomized decision trees of size s. For every $T_1,\ldots,T_t \in \mathsf{RDT}_s$ let $f_{T_1,\ldots,T_t} : \{0,1\}^n \to \{0,1\}$ be the function defined as $f_{T_1,\ldots,T_t}(x) = 1$ if and only if there exists $i \in [t]$ such that $\Pr[T_i(x) = 1] \geq 2/3$. Consider the class of functions

$$\mathcal{F}_{s,t} = \left\{ f_{T_1,\ldots,T_t} \,:\, T_1,\ldots,T_t \in \mathsf{RDT}_s \right\}.$$

We show that $\mathcal{F}_{\mathsf{poly}(n)\cdot 2^q, 2^p}$ satisfies the conditions of the lemma.

Let V be an $\mathcal{MAP}$ verifier of ε-proximity for Π_n that uses a proof of length p bits, q queries, and r random bits. The main observation is that for every fixed proof string $\pi \in \{0,1\}^p$, the (randomized) decision $V^x(n,\varepsilon,\pi)$ can be expressed as a randomized decision tree $T_{V,\pi}$ of depth $r+q$ (and size 2^{r+q}), which is defined as follows. The first r vertices in every path from the root to a leaf in the tree are labeled by $*$ (these vertices correspond to the random coin tosses of V). Every other internal vertex is labeled by some $i \in [n]$, corresponding to a query to x_i made by V. The two edges leaving every vertex, labeled by 0 and 1, correspond to the actual value of x_i, and these edges lead to a vertex that is labeled by the next query made by V, given the answer x_i to the query i. Given an input x and a random string $\rho \in \{0,1\}^r$, the leaf that is reached by evaluating the decision tree on input x and the random string ρ is labeled with the value $V^x(n,\varepsilon,\pi;\rho)$. (Recall that $V^x(n,\varepsilon,\pi;\rho)$ denotes the output of the verifier V given oracle access to x, direct access to n, ε, π and the random string ρ.) We are interested in $\Pr\left[V^x(n,\varepsilon,\pi) = 1\right]$.

Let $I_V : \{0,1\}^n \to \{0,1\}$ be defined as $I_V(x) = 1$ if and only if there exists $\pi \in \{0,1\}^p$ such that $\Pr[V^x(n,\varepsilon,\pi) = 1] \geq 2/3$. Since the randomized functions $V^x(n,\varepsilon,\pi)$ and $T_{V,\pi}(x)$ are identically distributed, it holds that $I_V \in \mathcal{F}_{2^{r+q},2^p}$.

By Lemma 4.6, we may assume without loss of generality that V has randomness complexity $r = O(\log n)$. The lemma follows by noting that $|\mathsf{RDT}_s| \leq (n+1)^s$ and therefore $|\mathcal{F}_{s,t}| \leq |\mathsf{RDT}_s|^t \leq (n+1)^{s\cdot t}$. □

Before proceeding to the proof of Theorem 5.1, we state a few standard propositions (Propositions 5.3, 5.4 and 5.6) whose proofs are deferred to Appendix B.1. We start by noting that sparse properties can be efficiently tested.

Proposition 5.3 (folklore)**.** *Every property $\Pi = \cup_{n\in\mathbb{N}}\Pi_n$ (where $\Pi_n \subseteq \{0,1\}^n$) can be tested by making $O(\log|\Pi_n|/\varepsilon)$ queries (without a proof).*

We note that Proposition 5.3 has standard proofs via learning theory techniques.[19] In Appendix B.1 we provide an alternative proof that uses the notion of $\mathcal{MAP}$s in a somewhat surprising,

[19] Either by an explicit reduction of property testing to learning (see [GGR98, Section 3]), or by applying Occam's razor directly to the testing problem.

but very natural way.

The following (standard) proposition shows that, with high probability, a random n-bit string will be far from any small subset of $\{0,1\}^n$.

Proposition 5.4 (folklore)**.** *For every constant $\varepsilon \in (0,1/4]$ and set $S \subseteq \{0,1\}^n$, it holds that $\Pr_{x\in_R\{0,1\}^n}[x \text{ is } \varepsilon\text{-close to } S] \leq |S| \cdot 2^{-n/8}$.*

For the last claim that we need, recall the definition of a PRG.

Definition 5.5. *A set $S \subseteq \{0,1\}^n$ is called a* pseudorandom generator (PRG) *for fooling a class $\mathcal{F}$ of functions from $\{0,1\}^n$ to $\{0,1\}$ if for every $f \in \mathcal{F}$ it holds that*

$$\left| \Pr_{x\in_R S}[f(x)=1] - \Pr_{x\in_R\{0,1\}^n}[f(x)=1] \right| < 1/10.$$

(note that the choice of the constant $1/10$ is arbitrary.)

The following (well-known) lemma shows that for every class of functions $\mathcal{F}$, a random set of size $O(\log|\mathcal{F}|)$ is a PRG that fools $\mathcal{F}$.

Proposition 5.6 (implicit in [GK92], see also [Gol08, Exercise 8.1])**.** *Let $\mathcal{F}$ be a class of functions from $\{0,1\}^n$ to $\{0,1\}$, of size at most $2^{2^{n/4}}$. Then, 99% of subsets of $\{0,1\}^n$ of size $s = O(\log|\mathcal{F}|)$ are PRGs that fool $\mathcal{F}$.*

We are now ready to prove Theorem 5.1.

Proof of Theorem 5.1. Fix $\epsilon \in (0,1/4)$. Let $t,p,q : \mathbb{N} \to \mathbb{N}$ be functions such that $t = t(n) < n/10$, $p = p(n) \leq n$, $q = q(n) \leq n$ and $t = p + q + O(\log n)$.

Fix $n \in \mathbb{N}$, and let $S_n \subseteq \{0,1\}^n$ be a random subset of $\{0,1\}^n$ of size $2^{t(n)}$. By Proposition 5.3, (for any choice of S) the property S can be tested using $O(\log(|S_n|)/\varepsilon) = O(t/\varepsilon)$ queries (without a proof).

Let $\mathcal{F}^{(n)}_{p,q}$ be the class of functions of size $2^{\mathsf{poly}(n)\cdot 2^{p+q}}$ guaranteed by Lemma 5.2, with respect to p and q. Since $O(\log|\mathcal{F}^{(n)}_{p,q}|) = O(2^{p+q}\cdot\mathsf{poly}(n)) = 2^t$, by Proposition 5.6 (applied to the class $\mathcal{F}^{(n)}_{p,q}$), with probability 0.99 over the choice of S_n, it holds that for every $f \in \mathcal{F}^{(n)}_{p,q}$:

$$\left| \Pr_{x\in_R S_n}[f(x)=1] - \Pr_{x\in_R\{0,1\}^n}[f(x)=1] \right| < 1/10. \tag{5.1}$$

Let S_n be a set for which Eq. (5.1) holds and assume toward a contradiction that there exists an $\mathcal{MAP}$ verifier V that uses a proof of length p and q queries, and tests ε-proximity to S_n.

By Lemma 5.2, it holds that $I_V \in \mathcal{F}^{(n)}_{p,q}$, where the function I_V is defined as $I_V(x) = 1$ if and only if there exists $\pi \in \{0,1\}^p$ such that $\Pr[V^x(n,\varepsilon,\pi)] \geq 2/3$. We proceed to show that I_V is a distinguisher for the PRG S_n, in contradiction to Eq. (5.1).

By the completeness of the $\mathcal{MAP}$, for every $x \in S_n$ it holds that $I_V(x) = 1$ and therefore

$$\mathop{\mathbf{E}}_{x\in_R S_n}[I_V(x)] = 1.$$

On the other hand, by the soundness of the $\mathcal{MAP}$, for every x that is ε-far from S_n it holds that $I_V(x) = 0$ and so

$$\mathop{\mathbf{E}}_{x\in_R\{0,1\}^n}[I_V(x)] \leq \mathop{\mathbf{E}}_{\substack{x \text{ that is}\\ \varepsilon\text{-far from } S_n}}[I_V(x)] + \Pr_{x\in_R\{0,1\}^n}\left[x \text{ is } \varepsilon\text{-close to } S_n\right] \leq |S_n|\cdot 2^{-n/8} \leq 2^{-\Omega(n)},$$

where the second inequality follows from Proposition 5.4 (and the fact that $I_V(x) = 0$ for every x that is ε-far from S_n), and the last inequality follows from our setting of $t \leq n/10$. Therefore,

$$\mathop{\mathbf{E}}_{x \in_R S_n}[I_V(x)] - \mathop{\mathbf{E}}_{x \in_R \{0,1\}^n}[I_V(x)] \geq 1 - 2^{-\Omega(n)},$$

in contradiction to Eq. (5.1). □

6 $\mathcal{MAP}$s for Parametrized Concatenation Problems

In this section we give a scheme for constructing efficient $\mathcal{MAP}$s for *parameterized concatenation problems*. For starters, we review the notion of (non-parameterized) concatenation problems: The k-concatenation problem of a property Π is defined as the property $\Pi^{\times k} \stackrel{\text{def}}{=} \{(x_1, \ldots, x_k) : \forall i \in [k],\ x_i \in \Pi \text{ and } |x_i| = |x_1|\}$. For every $i \in [k]$, we will refer to x_i as the i^{th} block or sub-input.

Concatenation problems (in the context of property testing) were recently studied by Goldreich [Gol13], who showed that the query complexity of the concatenation problem $\Pi^{\times k}$ (of a property Π) is roughly the same as the query complexity of the problem of testing a single instance of Π, regardless of the number of concatenations. More precisely, the query complexity of testing proximity of an input of length $n \cdot k$ (for $\Pi^{\times k}$) is the same, up to a polylogarithmic factor, as the query complexity of testing proximity of an input of length n (for Π), provided that the query complexity of Π increases at least linearly with $1/\varepsilon$ (which is typically the case).

We consider a generalization of the notion of a concatenation problem by allowing the underlying property to depend on some parameter, which may differ between the different blocks. Consider a family of properties $\{\Pi^\alpha\}_{\alpha \in A}$, where α is the parameter and A is some domain. As we shall show, some natural properties can be expressed as a concatenation $\Pi^{\alpha_1} \times \ldots, \Pi^{\alpha_k}$ of a property Π^α, with respect to different values of the parameter. For example, testing whether a given string x has Hamming weight w can be expressed as the question of testing whether x can be partitioned into k blocks such that the i^{th} block has Hamming weight w_i and $\sum_{i \in [k]} w_i = w$. (Other natural examples are reviewed below.)

In this section it will be convenient for us to view the input length $n \in N$, the proximity parameter $\varepsilon \in (0, 1)$, and the number of concatenations k as fixed. We note that although we fix n, ε, and k, these parameters should be viewed as generic, and so we allow ourselves to write asymptotic expressions such as $\mathsf{poly}(n)$, $\mathsf{poly}(\varepsilon)$, etc. If $\Pi \subseteq \{0,1\}^n$, then we say that a verifier V is an $\mathcal{MAP}(p, q)$ for Π with respect to proximity ε if V can distinguish between inputs that are in Π and inputs that are ε-far from Π using a proof of length p and q queries. (See the end of Section 6.1 for a discussion of the issues involved in providing a uniform treatment of parameterized concatenation problems.)

Additionally, throughout this section we study properties that are more naturally expressed as sets of strings (rather than functions), therefore we present them as such. Note that a function formulation (as in Definition 2.1) can be easily obtained by the (trivial) mapping that maps the string $x \in \Sigma^n$ to the function $f_x : [n] \to \Sigma$ defined as $f_x(i) = x_i$. We proceed to define parameterized concatenation problems.

Definition 6.1. *Let A be a finite set, and $n, k, n/k \in \mathbb{N}$. For every $\alpha \in A$, let $\Pi^\alpha_{n/k} \subseteq \{0,1\}^{n/k}$ be a property of n/k-bit strings that is parameterized by α. For every subset $\bar{A} \subseteq A^k$, we say that the*

property $\Pi_n^{\bar{A}}$ *is a* parameterized k-concatenation property *(of* n*-bit strings), where* $\Pi_n^{\bar{A}}$ *is defined as*

$$\Pi_n^{\bar{A}} \stackrel{\text{def}}{=} \bigcup_{(\alpha_1,\ldots,\alpha_k)\in\bar{A}} \Pi_{n/k}^{\alpha_1} \times \ldots \times \Pi_{n/k}^{\alpha_k}.$$

If we consider the task of testing $\Pi_n^{\bar{A}}$, it is not a priori clear (for the tester) what value of the parameter α_i to use for each block. This is where $\mathcal{MAP}$s can help us. That is, the proof of proximity will simply tell the $\mathcal{MAP}$ verifier the correct value of the parameter for each block. Using this idea, in Section 6.1 we construct an $\mathcal{MAP}$ for any parameterized concatenation problem. In Sections 6.2 to 6.3, we demonstrate the applicability of this technique by using it to construct efficient $\mathcal{MAP}$s (which manage to bypass some lower bounds for testers that do not use a proof) for a couple of natural properties:

1. **Approximate Hamming weight:** The first application of our scheme is an efficient $\mathcal{MAP}$ for the problem of approximating the Hamming weight of a given string. In this problem, which is parameterized by $w \in [n]$, the tester needs to distinguish between inputs that have Hamming weight exactly w and those that have Hamming weight $\notin [w - \varepsilon n, w + \varepsilon n]$.

   We complement this $\mathcal{MAP}$ with a (non-tight) *lower bound* on the $\mathcal{MAP}$ complexity of the approximate Hamming weight property. We leave the question of resolving the gap between the upper and lower bounds to future work. See Section 6.2.

2. **Graph orientation problems:** In addition, we show an $\mathcal{MAP}$ in the graph orientation model (see Section 6.3 for details on this model). Specifically, our $\mathcal{MAP}$ distinguishes between orientations (of a specific undirected graph) that are Eulerian and those that are far from Eulerian. Our $\mathcal{MAP}$ has lower query complexity than the best possible property tester for this problem, and the gap in query complexity increases with the size of the proof. See Section 6.3.

Properties with/without distance. Note that all of the explicit properties studied in Section 3 are properties of low-degree polynomials and error-correcting codes. The $\mathcal{MAP}$s that we have shown for these properties crucially relied on the fact that these properties have *distance* (i.e., properties wherein every two objects that have the property are far from each other), and moreover, they allow for a local form of self-correction.[20] We note that in contrast, all of the properties that we study in this section are without distance (as is the property of bipartiteness studied in Section 7). For example, the Hamming weight property is without distance since there are pairs of strings at distance 2 that have the same Hamming weight.

6.1 The Generic Scheme

In this section we show a generic scheme for parameterized concatenation problems.

Theorem 6.2. *Let* $c_1, c_2 \geq 0$ *be constants. Let* $\Pi_n^{\bar{A}}$ *be a parameterized* k*-concatenation property (of* n*-bit strings) with respect to* A*,* $\bar{A}$*, and* $\{\Pi_{n/k}^{\alpha}\}_{\alpha\in A}$*, as in Definition 6.1. Suppose that for every* $\alpha \in A$*, the property* $\Pi_{n/k}^{\alpha}$ *can be tested with respect to any proximity parameter* $\varepsilon' > 0$ *(without*

[20] 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.

using a proof) with query complexity $O\left((n/k)^{c_1} \cdot (\varepsilon')^{-c_2}\right)$*. Then, the property* Π *has an* $\mathcal{MAP}$*, with respect to proximity parameter* ε*, that uses a proof of length* $k \cdot \log|A|$ *and has query complexity:*

$$\begin{cases} \tilde{O}\left((n/k)^{c_1} \cdot \varepsilon^{-\max(1,c_2)}\right) & \text{if } c_1 > 0 \text{ and } c_2 \geq 0 \\ \tilde{O}\left((n/k)^{1-1/c_2} \cdot \varepsilon^{-1}\right) & \text{if } c_1 = 0 \text{ and } c_2 \geq 1. \end{cases}$$

Furthermore, if the testers for $\{\Pi^{\alpha}_{n/k}\}_{\alpha \in A}$ *have a one-sided error, then the resulting* $\mathcal{MAP}$ *has a one-sided error.*

Proof. The key idea is to use the proof in order to "break" the problem of testing property Π into the concatenation problem of testing several sub-properties with smaller inputs. Then, instead of solving each sub-problem independently, we efficiently verify that the (smaller) sub-inputs together are not too far from their corresponding sub-properties.

More specifically, we partition the input x (of length n) into k blocks $x_1, \ldots, x_k$ of length n/k each. If $x \in \Pi^{\bar{A}}_n$, then there must exist $(\alpha_1, \ldots, \alpha_k) \in \bar{A}$ such that $x_i \in \Pi^{\alpha_i}_{n/k}$ for each $i \in [k]$. The proof is simply $(\alpha_1, \ldots, \alpha_k)$; that is, the "hidden" parameter for each sub-property. The verifier, given this alleged proof, checks that indeed $(\alpha_1, \ldots, \alpha_k) \in \bar{A}$ (i.e., the parameterization of the sub-properties is valid), and is then left with the task of ascertaining that the k blocks are not "far" from $\Pi^{\alpha_1}_{n/k} \times \cdots \times \Pi^{\alpha_k}_{n/k}$.

Toward this end, similarly to the approach in [Gol13, Section 5], we note that given an input that is far from $\Pi^{\alpha_1}_{n/k} \times \cdots \times \Pi^{\alpha_k}_{n/k}$, the distance from the property can be either "spread" between all of the sub-inputs, or "concentrated" on a few sub-inputs — or anything in between. The main idea is that if the distance is "concentrated", then the deviation in these sub-inputs must be large, and so, we can detect that such particular sub-inputs do not have their corresponding sub-property by using a test with low query complexity. Since we only read a few bits for this test, we can afford to run it on many sub-inputs (thereby increasing our chance of catching a sub-input that is far from its corresponding sub-property). On the other hand, if the distance is "spread" among the sub-inputs, then it suffices to examine only a few sub-inputs, but for each such sub-input, we need to run a test with high query complexity. Interestingly, in the latter case it is sometimes beneficial for the verifier to simply read the entire block rather than to run the "expensive" tester.

Since the verifier does not know whether it is in one of the extreme situations or anywhere in between, naively we might want to consider the "worst of all worlds" (i.e., small spread and high query complexity per block). We improve upon the performance of the forgoing approach by using the precision sampling technique (originating in Levin [Lev87, last paragraph of Section 9], see also [Gol13, Appendix A.2]), which allows us to deal with all of the possible distributions of the distance economically (specifically, by considering only a logarithmic number of representative distributions). The resulting $\mathcal{MAP}$ protocol for parameterized concatenation problems is presented in Figure 5.

Note that the length of the proof, which is $(\alpha_1, \ldots, \alpha_k)$, is bounded by $k \cdot \log|A|$. As for the query complexity, first recall that for any α and $\varepsilon' > 0$, the property $\Pi^{\alpha}_{n/k}$ has a tester with query complexity $T(n/k, \varepsilon') = (n/k)^{c_1} \cdot (\varepsilon')^{-c_2}$. Thus, the total number of queries is at most:

$$\begin{aligned} O\left(\sum_{j \in [\lceil \log_2 2/\varepsilon \rceil]} \frac{\log(1/\varepsilon)}{2^j \varepsilon} \cdot \log(1/\varepsilon) \cdot T\left(n/k, 2^{-j}\right)\right) &= \tilde{O}\left(\frac{(n/k)^{c_1}}{\varepsilon} \sum_{j \in [\lceil \log_2(2/\varepsilon) \rceil]} 2^{j(c_2-1)}\right) \\ &= \tilde{O}\left((n/k)^{c_1} \, \varepsilon^{-\max(1,c_2)}\right). \end{aligned}$$

$\mathcal{MAP}$ for the parameterized k-concatenation problem $\Pi_n^{\bar{A}}$

Input: a proximity parameter $\varepsilon > 0$ and oracle access to a string $x \in \{0,1\}^n$.

The Proof:

- The string x is interpreted as a k sub-inputs $x = (x_1, \ldots, x_k) \in (\{0,1\}^{n/k})^k$.
- The proof consists of the parameters for the concatenated problems; namely, the values $(\alpha_1, \ldots, \alpha_k)$ such that $x_i \in \Pi_{n/k}^{\alpha_i}$, for every $i \in [k]$ (such values must exist for $x \in \Pi_n^{\bar{A}}$).

The Verifier:

1. If $(\alpha_1, \ldots, \alpha_k) \notin \bar{A}$, then reject.
2. For every $j \in [\lceil \log_2(2/\varepsilon) \rceil]$, perform the following test:
   (a) Select uniformly at random $O\left(\frac{\log(1/\varepsilon)}{2^j \varepsilon}\right)$ indices in $[k]$. Denote the chosen indices by I.
   (b) For every $i \in I$: Run the $\Pi_{n/k}^{\alpha_i}$ tester $O(\log(1/\varepsilon))$ times on input x_i, with respect to proximity parameter 2^{-j}. Reject if the majority of the tests failed.
3. If all of the previous tests passed, then accept.

Figure 5: $\mathcal{MAP}$ for Π

For the special case in which $c_1 = 0$, we tighten the analysis. Observe that, without loss of generality, for any proximity parameter ε, it holds that $T(n, \varepsilon) \leq n$ (simply since the tester can always just read the entire input). Therefore, the query complexity is bounded in this case by:

$$O\left(\sum_{j \in [\lceil \log_2 2/\varepsilon \rceil]} \frac{\log(1/\varepsilon)}{2^j \varepsilon} \cdot \log(1/\varepsilon) \cdot T\left(n/k, 2^{-j}\right)\right) = \tilde{O}\left(\frac{1}{\varepsilon} \sum_{j \in [\lceil \log_2 2/\varepsilon \rceil]} \min\left(\frac{n/k}{2^j}, 2^{j(c_2-1)}\right)\right)$$
$$\leq \tilde{O}\left(\frac{1}{\varepsilon} \sum_{j \in [\lceil \log_2 2/\varepsilon \rceil]} (n/k)^{1-1/c_2}\right),$$

where the last inequality follows from the fact that $c_2 \geq 1$ (by our assumption) and thus $\min\left(n/k \cdot 2^{-j}, 2^{(c_2-1)j}\right) \leq (n/k)^{1-1/c_2}$. Therefore, the total query complexity in this case is $\tilde{O}\left((n/k)^{1-1/c_2} \cdot \varepsilon^{-1}\right)$.

We proceed to prove the completeness and soundness of the protocol.

Completeness. Suppose that $x \in \Pi_n^{\bar{A}}$ and that $(x_1, \ldots, x_k) \in \Pi_{n/k}^{\alpha_1} \times \ldots \times \Pi_{n/k}^{\alpha_k}$. The tester for each sub-property is invoked $O(\log(1/\varepsilon))$ times in Step (2b) on some $x_i \in \Pi_{n/k}^{\alpha_i}$. Therefore, with probability $1 - \mathsf{poly}(\varepsilon)$ the majority of these invocations will accept. The total number of times that this step is run is at most $O(1/\varepsilon \cdot \log^2(1/\varepsilon))$ and therefore, by the union bound, the $\mathcal{MAP}$ verifier accepts with probability at least $2/3$.

Soundness. Suppose that $x \in \{0,1\}^n$ is ε-far from $\Pi_n^{\bar{A}}$. Let $(\alpha_1, \ldots, \alpha_k) \in \bar{A}$ be an alleged proof for the false statement $x \in \Pi_n^{\bar{A}}$ (notice that if $(\alpha_1, \ldots, \alpha_k) \notin \bar{A}$, then the tester immediately rejects).

Thus, $x = (x_1, \ldots, x_k) \in (\{0,1\}^{n/k})^k$ is ε-far from $\Pi_{n/k}^{\alpha_1} \times \ldots \times \Pi_{n/k}^{\alpha_k}$ (since otherwise x is ε-close to $\Pi_n^{\bar{A}}$).

The following claim shows that it suffices to consider $O(\log(1/\varepsilon))$ different distributions of the distance between the sub-inputs. Since the proof of the claim is similar to results of [Gol13, Section 5], we defer it to Appendix B.2).

Claim 6.2.1 (Precision Sampling (cf. [Lev87, last paragraph of Section 9] or [Gol13, Appendix A.2]))**.** *There exists $j \in [\lceil \log_2 2/\varepsilon \rceil]$ such that a $\frac{2^j \varepsilon}{4 \cdot \lceil \log_2(2/\varepsilon) \rceil}$ fraction of $x_1, \ldots, x_k$ are 2^{-j}-far from their corresponding sub-properties $\Pi_{n/k}^{\alpha_1}, \ldots, \Pi_{n/k}^{\alpha_k}$.*

Consider the execution of iteration j, where j is the index guaranteed by Claim 6.2.1. In this iteration, since the verifier selects uniformly at random $O\left(\frac{\log(1/\varepsilon)}{2^j \varepsilon}\right)$ indices in $[k]$, with probability at least 0.9, it selects at least one $i \in [k]$ such that x_i is 2^{-j}-far from Π^{α_i}.

Suppose that such an i is indeed selected. Since the base tester for $\Pi_i^{\alpha_i}$ is run with respect to proximity 2^{-j}, it will reject x_i with probability 2/3. Since the test is repeated $O(\log(1/\varepsilon))$ times, the majority of these tests will reject with probability at least 0.9. Thus, the $\mathcal{MAP}$ verifier rejects x with probability at least $0.9 \cdot 0.9 \geq 2/3$. □

On providing a uniform treatment. Recall that throughout this section we have fixed n, ε and k. Before proceeding to describe the applications of Theorem 6.2, we shortly discuss issues that arise when considering a uniform (asymptotic) treatment. In some cases, in order to optimize the total complexity (i.e., the sum of the proof complexity and the query complexity) of the $\mathcal{MAP}$ in Theorem 6.2, it is beneficial to allow the number k of concatenations to depend on the proximity parameter ε. However, if k depends on ε, then the following two issues arise.

First, notice that if k depends on ε, then the proof string in Theorem 6.2 becomes dependent on ε too, and therefore this protocol does not fall in our definition of $\mathcal{MAP}$ (Definition 2.1), which requires a single proof of proximity that works for *every* value of $\varepsilon > 0$. Hence, one can consider a slight relaxation of Definition 2.1 in which we allow the proof of proximity to depend on ε. Since formally such a protocol is not an $\mathcal{MAP}$, we call it an $\mathcal{MAP}_{\mathsf{PDP}}$ (where PDP stands for *proximity dependent proofs*). Note that in an $\mathcal{MAP}_{\mathsf{PDP}}$ both the contents of the proof of proximity, *and its length* may depend on the proximity parameter. See Section 2.1 for further discussion of $\mathcal{MAP}_{\mathsf{PDP}}$.

An additional issue that arises when the number of concatenations k depends on ε is that it is unclear how to define a k-concatenation property, as the naive definition that follows Definition 6.1 would make the property itself depend on k, and therefore also on the proximity parameter. While this issue can be overcome for the specific properties that are studied below, doing so in general would be extremely cumbersome, which is the main reason for our non-uniform treatment.

6.2 Approximate Hamming Weight

In this section we consider the problem of deciding whether a given string $x \in \{0,1\}^n$ has Hamming weight approximately w. More specifically, we would like a tester that accepts every string $x \in \{0,1\}^n$ that has Hamming weight $w \in [n]$, and rejects strings that have Hamming weight that is ε-far from having weight w. Namely, the tester should reject every string $x \in \{0,1\}^n$ for which $\mathsf{wt}(x) \notin [w - \varepsilon n, w + \varepsilon n]$, where $\mathsf{wt}(x)$ denotes the Hamming weight of x.

More formally, we consider a family of properties $\{\mathsf{Hamming}_n^w\}_w$, indexed by a weight $w \in \{0,\ldots,n\}$. The property $\mathsf{Hamming}_n^w$ is defined as the set that consists of all strings $x \in \{0,1\}^n$ that have Hamming weight exactly w.

By well-known sampling lower bounds (see, e.g., [BYKS01, Theorem 15], improving upon [CEG95]), the query complexity of any property tester (which does not use a proof) is $\Omega\big(\min\left(n, \varepsilon^{-2}\right)\big)$. 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 $\mathcal{IPP}$ for $\mathsf{Hamming}^w$ with complexity $\tilde{O}\left(\varepsilon^{-1}\right)$ and a *2-message* $\mathcal{IPP}$ with complexity $\tilde{O}\left(n^{\frac{1}{3}} \cdot \varepsilon^{-\frac{2}{3}}\right)$. (Note that for $\varepsilon = 1/\sqrt{n}$, the 2-message protocol of [RVW13] has *sublinear* complexity of $\tilde{O}\left(n^{2/3}\right)$, whereas testing without a proof requires $\Omega(n)$ queries.)

Using Theorem 6.2, we show that the performance of the [RVW13] 2-message $\mathcal{IPP}$ can be matched by an $\mathcal{MAP}$ (i.e., a 1-message $\mathcal{IPP}$), while essentially preserving its complexity.[21] 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 slightly stronger result. Namely, that for every $k \in [n]$ 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 obtained by setting $k = n^\alpha$). In order to minimize the total complexity (i.e., the sum of the proof complexity and

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

the query complexity) of the $\mathcal{MAP}$, we also consider $\mathcal{MAP}_{\mathsf{PDP}}$ verifiers (recall that $\mathcal{MAP}_{\mathsf{PDP}}$ is a slight relaxation of our definition of $\mathcal{MAP}$ that allows the proof of proximity to depend on the proximity parameter, see the discussion at the end of Section 6.1). With this relaxation, we can set $k = n^{\frac{1}{3}} \cdot \varepsilon^{-\frac{2}{3}}$ to obtain an $\mathcal{MAP}_{\mathsf{PDP}}$ with (total) complexity $\tilde{O}\left(n^{\frac{1}{3}} \cdot \varepsilon^{-\frac{2}{3}}\right)$. See further discussion in Section 2.1.

We complement the foregoing upper bound by showing a *lower bound* on the $\mathcal{MAP}$ complexity of $\mathsf{Hamming}$. Specifically, we show that every $\mathcal{MAP}$ for $\mathsf{Hamming}$ that uses a proof of length $p \geq 1$ must use $\Omega\left(\frac{\min(n,\varepsilon^{-2})}{p}\right)$ queries. Note that the two bounds do not match (e.g., for $\varepsilon = 1/\sqrt{n}$ and $p = n^{2/3}$, the upper bound is $\tilde{O}\left(n^{2/3}\right)$ and the lower bound is $\Omega(n^{1/3})$). We leave the question of resolving this gap for future work.

Theorem 6.3. *For every $w \in \{0, \ldots, n\}$, the property $\mathsf{Hamming}_n^w$ has a (two-sided error) $\mathcal{MAP}$, with respect to proximity parameter ε, that uses a proof of length $k \cdot \log n$ and $\tilde{O}\left(\sqrt{n/k} \cdot \varepsilon^{-1}\right)$ queries.*

We remark that by applying Theorem 4.3 to the $\mathcal{MAP}$ of Theorem 6.3, we can (somewhat surprisingly) construct a *one-sided error* $\mathcal{MAP}$ with proof complexity $O(k \log n + \mathsf{polylog} n)$ and query complexity $\tilde{O}\left(\sqrt{n/k} \cdot \varepsilon^{-1}\right)$. In contrast, the query complexity of every one-sided error property tester for $\mathsf{Hamming}_n^w$ (without a proof) is *linear* in the input size.

Proof of Theorem 6.3. Fix $w \in [n]$. It is well-known (and easy to show, e.g., via the Chernoff bound) that ε-proximity to $\mathsf{Hamming}_n^w$ can be tested, without a proof, using $O(\varepsilon^{-2})$ queries (with a two-sided error). Let

$$\bar{A} \stackrel{\text{def}}{=} \left\{(w_1, \ldots, w_k) \in \{0, \ldots, n/k\}^k : \sum_{i=1}^{k} w_i = w\right\}.$$

Observe that a string $x = (x_1, \ldots, x_k) \in (\{0,1\}^{n/k})^k$ has Hamming weight w if and only if, for every $i \in [k]$ the string x_i has Hamming weight w_i and $\sum_{i=1}^{k} w_i = w$. Hence,

$$\mathsf{Hamming}_n^w = \bigcup_{(w_1,\ldots,w_k)\in\bar{A}} \mathsf{Hamming}_{n/k}^{w_1} \times \ldots \times \mathsf{Hamming}_{n/k}^{w_k}.$$

The theorem follows from Theorem 6.2 (where $c_1 = 0$ and $c_2 = 2$). □

Relation to $\mathsf{TensorSum}$. The $\mathsf{Hamming}$ 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 $\mathsf{Hamming}$ problem we do not have the benefit of an error-correcting code but allow an approximate answer.

Next, we show a lower bound on the $\mathcal{MAP}$ complexity of the property $\mathsf{Hamming}_n^{n/2}$ (the set of all strings of Hamming weight exactly $n/2$, where n is the length of the string). We note that the lower bound can be extended to $\mathsf{Hamming}_n^w$ for more general values of w by reducing to $\mathsf{Hamming}_n^{n/2}$

using adequate padding (while taking care of the integrality issues that arise). We also note that the lower bound only holds for reasonable complexity measures (which are specified formally below).

The lower bound is proved using our extension of the [BBM11] framework to the $\mathcal{MAP}$ 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 $\mathcal{MAP}_{\mathsf{PDP}}$.

One natural candidate for a communication complexity problem on which we can base our Hamming lower bound is the *Hamming Distance* communication problem, wherein Alice and Bob need to decide whether the Hamming distance of their input strings is equal to a predetermined number. However, as opposed to the $\mathcal{MAP}$ lower bounds that we have shown before (e.g., for TensorSum, and EIM), Hamming is a property of non-robust objects; i.e., there is no significant distance between every pair of valid objects. In order to overcome the lack of distance between valid objects in Hamming, we wish to reduce Hamming to an $\mathcal{MA}$ 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 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). Formally,

Definition 6.4. *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 $\mathsf{GHD} \stackrel{\text{def}}{=} \mathsf{GHD}_{n,\frac{n}{2},\sqrt{n}}$.

We use the following lemma, which can be derived from a recent result of Gur and Raz [GR13b] by observing that the reductions of [CR11] are robust to $\mathcal{MA}$.

Lemma 6.5. *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* $\Omega\left(\frac{\min\left(n,(n/g)^2\right)}{p}\right)$.

Equipped with Lemma 6.5, we proceed to prove the lower bound for $\mathsf{Hamming}_n^w$.

Theorem 6.6. *For every* $n \in \mathbb{N}$ *and* $\varepsilon \stackrel{\text{def}}{=} \varepsilon(n) \in (0, 1/2)$, *if* $\mathsf{Hamming}_n^{n/2}$ *has an* $\mathcal{MAP}$ *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 6.6. Throughout the proof we fix the function w as $w(m) \stackrel{\text{def}}{=} m/2$. By Lemma 3.17, if $\mathsf{Hamming}_n^w \in \mathcal{MAP}(p,q)$, 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 6.5, the theorem follows by reducing $\mathsf{GHD}_{n,n/2-\varepsilon n,\varepsilon n}$ to $\mathcal{C}_{\oplus,\varepsilon}^{\mathsf{Hamming}_w}$, which is done next. (We stress that this reduction takes place entirely in the context of $\mathcal{MA}$ 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 $\mathsf{Hamming}_w$ is in the class $\mathcal{MAP}(p,q)$, where p and q are as in the hypothesis. Then, by Lemma 3.17, 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 6.5, it holds that $p \cdot q = \Omega\left(\min(n, \varepsilon^{-2})\right)$. □

6.3 Graph Orientation Problems

In this section we apply Theorem 6.2 to the problem of testing graph orientations for being Eulerian in the *graph orientation* model. In the graph orientation model, introduced by Halevy *et al.* [HLNT05],

an underlying *directed* graph $G = (V, E)$ with a canonical orientation (i.e., wherein each edge is directed from the vertex with the smaller lexicographical order to the vertex with the larger lexicographical order) is given as an explicit input to the tester, and the actual input, to which the tester only has oracle access, is an orientation $\overrightarrow{G} = \{ d(e) \in \{0,1\} : e \in E \}$ of G, wherein $d(e)$ represents the direction of the edge e.

Given a property Π_G (parameterized by the fixed directed graph G) of graph orientations, a tester for Π_G is given query access to an orientation of G; that is, every query is an edge $e \in E$, and the answer to the query is the direction of e in G (i.e., $d(e) \in \{0,1\}$). An orientation $\overrightarrow{G}$ of G is ε-close to Π_G if it can be modified to be in Π_G by inverting the direction of at most an ε-fraction of the edges of G. Note that the distance function in the orientation model naturally depends on the size of the underlying graph. Moreover, the testing algorithm may strongly depend on the structure of the underlying graph. We note that the graph orientation model falls within the standard property testing framework, as a special case of property testing of *massively parameterized* problems (see [New10] for a survey on massively parameterized properties).

We consider the graph orientation property of being *Eulerian*, which was first pointed out by Halevy *et al.* [HLNT07] as a natural property for the graph orientation model. Recall that a directed graph is Eulerian if for every vertex v in the graph, the in-degree of v is equal to its out-degree. If G is a directed graph (with canonical orientation), we denote by Euler_G the property that contains all orientations of G to (directed) Eulerian graphs. While no (non-trivial) upper bound is known for this property, Fischer *et al.* [FLM+12] showed that for general graphs, testing proximity to being Eulerian with 1-*sided error* is hard. Specifically, they showed that for $G = K_{2,n-2}$ (i.e., the full bipartite graph with 2 vertices on one side, and $n-2$ vertices on the other side), a one-sided error tester for Euler_G must use $\Omega(n)$ queries.

Using Theorem 6.2 we show, for every $\alpha \in (0,1]$, an $\mathcal{MAP}$ with 1-sided error for $\mathsf{Euler}_{K_{2,n-2}}$, which uses a proof of length $\tilde{O}(n^{\alpha})$ and $\tilde{O}(n^{1-\alpha}\varepsilon^{-1})$ queries. Hence, we have a smooth (up to poly-logarithmic factors) multiplicative trade-off between the query and proof complexities of the $\mathcal{MAP}$. We note that it seems that using similar techniques, it is possible to obtain, using Theorem 6.2, efficient $\mathcal{MAP}$s for several problems in the graph orientation model.

Formally, let $K_{2,n-2}$ be the graph with a set of vertices $V = \{ v_1, ..., v_n \}$ and a set of edges $E = \{ (v_i, v_j) : i \in \{1,2\},\ j \in \{3, ..., n\} \}$.

Theorem 6.7. *The property* $\mathsf{Euler}_{K_{2,n-2}}$ *has a one-sided error* $\mathcal{MAP}$*, with respect to proximity parameter* ε*, that uses a proof of length* $O(k \cdot \log n)$ *and has query complexity* $\tilde{O}\left(\frac{n}{k} \cdot \varepsilon^{-1}\right)$.

Proof. The main idea is to divide $K_{2,n-2}$ into sub-graphs of equal size, wherein v_1 and v_2 are the only vertices that appear in all sub-graphs. We require that for all $j \in \{3, \ldots, n\}$, the in-degree of v_j is equal to its out-degree. However, since v_1 and v_2 appear in all of the sub-graphs, we can allow their in-degree in each subgraph to be different than their out-degree in this subgraph, as long as the sum of their in-degrees is equal to the sum of their out-degrees.

We denote the in-degree of a vertex $v \in K_{2,n-2}$ by $d_{in}(v)$ and the out-degree of $v \in K_{2,n-2}$ by $d_{out}(v)$. We start by considering the following generalization of the $\mathsf{Euler}_{K_{2,n-2}}$ property. For every $a, b \in \mathbb{Z}$, let $\mathsf{Euler}^{(a,b)}_{K_{2,n-2}}$ be the set of all orientations of $K_{2,n-2}$ such that:

1. $d_{in}(v_1) - d_{out}(v_1) = a$.
2. $d_{in}(v_2) - d_{out}(v_2) = b$

3. $d_{in}(v_j) = d_{out}(v_j)$, for all $j \in \{3, \ldots, n\}$.

(note that a and b may be negative). Let $\bar{A}$ be the set of all sequences $\big((a_1, b_1), \ldots, (a_k, b_k)\big)$, where $a_i, b_i \in \{-(n-2), \ldots, n-2\}$ for every $i \in [k]$ and for which it holds that $\sum_{i=1}^{k} a_i = 0$ and $\sum_{i=1}^{k} b_i = 0$. Consider the property:

$$\Pi \stackrel{\text{def}}{=} \bigcup_{(a_1,b_1),\ldots,(a_k,b_k)\in\bar{A}} \mathsf{Euler}^{(a_1,b_1)}_{K_{2,n/k-2}} \times \ldots \times \mathsf{Euler}^{(a_k,b_k)}_{K_{2,n/k-2}}.$$

This property contains all sequences of k orientations of the graphs $K_{2,n/k-2}$ such that (1) the vertices on the "large" side have in degree that is equal to their out degree and (2) for the vertices on the "small" sides, the sum, over all graphs, of their in-degree equals the sum of their out-degrees. We note that there is a trivial mapping between Π and $\mathsf{Euler}_{K_{2,n-2}}$ which simply identifies the pair of vertices on the smaller side of graphs in Π as a single pair of vertices.

By applying Theorem 6.2 with $c_1 = 1$, $c_2 = 0$, and using the trivial tester (that queries the entire orientation) for every subgraph, the property Π has an $\mathcal{MAP}$ with proof of length $O(k \cdot \log n)$, and query complexity $\tilde{O}\left(\frac{n}{k} \cdot \varepsilon^{-1}\right)$. By the foregoing discussion, this $\mathcal{MAP}$ can be easily modified to work also for the property $\mathsf{Euler}_{K_{2,n-2}}$. □

7 Bipartiteness in Bounded Degree Graphs

In this section we consider the problem of testing *bipartiteness* for "rapidly-mixing" graphs in the bounded-degree graph model. In a classical result, Goldreich and Ron [GR99] showed that *any* graph can be tested for bipartiteness in the bounded-degree model, using a tester with query complexity $\tilde{O}(\sqrt{N}/\varepsilon)$, where N is the number of vertices in the tested graph. Goldreich and Ron first consider the (far simpler) case in which there is a promise that the graph is "rapidly-mixing" (see definition below). More recently, Rothblum, Vadhan and Wigderson [RVW13] showed a 2-message $\mathcal{IPP}$ for bipartiteness, in the rapidly-mixing case, with communication and query complexities that are $\mathsf{poly}(\log N, \varepsilon^{-1})$.

Roughly speaking, using similar techniques to (the rapidly-mixing case in) [GR99], we construct an $\mathcal{MAP}$ protocol for testing bipartiteness of rapidly-mixing graphs, with proof complexity p and query complexity q for every p and q such that $p \cdot q \geq N$. Thus, the query complexity of our $\mathcal{MAP}$ improves upon that of the [GR99] bipartiteness tester (which does not use a proof) only if the proof is of length $\omega(\sqrt{N})$. In particular, we obtain an $\mathcal{MAP}$ verifier that uses a proof of length $N^{2/3}$ and makes only $N^{1/3}$ queries. In contrast, a lower bound of $\Omega(\sqrt{N})$ for testers (which do not use a proof) was shown by Goldreich and Ron [GR02] (and this lower bound holds also in the rapidly-mixing case).

We leave the questions of (1) extending our result to graphs that are not rapidly-mixing, and (2) obtaining an $\mathcal{MAP}$ for bipartiteness with query and proof complexities that are both $o(\sqrt{N})$, for future research.

The Bounded Degree Graph Model. In the bounded degree graph model, introduced by Goldreich and Ron [GR02] (see also [Gol11]), the object that is being tested is a graph $G = (V, E)$ with degree bounded by some constant d. The graph is represented by a function $g : V \times [d] \to V \cup \{\perp\}$ such that $g(u, i) = v$ if v is the i^{th} vertex incident at u, and $g(u, i) = \perp$ if u has less than i neighbors. The distance between two graphs, represented by functions $g, g' : V \times [d] \to V \cup \{\perp\}$

is measured (as usual) as the fraction of pairs (u,i) such that $g(u,i) \neq g'(u,i)$. For further details, see [Gol11].

Rapidly-Mixing Graphs. Let $G=(V,E)$ be graph with degree bounded by d and let $N \stackrel{\text{def}}{=} |V|$. A (lazy) random walk of length ℓ starting at a vertex $s \in V$ is a random walk that involves ℓ steps. At each step, if the walk is currently at vertex v with degree $d_v \leq d$, then the walk continues to each neighbor of v with probability $1/2d$ and stays at v with probability $1-\frac{d_v}{2d} \geq 1/2$ (a so-called "lazy" step). We say that G is rapidly-mixing if for every $s,t \in V$, the probability that a (lazy) random walk of length $\Omega(\log N)$ that starts in s ends in t, is at least $1/(2N)$ and at most $2/N$. We will use the fact that in a rapidly-mixing graph $G=(V,E)$, for every vertex $s \in V$ and subset $T \subseteq V$, the probability that a random walk of length $\Omega(\log N)$ that starts at s ends in T, is at least $|T|/(2N)$ and at most $2|T|/N$. We mention the well-known fact that expander graphs are rapidly-mixing.

We proceed to describe our $\mathcal{MAP}$. Actually since we require a promise that the graph is rapidly-mixing, we will need a "promise-problem" variant of the notion of $\mathcal{MAP}$. For sake of brevity we only define this notion implicitly (in the next theorem).

Theorem 7.1. *There exists a probabilistic verifier V that given oracle access to a graph G of size N (in the bounded degree model), and explicit access to N, the degree bound d, a proximity parameter $\varepsilon \in (0,1)$, and a proof string w of length $k \cdot \log N$, makes at most $\tilde{O}(\frac{N}{k} \cdot \varepsilon^{-2})$ oracle queries, and satisfies the following two conditions:*

1. *(Completeness:) if G is bipartite, then there exists a proof string $w \in \{0,1\}^{k \log N}$ such that $V^G(N,d,\varepsilon,w)=1$, with probability 1.*
2. *(Soundness:) if G is* rapidly-mixing *and ε-far from every bipartite graph, then for every proof string w, with probability at least $1/2$, it holds that $V^G(N,d,\varepsilon,w)=0$.*

Note that our tester has a one-sided error.

Proof. We define the parity of a (lazy) random walk as the parity of the number of actual (i.e., non-lazy) steps that take place in it. Loosely speaking, the proof that the graph G is bipartite is a subset $S \subseteq V$ of k vertices that are allegedly on the same side of G. To verify the proof, the verifier selects roughly $O(\log N)$ starting vertices, and takes approximately N/k random walks of length $O(\log N)$ from each starting vertex s. If there exist two random walks that start in s and end in S with different parities, then two corresponding vertices in S must be on different sides and the verifier rejects. Otherwise, the verifier accepts.

Since the graph is rapidly-mixing, the probability that a random walk that starts in s ends in S is roughly $|S|/N$. The key point (which is proved formally below) is that if the graph is far from bipartite, then for many starting vertices, the probability that the random walk ends in S *with parity* 0 (or equivalently, with parity 1) is $\Omega(|S|/N)$. That is, the probability of reaching S with either parity is significant enough. The protocol is presented in Figure 6.

Note that the proof and query complexities are as stated. We proceed to show that completeness and soundness hold.

Completeness. If $G=((L,R),E)$ is a bipartite graph such that $|L| \geq |R|$, and $S \subseteq L$ is the proof string, then there is no path between two vertices in S that has an odd length. Therefore, for every vertex $s \in V$, there are no two paths with different parities that end in S.

$\mathcal{MAP}$ for Bipartiteness of rapidly-mixing graphs (in the bounded degree graph model)

Input: oracle access to a graph $G = (V, E)$, the size $N \stackrel{\text{def}}{=} |V|$ of the graph, a bound d on the maximal degree in G, a proximity parameter $\varepsilon \in (0, 1)$, and a parameter $k \in [N]$.

The Proof:
Let $V = (L, R)$ such that L, R are disjoint independent sets and $|L| \geq |R|$ (such a partition is guaranteed if the graph is bipartite). The proof is an (arbitrary) subset $S \subseteq L$ of size k.

The Verifier:

1. Repeat $O\left(\frac{\log N}{\varepsilon}\right)$ times:
   (a) Select uniformly at random $s \in V$.
   (b) Take $O\left(\frac{N}{k} \cdot \frac{\log N}{\varepsilon}\right)$ (lazy) random walks starting at s, each of length $\ell \stackrel{\text{def}}{=} O(\log N)$.
   (c) Reject if there are two walks that end in S, having different parities.
2. If all of the previous tests passed, then accept.

Figure 6: $\mathcal{MAP}$ for Bipartiteness of rapidly-mixing graphs

Soundness. Suppose that $G = (V, E)$ is a rapidly-mixing graph of size $N = |V|$ that is ε-far from every bipartite graph and let $S \subseteq V$. For every $v \in V$ and $\sigma \in \{0, 1\}$, let p_v^σ be the probability that a (lazy) random walk of length $\ell = O(\log N)$ that starts at v, ends in S with parity σ. Since the graph is rapidly-mixing, $p_v^0 + p_v^1 \geq \frac{|S|}{2N}$ for every $v \in V$.

The following claim shows that, for an average vertex v, the probability that one random walk that starts at v ends in S with parity 0 *and* a second random walk that starts at v ends in S with parity 1, is roughly $\Omega((|S|/N)^2)$ (i.e., roughly the same as the probability for two random walks that start at v to end in S without any restriction on the parities of the walks).

Claim 7.1.1. $\sum_{v \in V} p_v^0 p_v^1 > \frac{\varepsilon |S|^2}{64 \ell N}$.

Proof. Suppose otherwise. Consider the following partition of the graph into (V_0, V_1) where $V_0 = \{v \in V : p_v^0 \geq p_v^1\}$ and $V_1 = \{v \in V : p_v^1 > p_v^0\}$. Let $E' = E(V_0, V_0) \cup E(V_1, V_1)$ be the set of all internal edges within V_0 and within V_1. We will obtain a contradiction by showing that G is ε-close to the bipartite graph $((V_0, V_1), E \backslash E')$ that is obtained from G by removing all edges in E'.

For every $v \in V$ and $\sigma \in \{0, 1\}$, let $A_{v,m}^\sigma$ denote the event that a (lazy) random walk of length m (where m is a parameter) that starts at v, ends in S with parity σ. In particular, $\Pr[A_{v,\ell}^\sigma] = p_v^\sigma$. Then, for every $\sigma \in \{0, 1\}$ and $v \in V_\sigma$, it holds that

$$p_v^{1-\sigma} \geq \sum_{u \in V_\sigma \text{ s.t. } (v,u) \in E'} \frac{1}{2d} \cdot \Pr[A_{u,\ell-1}^\sigma], \tag{7.1}$$

since a walk from v to S with parity $1 - \sigma$ can be obtained by a step to one of the neighbors of v in V_σ (which happens with probability $1/2d$ for each neighbor), and a walk of length $\ell - 1$ from this neighbor u to S with parity σ (i.e., the event $A_{u,\ell-1}^\sigma$).

Intuitively, since we expect the number of *lazy* steps in a lazy random walk to be rather large (at least $\ell/2$ in expectation), the probability that the event $A_{u,\ell-1}^\sigma$ occurs is closely related to the

probability that the event $A^{\sigma}_{u,\ell}$ occurs (indeed, we expect the discrepancy in the number of steps to be "hidden" by the (deviation of the number of) lazy steps). The foregoing intuition is formalized as follows. Note that with very high probability at least one lazy step occurs. Furthermore, observe that the probability that $A^{\sigma}_{u,\ell}$ occurs, conditioned on a specific step being lazy, is equal to the probability that $A^{\sigma}_{u,\ell-1}$ occurs. Indeed, by the union bound,

$$\begin{aligned}
p^{\sigma}_u &= \Pr[A^{\sigma}_{u,\ell}] \\
&\leq \Pr[A^{\sigma}_{u,\ell} \wedge \text{ no lazy steps in the walk}] + \sum_{i\in[\ell]} \Pr[A^{\sigma}_{u,\ell} \wedge \text{ the } i^{\text{th}} \text{ step in the walk is lazy}] \\
&\leq \Pr[\text{no lazy steps in the walk}] + \sum_{i\in[\ell]} \Pr[A^{\sigma}_{u,\ell} \mid \text{the } i^{\text{th}} \text{ step in the walk is lazy}]
\end{aligned}$$

We can bound the first term by $2^{-\ell}$, which by setting $\ell = \log(4N)$, is at most $1/(4N)$. As for the second term, the probability that a random walk of length ℓ from u ends in S with parity σ *conditioned on the i^{th} step being lazy* is equal to the probability that a random walk of length $\ell-1$ from u ends in S with parity σ. Hence

$$p^{\sigma}_u \leq \frac{1}{4N} + \ell \cdot \Pr[A^{\sigma}_{u,\ell-1}] \tag{7.2}$$

Using Eq. (7.1) and Eq. (7.2), we obtain that:

$$\begin{aligned}
\sum_{v\in V} p^0_v p^1_v &= \sum_{\sigma\in\{0,1\}} \sum_{v\in V_\sigma} p^{\sigma}_v p^{1-\sigma}_v \\
&\geq \sum_{\sigma\in\{0,1\}} \sum_{\substack{(v,u)\in E' \\ \text{s.t. } v,u\in V_\sigma}} p^{\sigma}_v \cdot \frac{\Pr[A^{\sigma}_{u,\ell-1}]}{2d} \\
&\geq \sum_{\sigma\in\{0,1\}} \sum_{\substack{(v,u)\in E' \\ \text{s.t. } v,u\in V_\sigma}} p^{\sigma}_v \cdot \frac{1}{2\ell d} \cdot \left(p^{\sigma}_u - \frac{1}{4N}\right) \\
&\geq |E'| \cdot \frac{1}{2\ell d} \cdot \frac{|S|}{4N} \cdot \frac{|S|}{8N}
\end{aligned}$$

where the last inequality follows from the fact that for every $w \in V_\sigma$ it holds that $p^{\sigma}_w \geq (p^{\sigma}_w + p^{1-\sigma}_w)/2 \geq |S|/4N$.

Hence, by our hypothesis, $|E'| \leq \frac{\varepsilon|S|^2}{64\ell N} \cdot \left(\frac{1}{2\ell d} \cdot \frac{|S|}{4N} \cdot \frac{|S|}{8N}\right)^{-1} = \varepsilon d N$. Therefore, by removing an ε fraction of the edges of G we obtain a bipartite graph, in contradiction to our assumption that G is ε-far from bipartite. This concludes the proof of Claim 7.1.1. □

We say that a vertex v is good if $p^0_v p^1_v \geq \frac{\varepsilon|S|^2}{128\ell N^2}$. (Intuitively, a vertex v is good if two random walks that start at v are likely to end in S with different parities.) Let $\alpha \in [0,1]$ be the fraction of good vertices in V. By Claim 7.1.1,

$$\frac{\varepsilon|S|^2}{64\ell N} < \sum_{v\in V} p^0_v p^1_v = \sum_{v \text{ is good}} p^0_v p^1_v + \sum_{v \text{ is not good}} p^0_v p^1_v \leq \alpha N \cdot \left(\frac{2|S|}{N}\right)^2 + N \cdot \frac{\varepsilon|S|^2}{128\ell N^2},$$

where the last inequality uses the fact that for every vertex $v \in V$ it holds that $p_v^0 \cdot p_v^1 \leq (p_v^0 + p_v^1)^2 \leq (2|S|/N)^2$. Hence, the fraction of good vertices is at least $\alpha = \Omega(\varepsilon/\log N)$.

Hence, with probability at least 0.9, at least one of the starting vertices s (which were selected in one of the $O(\log N/\varepsilon)$ iterations) is good. Assume that indeed, in one of the iterations a good vertex s is selected. Hence, $p_s^0 p_s^1 \geq \frac{\varepsilon|S|^2}{128\ell N^2}$ and $p_s^0 + p_s^1 \leq \frac{2|S|}{N}$, which implies that $p_s^0, p_s^1 = \Omega\left(\frac{|S|\varepsilon}{N \log N}\right)$. Therefore, since we take $O\left(\frac{N}{|S|} \cdot \frac{\log N}{\varepsilon}\right)$ random walks starting in s, with probability 0.9, there will be at least one walk thats ends in S with parity 0 *and* one walk that ends in S with parity 1. Hence, by the union bound, the tester rejects with probability at least 1/2. □

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. Lastly, we thank the anonymous referees for valueable comments.

8 References

[AFNS09] Noga Alon, Eldar Fischer, Ilan Newman, and Asaf Shapira. A combinatorial characterization of the testable graph properties: it's all about regularity. *SIAM Journal on Computing*, 39(1):143–167, 2009.

[AKNS00] Noga Alon, Michael Krivelevich, Ilan Newman, and Mario Szegedy. Regular languages are testable with a constant number of queries. *SIAM J. Comput.*, 30(6):1842–1862, 2000.

[AS03] Sanjeev Arora and Madhu Sudan. Improved low-degree testing and its applications. *Combinatorica*, 23(3):365–426, 2003.

[AW09] Scott Aaronson and Avi Wigderson. Algebrization: A new barrier in complexity theory. *ACM Trans. Comput. Theory*, 1:2:1–2:54, February 2009.

[Bab85] László Babai. Trading group theory for randomness. In *Proceedings of the seventeenth annual ACM symposium on Theory of computing*, pages 421–429. ACM, 1985.

[BBM11] Eric Blais, Joshua Brody, and Kevin Matulef. Property testing lower bounds via communication complexity. In *IEEE Conference on Computational Complexity*, pages 210–220, 2011.

[BdW02] Harry Buhrman and Ronald de Wolf. Complexity measures and decision tree complexity: a survey. *Theor. Comput. Sci.*, 288(1):21–43, 2002.

[BFLS91] László Babai, Lance Fortnow, Leonid A Levin, and Mario Szegedy. Checking computations in polylogarithmic time. In *Proceedings of the twenty-third annual ACM symposium on Theory of computing*, pages 21–32. ACM, 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.

[BGH+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.

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

[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 *Proceedings of the thirty-third annual ACM symposium on Theory of computing*, pages 266–275. ACM, 2001.

[CCGT14] Amit Chakrabarti, Graham Cormode, Navin Goyal, and Justin Thaler. Annotations for sparse data streams. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 687–706. SIAM, 2014.

[CCM+15] Amit Chakrabarti, Graham Cormode, Andrew McGregor, Justin Thaler, and Suresh Venkatasubramanian. Verifiable stream computation and arthur–merlin communication. In *30th Conference on Computational Complexity (CCC 2015)*, volume 33, pages 217–243. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2015.

[CCMT14] Amit Chakrabarti, Graham Cormode, Andrew McGregor, and Justin Thaler. Annotations in data streams. *ACM Transactions on Algorithms (TALG)*, 11(1):7, 2014.

[CEG95] Ran Canetti, Guy Even, and Oded Goldreich. Lower bounds for sampling algorithms for estimating the average. *Inf. Process. Lett.*, 53(1):17–25, 1995.

[CGR+12] Artur Czumaj, Oded Goldreich, Dana Ron, C Seshadhri, Asaf Shapira, and Christian Sohler. Finding cycles and trees in sublinear time. *Random Structures & Algorithms*, 2012.

[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.

[DTV15] Samira Daruki, Justin Thaler, and Suresh Venkatasubramanian. Streaming verification in data analysis. *arXiv preprint arXiv:1509.05514*, 2015.

[EKR04] Funda Ergün, Ravi Kumar, and Ronitt Rubinfeld. Fast approximate probabilistically checkable proofs. *Inf. Comput.*, 189(2):135–159, 2004.

[FGL14] Eldar Fischer, Yonatan Goldhirsh, and Oded Lachish. Partial tests, universal tests and decomposability. In *Innovations in Theoretical Computer Science, ITCS'14, Princeton, NJ, USA, January 12-14, 2014*, pages 483–500, 2014.

[FLM+12] Eldar Fischer, Oded Lachish, Arie Matsliah, Ilan Newman, and Orly Yahalom. On the query complexity of testing orientations for being eulerian. *ACM Transactions on Algorithms*, 8(2):15, 2012.

[GGK14] Oded Goldreich, Tom Gur, and Ilan Komargodski. Strong locally testable codes with relaxed local decoders. *Electronic Colloquium on Computational Complexity (ECCC)*, 21:25, 2014.

[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.

[GH98] Oded Goldreich and Johan Håstad. On the complexity of interactive proofs with bounded communication. *Information Processing Letters*, 67(4):205–214, 1998.

[GK92] Oded Goldreich and Hugo Krawczyk. Sparse pseudorandom distributions. *Random Struct. Algorithms*, 3(2):163–174, 1992.

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

[Gol08] Oded Goldreich. *Computational complexity - a conceptual perspective*. Cambridge University Press, 2008.

[Gol10a] Oded Goldreich, editor. *Property Testing - Current Research and Surveys*, volume 6390 of *Lecture Notes in Computer Science*. Springer, 2010.

[Gol10b] Oded Goldreich. Short locally testable codes and proofs: A survey in two parts. In *Property Testing* [Gol10a], pages 65–104.

[Gol11] Oded Goldreich. Introduction to testing graph properties. In *Studies in Complexity and Cryptography. Miscellanea on the Interplay between Randomness and Computation*, pages 470–506. Springer, 2011.

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

[GR99] Oded Goldreich and Dana Ron. A sublinear bipartiteness tester for bounded degree graphs. *Combinatorica*, 19(3):335–373, 1999.

[GR02] Oded Goldreich and Dana Ron. Property testing in bounded degree graphs. *Algorithmica*, 32(2):302–343, 2002.

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

[GR13a] Oded Goldreich and Dana Ron. On sample-based testers. *Electronic Colloquium on Computational Complexity (ECCC)*, 20:109, 2013.

[GR13b] Tom Gur and Ran Raz. Arthur-Merlin streaming complexity. In *Proceedings of the 40th International Colloquium on Automata, Languages and Programming (ICALP)*, 2013.

[GR13c] Tom Gur and Ron Rothblum. Non-interactive proofs of proximity. *Electronic Colloquium on Computational Complexity (ECCC)*, 20:78, 2013.

[GR14] Oded Goldreich and Dana Ron. On learning and testing dynamic environments. *Electronic Colloquium on Computational Complexity (ECCC)*, 21:29, 2014.

[GS92] Peter Gemmell and Madhu Sudan. Highly resilient correctors for polynomials. *Inf. Process. Lett.*, 43(4):169–174, 1992.

[GS06] Oded Goldreich and Madhu Sudan. Locally testable codes and PCPs of almost-linear length. *J. ACM*, 53(4):558–655, 2006.

[GS10a] Dmitry Gavinsky and Alexander A Sherstov. A separation of NP and coNP in multiparty communication complexity. *arXiv preprint arXiv:1004.0817*, 2010.

[GS10b] Oded Goldreich and Or Sheffet. On the randomness complexity of property testing. *Computational Complexity*, 19(1):99–133, 2010.

[GS12] Oded Goldreich and Igor Shinkar. Two-sided error proximity oblivious testing - (extended abstract). In *APPROX-RANDOM*, pages 565–578, 2012.

[GS13] Lior Gishboliner and Asaf Shapira. Deterministic vs non-deterministic graph property testing. *Electronic Colloquium on Computational Complexity (ECCC)*, 20:59, 2013.

[GVW02] Oded Goldreich, Salil Vadhan, and Avi Wigderson. On interactive proofs with a laconic prover. *Computational Complexity*, 11(1-2):1–53, 2002.

[HLNT05] Shirley Halevy, Oded Lachish, Ilan Newman, and Dekel Tsur. Testing orientation properties. *Electronic Colloquium on Computational Complexity (ECCC)*, 2005.

[HLNT07] Shirley Halevy, Oded Lachish, Ilan Newman, and Dekel Tsur. Testing properties of constraint-graphs. In *IEEE Conference on Computational Complexity*, pages 264–277, 2007.

[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.

[Lev87] Leonid A Levin. One way functions and pseudorandom generators. *Combinatorica*, 7(4):357–363, 1987.

[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.

[New10] Ilan Newman. Property testing of massively parametrized problems–a survey. In *Property testing: current research and surveys*, pages 142–157. Springer, 2010.

[PY96] Christos H Papadimitriou and Mihalis Yannakakis. On limited nondeterminism and the complexity of the vc dimension. *Journal of Computer and System Sciences*, 53(2):161–170, 1996.

[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.

[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.

[Tha14] Justin Thaler. Semi-streaming algorithms for annotated graph streams. *arXiv preprint arXiv:1407.3462*, 2014.

A Background

A.1 Communication Complexity

Let X and Y be finite sets, and let $f : X \times Y \to \{0,1\}$ be a function. In the *two-party probabilistic communication complexity model* we have two computationally unbounded players, traditionally referred to as Alice and Bob. Both players share a random string. Alice gets as an input $x \in X$. Bob gets as an input $y \in Y$. At the beginning, neither one of the players has any information regarding the input of the other player. Their common goal is to compute the value of $f(x,y)$, while minimizing the communication between them. In each step of the protocol, one of the players sends one bit to the other player. This bit may depend on the player's input, the common random string, as well as on all previous bits communicated between the two players. At the end of the protocol, both players output $f(x,y)$ with high probability.

We say that a given protocol π computes a (possibly partial) function $f : X \times Y \to \{0,1\}$ if for every $x \in X$ and $y \in Y$ with probability at least $2/3$ Alice outputs $f(x,y)$ after interacting with Bob.[22] 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 (where the maximum is taken over all possible coin tosses). The communication complexity of a function f is defined as:

$$\mathsf{CC}(f) = \min_{\pi \text{ that compute } f} \mathsf{CC}(\pi).$$

For a family of functions $\mathcal{F} = \{f_n : X_n \to Y_n\}_{n\in\mathbb{N}}$ we define the communication complexity of $\mathcal{F}$ as $\mathsf{CC}_n(\mathcal{F}) = \mathsf{CC}(f_n)$.

Set-Disjointness. The (unique) *set-disjointness* problem is the classical communication complexity problem wherein Alice gets an n-bit string x, Bob gets an n-bit string y, and their goal is to decide whether there exists a unique $i \in [n]$ such that $x_i = y_i = 1$. Formally,

Definition A.1. *For every $n \in \mathbb{N}$, $\mathsf{DISJ}_n : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ is the communication complexity problem 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.

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

A.2 $\mathcal{MA}$ 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.2. *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*

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

   (where the probability is over the common random string).

2. Soundness*: for all $(x, y) \in f^{-1}(0)$ and for any string $w \in \{0,1\}^p$ we have*

$$\Pr\left[\pi\big((x, y), w\big) = 1\right] \leq 1/3$$

   (where the probability is over the common random string).

The $\mathcal{MA}$ 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.3. *Every $\mathcal{MA}$ communication complexity protocol for DISJ_n with proof complexity p and communication complexity c satisfies $p \cdot c = \Omega(n)$.*

A.3 Error Correcting Codes

We first introduce codes as objects of fixed length and then give asymptotic variants of the definitions. Let Σ be a finite alphabet. An error-correcting code (over Σ) is an injective function $C : \Sigma^k \to \Sigma^n$ where $k, n \in \mathbb{N}$ and $k < n$. Every element in the range of C is called a codeword. The length 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 $w \in \Sigma^n$ that is δ-close to a codeword $w' = C(m)$ for some $m \in \Sigma^k$, 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 length and constant query complexity (i.e., t does not depend on n), obtaining an LDC with constant query complexity and polynomial length is a major open problem in coding theory.

We will also consider a relaxation of LDCs, introduced by Ben-Sasson *et al.* [BGH+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 [BGH+06]) of relaxed-LDCs with constant query complexity and almost linear length.

Definition A.4 (relaxed-LDC, adapted from [BGH+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 $m \in \Sigma^k$, 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 [BGH+06] in two ways. The first difference is that [BGH+06] require an additional, third, condition that we do not need. (However, [BGH+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 [BGH+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 [BGH+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.4 Multivariate Polynomials and Low Degree Testing

In this section we recall some important facts on multivariate polynomials (see [Sud95] for a far more detailed introduction). In the following we fix a finite field $\mathbb{F}$ and a dimension m and consider m-variate polynomials over $\mathbb{F}$.

Lemma A.5 (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 be selected uniformly at random in S^m. Then,*

$$\Pr_{r \in_R S}[P(r) = 0] \leq \frac{d}{|S|}.$$

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.6 (Self-Correction Procedure (cf. [GS92, Sud95])**.** *Let $\delta < 1/3$ and $d, m \in \mathbb{N}$ such that $d \leq |\mathbb{F}|$. 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 d, then given $x \in \mathbb{F}^m$, the algorithm outputs $P(x)$ with probability 1.*

In Theorem A.6, 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.7 (Total Degree Test (a.k.a. Low Degree Test) (see [RS96, Sud95, AS03])**.** *Let $\varepsilon \in (0, 1/2)$ and $d, m \in \mathbb{N}$ such that $d \leq |\mathbb{F}|/2$. There exists an algorithm that, given oracle access to an m-variate function $P : \mathbb{F}^m \to \mathbb{F}$, makes $O(d \cdot \mathsf{poly}(1/\varepsilon))$ queries and:*

1. *Accepts every function that is a polynomial of* total *degree d with probability 1; and*
2. *Rejects functions that are ε-far from every polynomial of total degree d 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.8 (Individual Degree Test)**.** *Let $d, m \in \mathbb{N}$ such that $dm < |\mathbb{F}|/10$ and $\varepsilon \in (0, 1/10)$. 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 $P : \mathbb{F}^m \to \mathbb{F}$, the tester T first runs the total degree test of Theorem A.7 on P with respect to proximity ε and total degree dm. If the total degree verifier rejects, then T rejects. Otherwise, for every axis $i \in [m]$, the tester T chooses at random $r_1, \ldots, r_{i-1}, r_{i+1}, \ldots, r_m \in_R \mathbb{F}$ and run 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 respect to proximity 0.5 and with soundness error 0.1 (e.g., by selecting at random $O(d)$ points and checking, via interpolation, that they lie on the same degree d polynomial). The tester T accepts if all tests pass, and otherwise it rejects.

Completeness. Completeness follows 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 $1/2$ and we are done. Thus, we focus on the case that P is ε-close to a total degree dm polynomial P'.

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 $d' \in [d+1, dm]$ 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 (Lemma A.5), it can vanish on only a $\frac{dm}{|\mathbb{F}|}$ fraction of its domain. Thus,

$$\Pr_{r_1,\ldots,r_{i-1},r_{i+1},\ldots,r_m \in \mathbb{F}} \left[P'_{d'}(r_1, \ldots, r_{i-1}, r_{i+1}, \ldots, r_m) = 0\right] \leq \frac{dm}{|\mathbb{F}|}. \tag{A.1}$$

On the other hand, since P and P' are ε-close, by Markov's inequality:

$$\Pr_{r_1,\ldots,r_{i-1},r_{i+1},\ldots,r_m \in_R \mathbb{F}} \left[Q_i \text{ is } 4\varepsilon\text{-far from } Q'_i\right] \leq \frac{1}{4} \tag{A.2}$$

where $Q_i(z) \stackrel{\text{def}}{=} P(r_1, \ldots, r_{i-1}, z, r_{i+1}, \ldots, r_m)$ and $Q'_i(z) \stackrel{\text{def}}{=} P'(r_1, \ldots, r_{i-1}, z, r_{i+1}, \ldots, r_m)$. By Eqs. (A.1) and (A.2) and a union bound, with probability at least $0.75 - \frac{dm}{|\mathbb{F}|} > 0.6$ over $r_1, \ldots, r_{i-1}, r_{i+1}, \ldots, r_m \in \mathbb{F}$ both (1) $P'_{d'}(r_1, \ldots, r_{i-1}, r_{i+1}, \ldots, r_m) \neq 0$, and (2) Q_i is 4ε-close to Q'_i. In the following we fix $r_1, \ldots, r_{i-1}, r_{i+1}, \ldots, r_m$ that satisfy the two foregoing conditions.

Since $P'_{d'}(r_1, \ldots, r_{i-1}, r_{i+1}, \ldots, r_m) \neq 0$, the polynomial Q'_i has degree $d' > d$. Suppose that Q_i is ε'-close to some degree d polynomial, for some $\varepsilon' \in [0, 1]$. Then, by the triangle inequality Q'_i is $4\varepsilon + \varepsilon'$ close to the same polynomial (which is different from Q'_i since Q'_i has degree $d' > d$). Two distinct degree $\leq d'$ univariate polynomials have relative distance at least $1 - d'/|\mathbb{F}|$ and so $\varepsilon' \geq 1 - \frac{d'}{|\mathbb{F}|} - 4\varepsilon \geq 1 - \frac{dm}{|\mathbb{F}|} - 4\varepsilon \geq 0.5$, or in other words, Q_i is 0.5-far from every degree d polynomial. The univariate degree d test w.r.t. proximity 0.5 (and soundness error 0.1) detects this with probability 0.9. Hence, overall the tester rejects with probability at least $0.6 \cdot 0.9 > 0.5$. □

A.5 The Sum-Check Protocol

In this appendix we provide some background on the sum-check protocol that was first introduced by Lund *et al.* [LFKN92]. Recall that the sum-check protocol is an interactive proof for a statement of the form

$$\sum_{x_1,\ldots,x_m \in H} P(x_1, \ldots, x_m) = 0.$$

where P is a (relatively) low-degree polynomial over a 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 .[23] 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.7) and self-correction (Theorem A.6).

B Proofs and Adaptations of Known Results

In this section we provide proofs and adaptations of known results, which are included here for completeness.

B.1 Proofs of Standard Claims from Section 5

In this section we provide the missing proofs of the standard claims used in Section 5.

Proposition 5.3 (folklore)**.** *Every property $\Pi = \cup_{n \in \mathbb{N}} \Pi_n$ (where $\Pi_n \subseteq \{0,1\}^n$) can be tested by making $O(\log |\Pi_n|/\varepsilon)$ queries (without a proof).*

Proof. We show that every property $\Pi = \cup_{n \in \mathbb{N}} \Pi_n$ (where $\Pi_n \subseteq \{0,1\}^n$) can be tested by making $O(\log |\Pi_n|/\varepsilon)$ queries. Recall that the lemma can be proved via learning theory techniques, but we provide an alternative proof that makes use of the notion of $\mathcal{MAP}$s.

Consider an $\mathcal{MAP}$ for Π in which the proof, of length $\log_2 |\Pi_n|$, is an explicit and concise description of the object $x \in \Pi_n$ (e.g., its index with respect to the lexicographical ordering of the strings in Π_n). The verifier can verify the proof by querying the object x at $O(1/\varepsilon)$ locations uniformly at random (and compare the answers to the string reconstructed based on the proof). The lemma follows by noting that this $\mathcal{MAP}$ makes *proof-oblivious queries* and applying Theorem 4.2, which guarantees that if Π has an $\mathcal{MAP}$ verifier that makes q proof oblivious queries and uses a proof of length p, then Π has a tester that makes $O(p \cdot q)$ queries without using a proof. □

Proposition 5.4 (folklore)**.** *For every constant $\varepsilon \in (0, 1/4]$ and set $S \subseteq \{0,1\}^n$, it holds that $\Pr_{x \in_R \{0,1\}^n}[x \text{ is } \varepsilon\text{-close to } S] \leq |S| \cdot 2^{-n/8}$.*

[23] More precisely, a variant of the $(m-1)$-dimensional TensorSum problem in which 0 is replaced with an arbitrary field element.

Proof. We show that for every constant $\varepsilon \in (0, 1/4]$ and set $S \subseteq \{0,1\}^n$ it holds that $\Pr_{x \in_R \{0,1\}^n}[x \text{ is } \varepsilon\text{-close to } S] \leq |S| \cdot 2^{-n/8}$. Observe that

$$\begin{aligned}\Pr_{x \in_R \{0,1\}^n}[\exists s \in S \text{ such that } x \text{ is } \varepsilon\text{-close to } s] &\leq \sum_{s \in S} \Pr_{x \in_R \{0,1\}^n}[x \text{ is } \varepsilon\text{-close to } s] \\ &= |S| \cdot \Pr_{x \in_R \{0,1\}^n}[x \text{ has at most } \varepsilon n \text{ 1's}] \\ &\leq |S| \cdot \exp(-2 \cdot (1/4)^2 \cdot n).\end{aligned}$$

where the first inequality follows from the union bound, and the last inequality follows from the Chernoff bound and the fact that $\varepsilon < 1/4$. □

Proposition 5.6 (implicit in [GK92], see also [Gol08, Exercise 8.1])**.** *Let $\mathcal{F}$ be a class of functions from $\{0,1\}^n$ to $\{0,1\}$, of size at most $2^{2^{n/4}}$. Then, 99% of subsets of $\{0,1\}^n$ of size $s = O(\log |\mathcal{F}|)$ are PRGs that fool $\mathcal{F}$.*

Proof. Let $\mathcal{F}$ be a class of functions of size at most $2^{2^{n/4}}$. We show that 99% of sets of size $O(\log |\mathcal{F}|)$ are PRGs that fool $\mathcal{F}$.

For every set $S \subseteq \{0,1\}^n$ and function $f \in \mathcal{F}$, let $\delta_f(S) = |\Pr_{x \in_R S}[f(x) = 1] - \mu_f|$ where $\mu_f \stackrel{\text{def}}{=} \Pr_{x \in_R \{0,1\}^n}[f(x) = 1]$. Let $s \in [2^{n/4}]$ be an integer and let S be a random set of size s. Then, for every $f \in \mathcal{F}$ it holds that

$$\Pr_S[\delta_f(S) \geq 1/10] = \Pr_S\left[\left|\Pr_{x \in_R S}[f(x) = 1] - \mu_f\right| \geq 1/10\right] \leq 2^{-\Omega(t)},$$

where the last inequality follows from the Chernoff bound.[24] Thus, by the union bound, the probability that for *every* $f \in \mathcal{F}$ it holds that $\delta_f(S) < 1/10$, is at least $|\mathcal{F}| \cdot 2^{-\Omega(s)}$ (where the probability is over the choice of S). The lemma follows by setting $s = \Theta(\log |\mathbb{F}|)$. □

B.2 Precision Sampling

Proof of Claim 6.2.1. We show that there exists $j \in [\lceil \log_2 2/\varepsilon \rceil]$ such that a $\frac{2^j \varepsilon}{4 \cdot \lceil \log_2(2/\varepsilon) \rceil}$ fraction of $x_1, \ldots, x_k$ are 2^{-j}-far from their corresponding sub-properties $\Pi^{\alpha_1}_{n/k}, \ldots, \Pi^{\alpha_k}_{n/k}$.

Let $d \stackrel{\text{def}}{=} \lceil \log_2(2/\varepsilon) \rceil$. Let $\Delta_{\mathsf{REL}}(z, W)$ be defined as the minimal *relative* Hamming distance of z from the set W. For every $j \in [d]$, let

$$S_j \stackrel{\text{def}}{=} \left\{ i \in [k] : \Delta_{\mathsf{REL}}\left(x_i, \Pi^{\alpha_i}_{n/k}\right) \in \left(2^{-j}, 2^{-(j-1)}\right] \right\},$$

and let $T = [k] \setminus (\cup_{i \in [d]} S_j)$. Notice that the sets $T, S_1, S_2, \ldots, S_d$ form a partition of the k inputs. Also note that, by our setting of d, for every $i \in T$, it holds that x_i is $\varepsilon/2$-close to $\Pi^{\alpha_i}_{n/k}$.

[24] We note that since the set S is chosen *without repetitions* one cannot directly apply the Chernoff bound. Still, since $s \leq 2^{n/4}$ the probability for a repetition is at most $s^2/2^n \leq 2^{-\Omega(n)}$. Conditioning on an event (i.e., that there are no repetitions) that occurs with probability $1 - \delta$ can increase the probability by at most a $1/(1-\delta)$ factor.

Suppose towards a contradiction that for every $j \in [d]$ it holds that $|S_j| < \frac{2^j \varepsilon}{4d} \cdot k$. Using the fact that for every $i \in S_j$ it holds that x_i is $2^{-(j-1)}$-close to Π^{α_i}, we get

$$\begin{aligned}
\Delta_{\mathsf{REL}}\left(x, \Pi^{\alpha_1}_{n/k} \times \ldots \times \Pi^{\alpha_k}_{n/k}\right) &\leq \frac{1}{k}\sum_{i=1}^{k} \Delta_{\mathsf{REL}}\left(x_i, \Pi^{\alpha_i}\right) \\
&= \frac{1}{k}\sum_{i \in T} \Delta_{\mathsf{REL}}\left(x_i, \Pi^{\alpha_i}\right) + \frac{1}{k}\sum_{j \in [d]}\sum_{i \in S_j} \Delta_{\mathsf{REL}}\left(x_i, \Pi^{\alpha_i}_{n/k}\right) \\
&\leq \frac{|T|}{k} \cdot \frac{\varepsilon}{2} + \frac{1}{k}\sum_{j \in [d]} 2^{-(j-1)} \cdot |S_j| \\
&< \frac{\varepsilon}{2} + \sum_{j \in [d]} \frac{\varepsilon}{2d} \\
&= \varepsilon,
\end{aligned}$$

contradicting our assumption that x is ε-far from $\Pi^{\bar{A}}$. □