WARWICK
THE UNIVERSITY OF WARWICK

A Thesis Submitted for the Degree of PhD at the University of Warwick

Permanent WRAP URL:

http://wrap.warwick.ac.uk/182844

Copyright and reuse:
This thesis is made available online and is protected by original copyright.
Please scroll down to view the document itself.
Please refer to the repository record for this item for information to help you to cite it.
Our policy information is available from the repository home page.

For more information, please contact the WRAP Team at: wrap@warwick.ac.uk

warwick.ac.uk/lib-publications

Classical and Quantum Sublinear Algorithms

by

Marcel de Sena Dall'Agnol

Thesis submitted to the University of Warwick

in partial fulfilment of the requirements

for admission to the degree of

Doctor of Philosophy

Department of Computer Science

June 2023

Contents

List of Figures

Acknowledgments

I would like to thank Tom, first and foremost, for the amazing supervision I was fortunate to receive. You taught me how to be independent and was available when I needed it most – all with patience and humility. I always felt like an equal working with you, and will work hard to pay it forward.

I couldn't have asked for better company than Aditya, Grzegorz, Rado, Nathan, Thejaswini, Iman, Tatiana, Ninad, Bruno, Sathya, Hugo, Jack and so many others I had the pleasure of sharing these years at Warwick with. Thank you for being here throughout the ups and downs, but mostly for being the wonderful and inspiring people you are.

To my collaborators Oded, Subhayan, Justin, Nick, Graham and Chris (and Tom, of course); and to my examiners Igor and Henry, for kindly offering your knowledge and expertise to make this the best work it can be.

Special thanks go to Igor along with everyone else who played a role in my coming to Warwick in the first place: Maria, Daniel, Roberto and Eduardo in particular, who saw potential in me when I didn't; who showed me the thrill and wonder of mathematics; who validated my highest ambitions and guided me from the moment I stepped into IMPA until I left for the UK.

To the unbelievably talented people at IMPA whom I shared the two most challenging years of my life with: Mitul, Santiago, Victor, Bruno, Marcelo, Jonathan, Maurício and everyone else. When things are hard, it always helps to remind myself of that time and the new heights you're all reaching.

Thamiris, meu amor! Não é exagero dizer que, sem você, dificilmente teria conseguido chegar aqui. Não foi só coragem; só espero um dia poder retribuir à altura.

Mãe e pai, muito obrigado por tudo. Não fosse por vocês, nada disso estaria sequer no horizonte. O agradecimento se estende para a família toda, mas em especial a você, vô. Por muito pouco não pude compartilhar contigo a alegria de começar o doutorado; acho que você estaria orgulhoso!

Por último, a todas as pessoas incríveis com quem compartilhei meus anos no Âncora. Nada antes ou depois fez tanto sentido. Sempre que fico com medo de me perder, basta lembrar de vocês: qualquer vida guiada por valores assim é uma vida bem vivida.

Declaration

This thesis is submitted to the University of Warwick in support of my application for the degree of Doctor of Philosophy. It has been composed by myself and has not been submitted in any previous application for any degree.

Much of the work present in this thesis has been published in peer-reviewed conferences and journals, all with substantial work carried out by the author:

- **A Structural Theorem for Local Algorithms with Applications to Coding, Testing, and Privacy** [DGL21] (Chapter 4)
  Marcel Dall'Agnol, Tom Gur and Oded Lachish
  Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms, pp. 1651–1665. DOI: 10.5555/3458064.3458164.
- **Quantum Proofs of Proximity** [DGRT22] (Chapter 5)
  Marcel Dall'Agnol, Tom Gur, Subhayan Roy Moulik, and Justin Thaler
  Quantum 6 (Oct. 2022), p. 834. DOI: 10.22331/q-2022-10-13-834.
  Also presented at the 16th Conference on the Theory of Quantum Computation, Communication, and Cryptography (TQC 2021).
- **Streaming Zero-Knowledge Proofs** [CDGH23] (Chapter 6)[1]
  Graham Cormode, Marcel Dall'Agnol, Tom Gur, and Chris Hickey
  In submission. DOI: 10.48550/arxiv.2301.02161.

The author also contributed to the following publication during his doctoral study.

- **On the Necessity of Collapsing for Post-Quantum and Quantum Commitments** [DS23]
  Marcel Dall'Agnol and Nicholas Spooner
  18th Conference on the Theory of Quantum Computation, Communication and Cryptography (TQC 2023), to appear. DOI: 10.4230/LIPIcs.TQC.2023.2

[1]We note that Chapter 6 contains original follow-up work on the same topic as a coauthor's Ph.D. thesis [Hic21, Chapter 5].

Abstract

This thesis investigates the capabilities of classical and quantum sublinear algorithms through the lens of complexity theory. The formal classification of problems between "tractable" (by constructing efficient algorithms that solve them) and "intractable" (by proving no efficient algorithm can) is among the most fruitful lines of work in theoretical computer science, which includes, amongst an abundance of fundamental results and open problems, the notorious P vs. NP question.

This particular incarnation of the decision-versus-verification question stems from a choice of computational *model*: polynomial-time Turing machines. It is far from the only model worthy of investigation, however; indeed, measuring time up to polynomial factors is often too "coarse" for practical applications. We focus on *quantum computation*, a more complete model of physically realisable computation where quantum mechanical phenomena (such as interference and entanglement) may be used as computational resources; and *sublinear algorithms*, a formalisation of ultra-fast computation where merely reading or storing the entire input is impractical, e.g., when processing massive datasets such as social networks or large databases.

We begin our investigation by studying structural properties of *local algorithms*, a large class of sublinear algorithms that includes property testers and is characterised by the inability to even *see* most of the input. We prove that, in this setting, queries – the main complexity measure – can be replaced with random samples. Applying this transformation yields, among other results, the state-of-the-art query lower bound for relaxed local decoders.

Focusing our attention onto *property testers*, we begin to chart the complexity-theoretic landscape arising from the classical vs. quantum and decision vs. verification questions in testing. We show that quantum hardware and communication with a powerful but untrusted prover are "orthogonal" resources, so that one cannot be substituted for the other. This implies all of the possible separations among the analogues of QMA, MA and BQP in the property-testing setting.

We conclude with a study of *zero-knowledge* for (classical) streaming algorithms, which receive one-pass access to the entirety of their input but only have sublinear *space*. Inspired by cryptographic tools, we construct commitment protocols that are *unconditionally secure* in the streaming model and can be leveraged to obtain zero-knowledge streaming interactive proofs – and, in particular, show that zero-knowledge is achievable in this model.

Chapter 1

Introduction

This introduction begins with a broad overview of our topics of investigation: *sublinear algorithms* and *quantum computation*. In the remainder (Sections 1.1 to 1.3) we discus the mathematical landscape where they stand, key results in the literature and our main contributions; formal definitions and proofs are then covered in Chapters 4 to 6.

An algorithm is *sublinear* if it does not have sufficient resources to access and/or store the entirety of its input,[1] but still performs some nontrivial computational task under these constraints. Such limitations can arise in different scenarios, which, in turn, characterise different models: if an algorithm can only probe its input at a few positions, we call it *local*; settings where data is distributed across devices that are distant from one another, or where it is processed in a long stream of updates, correspond to other (sublinear) models.

Clearly, not every problem is solvable locally. The simplest of examples is computing the parity of a string $x \in \{0,1\}^n$: since flipping a single bit flips the result, it is not hard to see that this task requires inspecting all n coordinates. (Interestingly, this is false for certain quantum algorithms; see Section 5.5.3.1.) Nonetheless, there is a rich collection of problems solvable by an algorithm under severe constraints, such as:

- *property testers* [RS96, GGR98], probabilistic algorithms that solve approximate decision problems by only probing a minuscule portion of their input;
- *locally decodable codes* (LDCs) [KT00] and *locally testable codes (LTCs)* [GS06], error-correcting codes which admit, using a small number of queries to their

[1]While sublinear-*time* is often used as a synonym for sublinear, this terminology is often inaccurate. Efficiency in query complexity or space does not always imply time-efficient algorithms; in this thesis, we do not study time complexity unless explicity stated.

input, algorithms able to decode individual symbols and test the validity of the encoding, respectively;

- *proof systems* that allow for efficient verification of a computation that is too costly to perform locally. Notable among these are *probabilistically checkable proofs* (PCPs) [FGL$^+$91, AS98, ALM$^+$98], which are encodings of NP proofs that can be verified by examining only (say) 3 of its bits; and
- *streaming algorithms* [MP80, AMS99, BBD$^+$02, Mut05], whose fundamental constraint is *space*. In the streaming model an algorithm receives complete but *one-pass* access to an n-bit string, and must compute some feature of it with sublinear space. Interest in this type of algorithm is motivated by applications to massive datasets and online algorithms (see, e.g., the book [Gam10]).

Quantum computation. Quantum mechanics has upended not only our understanding of physics, but has deep and far-reaching implications on the nature of computation. Discussions on them draw as far back as Feynman, who noted quantum computers may be able to simulate quantum physics exponentially more efficiently than classical computers [Fey82] and also considered their limits [Fey86].

Loosely speaking, while a classical bit (of a randomised algorithm) is described by a *distribution* over the "states" 0 and 1, a quantum bit (qubit) is given by a *superposition* of the quantum states $|0\rangle$ and $|1\rangle$. A quantum computer manipulates a collection of (say) n qubits and then performs a *measurement* on them, which yields an n-bit string (drawn from some distribution) as outcome. While the end result is of the same "type" – a distribution over bit strings – extremely complex distributions can be obtained from a small number of operations on a few qubits, possibly requiring an *exponentially larger* number of operations to simulate classically; this is at the heart of the speedups achievable by such computers.

Although the question of whether quantum computers are provably more powerful than their classical counterparts has been explored extensively, many fundamental problems remain unanswered, and those that have been settled include important caveats. On the theoretical side, it has been known for two decades that quantum algorithms able to compute a boolean function on every input can be at most polynomially more efficient than classical ones [BBC$^+$01] (in terms of *query complexity*, the main benchmark for most sublinear models of computation);[2] if the algorithm is only required to compute the function on a subset of the inputs, however,

[2]The sixth-power separation in [BBC$^+$01] has recently been improved to a tight fourth-power [ABK$^+$21] as a consequence of the *sensitivity theorem* [Hua19].

exponential separations are well known [Sim97, AA18]. Recently, moreover, an *oracle* separation between BQP (the complexity class that captures polynomial-time quantum computation) and the entire *polynomial-time hierarchy* was shown [RT22].[3]

Interest in quantum computation intensified after Shor's construction of quantum polynomial time algorithms for factorisation and discrete logarithms [Sho99], "breaking" two fundamental problems that underpin standard cryptographic hardness assumptions. On the practical side, despite a flurry of recent technical breakthroughs [AAB+19, ZWD+20, WBC+21, MLA+22] towards *quantum supremacy* – a claim that is, and is likely to remain, highly contested [HZN+20, KRS23] – any forecast of reliable, programmable large-scale quantum computation is far from certain.

1.1 A structural theorem for local algorithms

This section overviews the results of Chapter 4.

Sending information across a noisy channel is a ubiquitous problem, whose theoretical study dates back to the foundational work of Shannon [Sha48]. The solution, *error-correcting codes* (or *codes*, for short), aims to strike a balance between redundancy (to protect against corruption) and efficiency: when a k-bit message is encoded into n bits, we would like n to not be much larger than k.

But what if the receiver is only interested in part of the message? Perhaps they only wish to confirm the date of an event, or inspect the last few lines of an email to determine if they recognise the sender. In this case, standard codes offer no choice but to decode the entire message and then discard irrelevant information.

Locally decodable codes (LDCs) enable the receiver to efficiently recover only the information it requires: inspecting a few bits of the encoding suffices to decode a bit of the message. Relaxed LDCs (rLDCs) also allow the decoding algorithm to abort if it detects a corruption, a mild relaxation that enables dramatically improved constructions. Note that LDCs and rLDCs do not read the entirety of their input, and make decisions based on a small local view thereof; that is, they are *local algorithms*.[4]

However, this umbrella term groups algorithms that are in fact very distinct. Local algorithms perform fundamentally different tasks, such as testing, self-correcting, decoding, computing a local function, or verifying the correctness of a proof. Moreover, these tasks are often performed under different promises (e.g., proximity to a valid

[3]Oracle separations are obtained by endowing algorithms in both classes with the ability to solve some (fixed, possibly very complex) problem in a single time step. They provide reasonable, albeit not conclusive, evidence that the classes are indeed distinct.

[4]This terminology makes explicit that the (sublinear) parameter of interest is the algorithm's query complexity, as opposed to, say, space (as in the streaming model).

codeword in the case of LDCs, or, in the case of property testing, having either zero or large distance to a prescribed set).

Nevertheless, despite such diversity, one of our main *conceptual* contributions is capturing a fundamental structural property that is common to all of algorithms above and beyond, which in turn implies sufficient structure for obtaining Theorem 1, the main result of Chapter 4. We build on work of Fischer, Lachish and Vasudev [FLV15] as well as Gur and Lachish [GL21], which imply an essentially equivalent structure for *non-adaptive* testers and local decoders, respectively; our generalisation captures both and extends beyond them to the adaptive setting, as well as to other classes of algorithms.

More specifically, we first formalise the notion of local algorithms in the natural way: they are simply probabilistic algorithms that compute some function $f(x)$, with high probability, by making a small number of queries to the input x. We then observe that, except for degenerate cases, having a promise on the input is necessary for algorithms that make a sublinear number of queries to it. Finally, we formalise a natural robustness condition that captures this phenomenon and is shared by most reasonable interpretations of local algorithms.

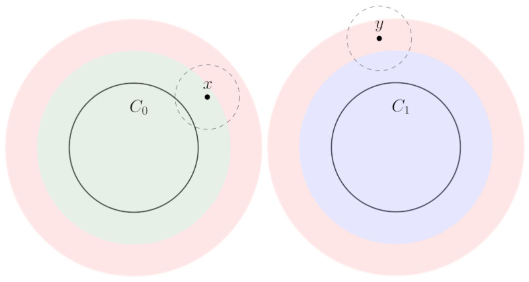
1.1.1 Robustness

We say that a local algorithm is *robust* if its output is stable under minor perturbations of the input.[5] More precisely, a (ρ_0, ρ_1)*-robust local algorithm* M for computing a partial function $f\colon \mathcal{P} \to \{0,1\}$ (where $\mathcal{P} \subset \{0,1\}^n$) is a local algorithm that satisfies the following: $M^w = 0$, with high probability, for every input w (which may or may not belong to $\mathcal{P}$) that is ρ_0-close to x such that $f(x) = 0$; and, $M^w = 1$ (w.h.p.) for every w that is ρ_1-close to x such that $f(x) = 1$.

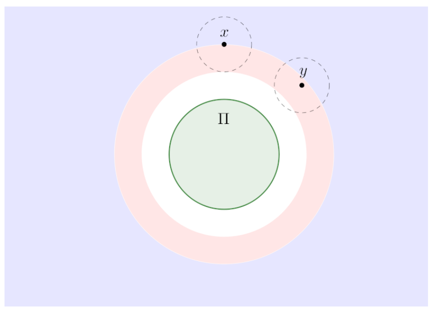
We illustrate the expressivity of robust local algorithms via two examples: property testers and locally decodable codes. We remark that, similarly, locally testable codes, locally correctable codes, relaxed LDCs, PCPs of proximity, and other notions can all be cast as robust local algorithms (see Sections 4.2.2 to 4.2.4).

Locally decodable codes. An LDC is a code that admits algorithms for decoding each individual bit of the message of a moderately corrupted codeword; that is, a code $C\colon \{0,1\}^k \to \{0,1\}^n$ with *decoding radius* δ for which there exists a probabilistic algorithm D that, given an index $i \in [k]$, makes queries to a string w promised to

[5]Note that the notion of robustness is a priori orthogonal to locality; however, as robustness is arguably the main structural property of local algorithms, we restrict the discussion to their intersection.



(a) A local decoder for the code C with decoding radius δ. Codewords whose i^{th} message bit equals 0 (resp. 1) for a fixed i comprise C_0 (resp. C_1). The decoder is robust in the $\frac{\delta}{2}$-neighbourhood of C, shaded blue and green. Inputs in the red area are within distance δ from C, but their $\frac{\delta}{2}$-neighbourhoods are not.



(b) An ε-tester for property Π. Inputs in the blue shaded area are 2ε-far from Π, where the tester is robust. While inputs in the red shaded area are rejected by the tester, this is not necessarily the case for their ε-neighbourhoods.

Figure 1.1: Casting local decoders and property testers as robust local algorithms.

be δ-close to a codeword $C(x)$ and outputs $D^w(i) = x_i$ with high probability. LDCs made a profound impact on several areas of theoretical computer science (see, e.g., [Tre04, Yek12, KS17] and references therein), and led to practical applications in distributed storage [HSX+12].

Observe that D can be viewed as a $\left(\frac{\delta}{2}, \frac{\delta}{2}\right)$-robust local algorithm for local decoding with decoding radius $\frac{\delta}{2}$, that is, for the function $f \colon [k] \times \mathcal{P} \to \{0,1\}$ where $\mathcal{P} = B_{\frac{\delta}{2}}(\operatorname{Im} C)$ is the $\frac{\delta}{2}$-neighbourhood of C.[6] This is because the $\frac{\delta}{2}$-neighbourhood of any point that is $\frac{\delta}{2}$-close to a codeword is still within the decoding radius, and thus the algorithm decodes the same value when given either a codeword or a string in its neighbourhood; see Figure 1.1a.

Property testing. Property testers are algorithms that solve approximate decision problems by only probing a minuscule part of their input, and are one of the most widely studied types of sublinear algorithms (see, e.g., the textbook [Gol17]).

An ε-tester T for a property Π makes queries to a string x and, with high probability, outputs 1 if $x \in \Pi$, and outputs 0 if x is ε-far from Π. Here, unlike with local decoders, there is no robustness at all with respect to 1-inputs:[7] we can only cast T as an $(\varepsilon, 0)$-robust local algorithm for the function $f \colon \mathcal{P} \to \{0,1\}$ where $\mathcal{P}$

[6]Note that f additionally receives a coordinate $i \in [k]$ as explicit input, which is allowed by the formal definition of robustness (see Definition 4.3).

[7]Unless the tester is *tolerant*: an $(\varepsilon_1, \varepsilon_2)$-tolerant tester is, by definition, ε_1-robust with respect to its 1-inputs.

is the union of Π and the complement of its 2ε-neighbourhood. We refer to such robustness as *one-sided.*

Note that while T does not ε-test Π robustly, it is robust when viewed as a 2ε-tester: doubling the proximity parameter ensures the ε-neighbourhood of each 0-input is still rejected (see Figure 1.1b).

By the previous discussion, our scope includes local algorithms that only exhibit *one-sided* robustness. Accordingly, we define *robust local algorithms* (without specifying parameters) as (ρ_0, ρ_1)-robust local algorithms where $\max\{\rho_0, \rho_1\} = \Omega(1)$. While dealing with one-sided robustness is significantly more technically involved, with some additional effort (see Section 4.5.2) our results also hold for this type of local algorithm.

1.1.2 Main result

By capturing structural properties that are common to all robust local algorithms, we obtain a transformation that converts them into (uniform) *sample-based algorithms.*

Sample-based algorithms are provided with uniformly distributed labeled samples, or, alternatively, query each coordinate independently with some fixed probability; they received significant attention in recent years [GR16, FGL14, BGS15, FLV15, CFSS17, BMR19b, BMR19a].[8] Adopting the latter perspective, the *sample complexity* of such an algorithm is the expected number of coordinates that it samples.

In the following, we use n to denote the input size and assume the alphabet Γ over which the input is defined is not too large (e.g., $|\Gamma| \leq n^{1/q^4}$ suffices).

Theorem 1 (Theorem 4.7, informally stated)**.** *Every robust local algorithm with query complexity q can be transformed into a* sample-based *local algorithm with sample complexity $n^{1-1/O(q^2 \log^2 q)}$.*

We stress that the robustness in Theorem 1 is only required on *part of the input space* (i.e., need only be one-sided); indeed, otherwise the structural properties captured become much more restrictive (and are not shared by, e.g., property testers).

Remark. Although Theorem 1 assumes an algorithm M that is $\Omega(1)$-robust, a weaker condition suffices. Supposing (without loss of generality) that $\rho_0 \geq \rho_1$, only *a single input x* must imply $M^w = 0$ when w is $\Omega(1)$-close to x; then the result follows even for $\rho_0 = \Theta(n^{-1/q}) = o(1)$.

[8] More accurately, these are *uniform* sample-based testers (in contrast to the [BGS15] tester, which queries coordinates in a random subspace). We adopt the original terminology of [GR16] for simplicity.

Moreover, the transformation in Theorem 1 is optimal up to a quadratic factor in the dependency on the query complexity; that is, q-query robust local algorithms cannot be transformed into sample-based algorithms with sample complexity $n^{1-1/o(q)}$ (see Theorem 4.12).

The proof of Theorem 1 relies on an analysis of the query behaviour of robust local algorithms by partitioning their local views into relaxed sunflowers and using volume lemmas implied by their robustness. We build on the Hajnal–Szemerédi theorem to analyse sampling from relaxed sunflowers (see Section 3.1 for a detailed technical overview).

By the generality of our definition, we can apply Theorem 1 to a wide family of well-studied algorithms, such as locally testable codes, locally decodable and correctable codes, relaxed LDCs, universal LTCs, PCPs of proximity, and more (see Section 4.2).

We note that [FLV15] and [GL21] obtain an essentially equivalent transformation for testers and decoders, respectively, through "lossy" versions of our relaxed sunflower lemmas (they extract *one* relaxed sunflower from the local views, rather than partition them) that applies to non-adaptive algorithms; by a trivial transformation from adaptive to non-adaptive algorithms that incurs an exponential increase in the query complexity, these previous works show transformations whose sample-based algorithms have complexity $n^{1-1/\exp(q)}$, which we reduce to $n^{1-1/\operatorname{poly}(q)}$ (indeed, as far down as $n^{1-1/\tilde{O}(q^2)}$).

1.1.3 Applications

We proceed to the main applications of Theorem 1, which range over three fields of study: coding theory, property testing and probabilistic proof systems.

Relaxed locally decodable codes. Despite the success and attention that LDCs received since their systematic study was initiated by Katz and Trevisan [KT00], the best construction of $O(1)$-query LDCs has *super-polynomial* blocklength (cf. [Efr12], building on [Yek08]). This barrier led to the study of *relaxed* LDCs, introduced in the foundational work of Ben-Sasson, Goldreich, Harsha, Sudan, and Vadhan [BGH⁺06]. In a recent line of research, relaxed LDCs and variants thereof were applied to PCPs [MR10, DH13, RR20a], property testing [CG18], data structures [CGW13] and probabilistic proof systems (e.g., [GR17]; see also Section 5.2.2).

Loosely speaking, this relaxation allows the local decoder to abort on a small fraction of the indices, yet, crucially, still avoid errors. More accurately, a *relaxed* LDC $C\colon \{0,1\}^k \to \{0,1\}^n$ with *decoding radius* δ is a code that admits a probabilistic algorithm D (a decoder) which, on input index $i \in [k]$, queries a string $w \in \{0,1\}^n$ that is δ-close to a codeword $C(x)$, and satisfies the following: (1) if the input is a valid codeword (i.e., $w = C(x)$), the decoder outputs x_i with high probability; and (2) otherwise, with high probability, the decoder must either output x_i or an "abort" symbol $\perp$, indicating it detected an error and is unable to decode.[9]

This seemingly modest relaxation allows for obtaining dramatically stronger parameters. Indeed, [BGH+06] constructed a q-query relaxed LDC with blocklength $n = k^{1+1/\Omega(\sqrt{q})}$, and raised the problem of whether it is possible to obtain better rates; the best known construction, obtained in recent work of Asadi and Shinkar [AS21], improves it to $n = k^{1+1/\Omega(q)}$. We stress that *proving lower bounds on relaxed LDCs is significantly harder than on standard LDCs*, and indeed, the first non-trivial lower bound was only recently obtained in [GL21]. It shows that, to obtain query complexity q, the code must have blocklength

$$n \geq k^{1+\frac{1}{O(2^{2q}\cdot\log^2 q)}} .$$

This shows that $O(1)$-query relaxed LDCs cannot obtain quasilinear length, a question raised in [Gol11], but still leaves exponential room for improvement in the dependency on query complexity (note that even for $q = O(1)$ this strongly affects the asymptotic behaviour). Indeed, eliminating this exponential dependency was raised as the main open problem in [GL21].

Since the robustness framework captures *relaxed* LDCs as well, we resolve the aforementioned open problem by obtaining a lower bound with *exponentially* better dependency on the query complexity. Along the way, we also extend the lower bound to hold for relaxed decoders with *two-sided error*, resolving another problem left open in [GL21].

Theorem 2 (Corollary 4.3, informally stated)**.** *Any relaxed locally decodable code $C\colon \{0,1\}^k \to \{0,1\}^n$ with constant decoding radius δ and query complexity q must have blocklength at least*

$$n \geq k^{1+\frac{1}{O(q^2\log^2 q)}} .$$

This also makes significant progress towards resolving the problem due to [BGH+06], by narrowing the gap between lower and upper bounds to merely a quadratic factor.

[9]As observed in [BGH+06], these two conditions suffice for obtaining a third condition which guarantees that the decoder only outputs $\perp$ on an arbitrarily small fraction of the coordinates.

Property testing. As an immediate corollary of Theorem 1, we obtain that any constant-query testable property (up to $\sqrt[5]{\log n}$-query, in fact) admits a sample-based tester with sublinear sample complexity.

Theorem 3 (Corollary 4.1, informally stated)**.** *Any property $\Pi \subseteq \{0,1\}^n$ that is ε-testable with q queries admits a sample-based 2ε-tester with sample complexity $n^{1-1/O(q^2 \log^2 q)}$.*

Using Theorem 3, we also show an application to *multi-testing* (Corollary 4.2), where the goal is to simultaneously test a large number of properties (namely, up to $k = \exp\left(n^{1/\omega(q^2 \log^2 q)}\right)$ of them) that are each testable with q adaptive queries.

Proofs of proximity. Proofs of proximity [RVW13] are probabilistic proof systems that allow for delegation of computation in sublinear time. They were studied extensively in recent years, finding applications in cryptography with both computational [KR15] and information-theoretic security [RRR21, BRV18].

In the non-interactive setting, we have a verifier that wishes to ascertain the validity of a given statement, using a short (sublinearly long) explicitly given proof, and a sublinear number of queries to its input. Since the verifier cannot even read the entire input, it is only required to reject inputs that are far from being valid. Thus, the verifier is only assured of the proximity of the statement to a correct one. Such proof systems can be viewed as the NP (or, more accurately, MA) analogue of property testing, and are referred to as MA proofs of proximity (MAPs).

As such, one of the most fundamental questions regarding proofs of proximity is their relative strength in comparison to testers; that is, whether verifying a proof for an approximate decision problem can be done significantly more efficiently than solving it. One of the main results in [GR18] is that this can indeed be the case. Namely, there exists a property Π which: (1) admits an adaptive MAP with proof length $O(\log n)$ and query complexity $q = O(1)$; and (2) requires at least $n^{1-1/\Omega(q)}$ queries to be tested without access to a proof.[10]

In Section 4.5.3 we use Theorem 1 to show that the foregoing separation is nearly tight.

Theorem 4 (Theorem 4.11, informally stated)**.** *Any property $\Pi \subseteq \{0,1\}^n$ that admits an adaptive MAP with query complexity q and proof length p also admits a tester with query complexity $p \cdot n^{1-1/O(q^2 \log^2 q)}$.*

[10]We remark that the bound in [GR18] is stated in a slightly weaker form. However, it is straightforward to see that the proof achieves the bound stated above. See Section 4.5.3.

Interestingly, we remark that we rely on Theorem 4 to prove the (near) optimality of Theorem 1 (see Section 4.5.3 for details).

1.1.4 Open problems

There remain several interesting directions and open problems related to robust local algorithms that we wish to highlight. Firstly, we stress that our structural theorem is extremely general, and indeed the robustness condition that induces the structure required by Theorem 1 appears to hold for most reasonable interpretations of robust local algorithms. It would be interesting to see whether our framework (or a further generalisation of it) could imply applications to other families of local algorithms, such as PAC learners, local computation algorithms (LCAs), and beyond.

Open Problem 1. Can Theorem 1 and the framework of robust local algorithms be used to obtain query-to-sample transformations for PAC learners and LCAs?

One promising direction that we did not explore is on rate lower bounds for PCPs of proximity (PCPPs). Such bounds are notoriously hard to get, and indeed the only such bounds we are aware of are those in [BHLM09], which are restricted to special setting of 3-query PCPPs. Since our framework captures PCPPs, and in light of the rate lower bounds it allowed us to obtain for relaxed LDCs, it seems feasible to obtain rate lower bounds on PCPPs as well.

Open Problem 2. Can we obtain rate lower bounds on q-query PCPs of proximity for $q > 3$?

Another interesting question involves the optimality of our transformation. Recall that Theorem 1 transforms q-query robust local algorithms into sample-based local algorithms with sample complexity $n^{1-1/O(q^2 \log^2 q)}$, whereas in Section 4.5.3 we show that any such transformation must yield an algorithm with sample complexity $n^{1-1/\Omega(q)}$. This still leaves a quadratic gap in the dependency on query complexity. We remark that closing this gap could lead to fully resolving an open question raised in [BGH+06] regarding the power of relaxed LDCs.

Open Problem 3. What is the optimal sample complexity obtained by a transformation from robust local algorithms to sample-based local algorithms?

Note, moreover, that we focus on query (or sample) complexities and provide a computationally inefficient transformation, iterating over exponentially many input strings; the *computational* cost of such transformations is an interesting problem in its own regard.

Open Problem 4. Are there efficient transformations from robust to sample-based algorithms?

1.2 Complexity separations in quantum property testing

This section overviews Chapter 5.

Is it easier to check a candidate solution for a problem than computing one from scratch? The *decision vs. verification problem* has long been a powerful driving force in theoretical computer science. P vs. NP is but one facet of the question, whose analogues are major open problems in the settings of randomised algorithms (BPP vs. MA) and quantum computation (BQP vs. QMA), among a multitude of computational models. Indeed, the study of proof systems either in the quantum setting or in (classical) property testing is well established.[11]

Exploration of the power of quantum algorithms for verification (rather than decision) began with Kitaev and Watrous [Wat00, KW00], who defined the complexity class QMA. The setting is as follows. A polynomial-time algorithm ("Arthur") with quantum capabilities receives a *classical* bit string x as input and must decide whether it belongs in a language L; to do so, it receives an additional quantum state as a *proof* from an all-powerful agent ("Merlin") and must satisfy the following with high probability: if $x \in L$, there exists *at least* one quantum state that convinces Arthur of this fact; but if $x \notin L$, then every quantum state fails to convince Arthur otherwise.

Several features of QMA have been mapped out, but some – in particular the relationship to its classical counterpart, MA – remain elusive. For example, as opposed to MA, it is not known whether QMA can achieve perfect completeness (although there exists an oracle separation [Aar09]). Likewise, it is not known whether problems verifiable by quantum algorithms provided with a classical proof (QCMA) are a strict subset of QMA, but there exists an oracle separation between the two as well [AK07].

One may consider the setting where *both* quantum resources and interaction are allowed; then certain classical results translate to the quantum setting, but others do not: with interaction, one may achieve perfect completeness [KW00] (in contrast to QMA); on the other hand, quantum resources provide at most a polynomial advantage, as $\mathsf{QIP} = \mathsf{IP}$ [Wat03]. We refer to the survey [VW16] for further discussion of quantum proofs, but highliht that the breakthrough characterisation $\mathsf{MIP}^* = \mathsf{RE}$

[11]The standard terminology in property testing is *proofs of proximity*, which include, among others, PCPs of proximity, interactive proofs of proximity, and MA proofs of proximity (MAPs). We henceforth adopt such terminology, noting it is synonymous with *proof systems in the property testing setting.*

[IV12, NW19, JNV$^+$21] is the culmination of a rich line of work on interactive protocols with *multiple quantum provers*.

Quantum property testing. Quantum property testing is a fundamental model of sublinear-time quantum computation. Its importance stems both from the practical difficulty in manipulating large quantum states, as well as from the fertile ground that it provides for complexity theoretic investigations of the power of quantum mechanics as a computational resource. Accordingly, this model has garnered a large amount of attention in the last decade (see, e.g., [CFMW10, HA11, ACL11, CM13, OW15, ABRW16, NV17, AA18, BOW19, GL20, BCL20], and the survey [MW16]).

Accordingly, quantum testers are defined as quantum query algorithms that solve the approximate decision problem of membership in a subset Π (of possibly quantum objects); that is, the tester must accept if its input is in the property Π and reject if it is *far* from Π with respect to a natural metric.[12]

Recall that property testing problems are a special case of promise problems, which suggests that quantum algorithms may outperform classical ones on (at least some) testing tasks. There is more than one way the problem can be "made quantum", however: we may (1) test classical properties with a quantum algorithm;[13] (2) test quantum properties with a classical algorithm; or (3) test quantum properties quantumly. We summarise some results of the first type (which is our focus) below, and refer to the survey [MW16] for the others.

The technique of amplitude amplification [BHMT02] may be applied to any tester with perfect completeness to immediately obtain a speedup (see Section 5.4); in particular, the trivial ε-tester that checks identity to some fixed string can have its query complexity improved from $\Theta(1/\varepsilon)$ to $\Theta(1/\sqrt{\varepsilon})$, and so can the well-known tester for linearity of boolean functions [BLR93].

For classically testing if a boolean function is a k-junta (i.e., depends on at most k coordinates of its input), it is known that $\tilde{\Theta}(k)$ queries are both necessary [CG04] and sufficient [Bla09] (for a constant proximity parameter ε). Quantum testing, however, can be performed with $O(\sqrt{k})$ queries [ABRW16], below the classical lower bound.

Distribution testing can also be sped up with quantum resources: testing

[12]Note that, unlike in the classical case, where Hamming distance is with few exceptions the natural choice, there are many natural metrics on the set of unitary matrices (e.g., those induced by the operator or Hilbert-Schmidt norms).

[13]A quantum tester has access to an n-bit string x via a *query oracle* that extends a classical query the natural way: when the algorithm sends the state $|i\rangle|0\rangle$ to the oracle, it receives the state $|i\rangle|x_i\rangle$ (and this extends to superpositions by linearity). The tester's output is the measurement outcome of some fixed qubit.

if a distribution over $[n]$ is uniform requires $\Omega(\sqrt{n})$ samples [GR11], but $O(n^{1/3})$ quantum "samples" suffice [BHH11, CFMW10] (with constant proximity parameter, and an appropriate definition of a quantum sample); moreover, for testing whether two distributions are identical, a classical lower bound of $\Omega(n^{2/3})$ [Val11] can be overcome by a quantum algorithm with query complexity $O(\sqrt{n})$ [BHH11].

Chapter 5 is concerned with the notion of QMA proofs, the quantum analogue of NP proofs, in property testing. Namely, we investigate the following question:

What is the power of QMA proofs for quantum property testing?

We show that such *quantum proofs of proximity* (i.e., quantum testers with additional access to a proof, or, equivalently, quantum proof systems in the property-testing setting) are an extremely interesting class of algorithms: they achieve improved parameters for a large family of properties (those that are *decomposable*; see Theorem 5.14) and have a rich complexity-theoretic landscape (see Figure 1.2).

1.2.1 Quantum proofs of proximity

A *Quantum Merlin-Arthur* (QMA) proof of proximity protocol for a property of unitaries Π, with respect to proximity parameter ε, is naturally defined as follows. The *verifier*, a computational device given oracle access to a unitary U, receives a quantum state $|\psi\rangle$ from an all-powerful but untrusted *prover*. Making use of these two resources, it must decide whether $U \in \Pi$ or U is ε-far from Π with respect to a specific metric. Such a protocol is said to verify (or *test*) Π if, with high probability, the verifier accepts in the former case and rejects in the latter (see Section 5.1 for a formal definition). We remark that the notion of QMA proofs of proximity is implicit in the literature as QMA query algorithms for approximate decision problems (e.g., the permutation testing problem [Aar12, ST23]). In Chapter 5 we initiate the systematic study of the notion of QMA proofs of proximity (QMAPs), and explore its power and limitations.

The complexity of a QMAP protocol is measured with respect to the amount of resources required by the verifier. Namely, we will evaluate the efficiency of a protocol by its *proof complexity* p (the number of qubits in the proof $|\psi\rangle$) and *query complexity* q (the number of oracle calls made by the verifier for a worst-case input U). In particular, both parameters should be *sublinear* in nontrivial protocols.

1.2.2 Our results

Our main results are divided into two parts: we first chart fundamental aspects of the complexity landscape surrounding quantum proofs of proximity; and then proceed to algorithmic results, where we show sufficient conditions for properties to admit efficient QMAP protocols.

We write $\mathsf{QMAP}(\varepsilon, p, q)$ for the class of ε-testable properties by a QMA proof of proximity protocol with proof length p and query complexity q (acronyms in regular font refer to algorithms and protocols, while, e.g., QMAP and MAP denote complexity classes).

Complexity separations. Our first collection of results aims to chart the complexity landscape of quantum proofs of proximity. Recall that $\mathsf{QMAP}(\varepsilon, p, q)$ is the class of ε-testable properties by a QMAP with proof complexity p and query complexity q. The classes MAP (MA proofs of proximity) and IPP (interactive proofs of proximity) are defined analogously. $\mathsf{PT}(\varepsilon, q)$ and $\mathsf{QPT}(\varepsilon, q)$ are the properties admitting classical and quantum ε-testers with query complexity q, respectively, and $\mathsf{QCMAP}(\varepsilon, p, q)$ is the restriction of $\mathsf{QMAP}(\varepsilon, p, q)$ where the proofs are classical bit strings. Formal definitions of all of these classes can be found in Section 2.4 and Definition 5.1.

We write complexity classes with the parameters omitted (e.g., QMAP) to denote the corresponding class of properties such that for some proximity parameter $\varepsilon \in (0, 1)$ that is a universal constant, there is a protocol with proof and query complexities bounded by $\operatorname{polylog}(n)$.

We begin by showing the existence of a property that admits efficient QMAPs, yet neither quantum property testers nor MAPs can efficiently test it.

Theorem 5. *There exists a property* Π *such that, for any small enough* $\varepsilon = \Omega(1)$*,*

$$\Pi \in \mathsf{QMAP}\big(\varepsilon, \log n, O(1)\big) \text{ and}$$
$$\Pi \notin \mathsf{QPT}\big(\varepsilon, o(n^{0.49})\big) \cup \mathsf{MAP}(\varepsilon, p, q)$$

when $p \cdot q = o(n^{1/4})$*. In particular,*

$$\mathsf{QMAP} \not\subseteq \mathsf{QPT} \cup \mathsf{MAP} \ .$$

Theorem 5 is, in fact, implied by a stronger result. We show that, for certain properties, MAPs are stronger than quantum testers ($\mathsf{MAP} \not\subseteq \mathsf{QPT}$, Theorem 5.5); and, for others, quantum testers are stronger than MAPs ($\mathsf{QPT} \not\subseteq \mathsf{MAP}$, Theorem 5.6). Combining these results, we conclude that $\mathsf{QCMAP} \not\subseteq \mathsf{QPT} \cup \mathsf{MAP}$, i.e., even QMAPs with classical proofs suffice to obtain an exponential speedup over both MAPs and quantum testers.

Having shown the aforementioned separation, a natural question poses itself: are there cases in which a *quantum proof* cannot be substituted for a classical one? We observe that a straightforward adaptation of a known result shows this is indeed the case, albeit *only for subconstant proximity parameters*: the QMA vs. QCMA oracle separation of Aaronson and Kuperberg [AK07] carries over to the property testing setting, implying $\mathsf{QMAP}(1/\sqrt{n}, \log n, 1) \not\subseteq \mathsf{QCMAP}(1/\sqrt{n}, p, q)$ if $\sqrt{pq} = o(\sqrt{n})$ (see Section 5.7 for details).

We then shift to proving *limitations* on the algorithmic power of QMAPs, showing that there exist explicit properties that are extremely difficult for such protocols. First, we observe that known lower bounds on the complexity of QMA protocols for the *Permutation Testing* problem [Aar12, ST23] yield an explicit property that does not have any QMA protocol of polylogarithmic proof length and query complexity, and establishes that $\mathsf{IPP} \not\subseteq \mathsf{QMAP}$ (see Section 5.8 for details). We thus obtain the complexity landscape shown in Figure 1.2.

Finally, we present an entire class of properties that cannot be solved by efficient QMAP protocols. This extends and simplifies one of the main results of [FGL14], which obtains the same result for classical MAPs.

Theorem 6 (Corollary 5.1, informally stated)**.** *If a non-trivial property Π is k-wise independent and $\varepsilon = \Omega(1)$ is sufficiently small, then $\Pi \notin \mathsf{QMAP}(\varepsilon, p, q)$ when $pq = o(k)$.*

Algorithms. We show two general classes of properties whose structure allows for efficient QMAP (QMA proof of proximity) protocols. Moreover, these protocols only require *classical* proofs (though the verifier is quantum).[14] The first class is comprised of what we call *decomposable properties*, which generalise the "parameterised concatenation properties" introduced in [GR18].

Roughly speaking, a property Π is (k, s)-decomposable if testing whether $x \in \Pi$ can be reduced, with a message of length s from the prover, to that of testing whether $x^{(i)} \in \Lambda^{(i)}$ for k smaller strings $x^{(i)}$ and properties $\Lambda^{(i)}$ (see Definition 5.8). Since there may be many decompositions of the same string, the prover message is said to *specify* a decomposition, i.e., a mapping $x \mapsto x^{(i)}$ and $\Pi \mapsto \Lambda^{(i)}$.

[14]Equivalently, all of the results in this section rely on QCMAP protocols. They are thus slightly stronger, continuing to hold if we replace QMA by QCMA in the theorem statements.

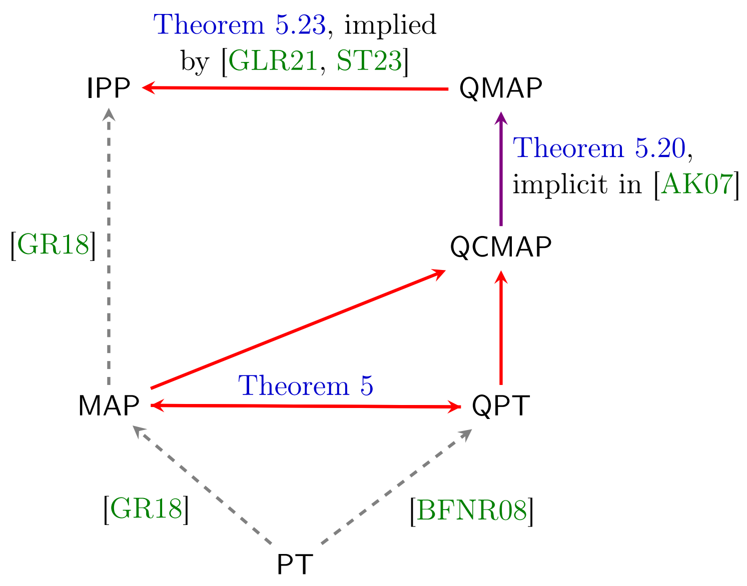

Theorem 5.23, implied by [GLR21, ST23]
IPP
QMAP
Theorem 5.20, implicit in [AK07]
[GR18]
QCMAP
Theorem 5
MAP
QPT
[GR18]
[BFNR08]
PT

Figure 1.2: Classification of complexity classes. An arrow from $\mathcal{A}$ to $\mathcal{B}$ indicates a property requiring $n^{\Omega(1)}$ proof length or query complexity by algorithms of $\mathcal{A}$ but only polylog(n) proof/query complexity by algorithms of $\mathcal{B}$ (coloured red or grey when with respect to a proximity parameter $\varepsilon = \Omega(1)$ that is a universal constant; and violet when $\varepsilon = \Omega(1/\sqrt{n})$.) The (dashed) grey arrows are previously known separations.

Theorem 7 (Theorem 5.14, informally stated)**.** *If a property* Π *is* (k, s)*-decomposable into strings of length* m*, each of which is* ε*-testable by a MAP protocol with proof complexity* p *and query complexity* $q = q(m, \varepsilon) = m^\alpha/\varepsilon^\beta$*, then*

$$\Pi \in \mathsf{QMAP}(\varepsilon, s + kp, q') ,$$

where q' *is much smaller than* q *for many parameter values* α *and* β*.*

As applications of Theorem 5.14, we show: a QMAP for k-monotonicity that is more efficient than the best known (classical) testers and MAPs for a wide range of parameters (in this case $\alpha = 0$ and $\beta = 1$; see Corollary 5.6 and the discussion in Section 5.5.2); and a QMAP for the property of Eulerian graph orientations (Corollary 5.5, where $\alpha = 1$ and $\beta = 0$) with a quadratic speedup over the best known MAP [GR18].

The second class of properties amenable to QMA proofs of proximity are those admitting *one-sided* (classical) MAPs which do not receive a proximity parameter explicitly, but rather reject strings ε-far from the property with probability that is a function of ε. Such algorithms are called *proximity-oblivious* MAPs and readily admit quantum speedups (via the technique of amplitude amplification; see Section 5.4).

Theorem 8. *If a property Π admits a proximity-oblivious MAP protocol with proof complexity p and query complexity q, which always accepts $x \in \Pi$ and rejects when x is ε-far from Π with probability $\rho(\varepsilon) > 0$, then*

$$\Pi \in \mathsf{QMAP}\left(\varepsilon, p, O\left(\frac{q}{\sqrt{\rho(\varepsilon)}}\right)\right) .$$

Applying Theorem 8, we obtain quadratically better QMAPs for read-once branching programs and context-free languages (Corollaries 5.3 and 5.4) as compared to the best-known classical MAPs [GGR18].

Exact decision. Classically, casting *exact decision* problems in the framework of proofs of proximity (i.e., testing with respect to proximity parameter $\varepsilon = 1/n$) is completely trivial except for degenerate cases, as most functions of sublinear query complexity are extremely simple. Rather surprisingly, this is *not* the case quantumly, and indeed, setting $\varepsilon = 1/n$ in the aforementioned corollaries of Theorems 7 and 8 (the applications to k-monotonicity, graph orientations, branching programs and context-free languages in Corollaries 5.3 to 5.6) yields sublinear algorithms for the corresponding exact decision problems. For *layered* branching programs, we also prove Theorem 5.17, which improves on the parameters of Corollary 5.3 and lifts the read-once restriction.

Lastly, we prove that QMAP protocols are useful beyond proximity-oblivious and decomposable properties. The problem of testing bipartiteness of a graph does not fit either class, yet admits an efficient protocol nonetheless (Theorem 5.18).

1.2.3 Open problems

We merely begin the exploration of quantum proofs of proximity, leaving a host of uncharted research directions. We wish to highlight a small number of open problems of particular interest.

In Figure 1.2, the diagram of complexity class separations, an evident shortcoming is the absence of $\mathsf{QMAP} \not\subseteq \mathsf{QCMAP}$ in the natural setting of parameters, i.e., with proximity $\varepsilon = \Omega(1)$ rather than $\varepsilon = \Theta(1/\sqrt{n})$.

Open Problem 5. What is the largest proximity parameter ε such that

$$\mathsf{QMAP}\big(\varepsilon, \mathrm{polylog}(n), \mathrm{polylog}(n)\big) \not\subseteq \mathsf{QCMAP}(\varepsilon, p, q)$$

with some proof and query complexities satisfying $pq = n^{\Omega(1)}$?

Given our focus on quantum MA (i.e., *non-interactive*) proofs of proximity, it is natural to ask what is achievable by allowing quantum property testers to interact with quantum provers, as opposed to static proofs.

Open Problem 6. What is the power of quantum IP proofs of proximity (QIPPs)?

More specifically, it is known that there exist classical interactive proof of proximity (IPP) protocols with $\tilde{O}(\sqrt{n})$ proof and query complexities for large classes of languages [RVW13, RR20b]. Moreover, these complexities are optimal (up to polylogarithmic factors) for *classical* protocols, under reasonable cryptographic assumptions [KR15]. Could quantum interactive proofs break the square-root barrier?

Open Problem 7. Can QIPPs test logspace-uniform NC languages with $o(\sqrt{n})$ proof and query complexities?

Finally, while we show a strong lower bound for QMAPs for k-wise independent properties, they do not rule out the existence of sublinear QMAP protocols for such properties. Could a stronger lower bound be shown?

Open Problem 8. Do there exist maximally hard properties for QMAPs, requiring $\Omega(n)$ query complexity when the proof complexity is $p = cn$ for some $c = \Omega(1)$?

We note that the question has an easy (negative) answer if we take $c = 1$: with a proof (allegedly) equal to the input string x, a QMAP based on Grover search can test with $O(1/\sqrt{\varepsilon}) = O(\sqrt{n})$ queries. Moreover, [RS04] provides a (positive) answer for proximity parameter $\varepsilon = 1/n$: there exists a property $\Pi \subset \{0,1\}^n$ for which deciding whether $x \in \Pi$ or $x \notin \Pi$ requires proof and query complexities satisfying $p + q = \Omega(n)$ (indeed, this is true of most bipartitions of $\{0,1\}^n$). Then taking $c = \Omega(1)$ small enough implies $q = \Omega(n)$. Thus, similarly to Open Problem 5, while we have answers for subconstant ε, the problem is open when $\varepsilon = \Omega(1)$.

1.3 Streaming zero-knowledge

This section discusses Chapter 6.

Processing more information than one is able to store is an increasingly common requirement for real-world algorithms. From network analytics to particle colliders, algorithms must often read and quickly summarise bursts of information – or risk missing the next. Tasks of this type are solved by *streaming* algorithms, which have a small amount of memory and *one-pass* access to a massive data stream.

Zero-knowledge protocols enable a computational party to prove a mathematical statement without revealing *anything* other than the fact that it is true. They are fundamental objects of study in theoretical computer science, having provided a highly fertile ground for investigations in complexity theory and cryptography (see, e.g., [Vad99, Gol02, Vad07, Gol08] and references therein), and led to consequential practical applications [BCG+13, BCG+14, BBHR18, BCG+19, BCR+19] (see also the survey [SYZ+21] and references therein).

In recent years, interactive proofs in the data stream model received a great deal of attention [CTY11, CMT12, CMT13, Tha13, CCGT14, CCM+15, Tha16, DTV15, CH18, CG19, CGT20]. Streaming interactive proofs (SIPs) are proof systems where the computationally bounded party is bounded *not* in its time complexity, but rather in space and input access. More precisely, an SIP is an interactive protocol between a powerful but untrusted prover P and a space-bounded *streaming verifier* V whose sequential, one-pass access applies to the input as well as the prover's messages. (That is, V receives its input x as a stream of symbols $x_1, x_2, \ldots, x_n$; and likewise for messages sent by P.) We note that prover and verifier observe the same stream, which only the former can store in its entirety.

Remarkably, SIPs allow low-space streaming algorithms to efficiently verify key problems in the data stream model that are completely intractable without the assistance of a prover. Indeed, the aforementioned sequence of works constructed SIPs with polylogarithmic-space verifiers for a large collection of problems, many of which require linear space for a streaming algorithm alone (such as the INDEX and frequency moment problems). The underlying power that enables current constructions of SIPs to achieve exponentially improved space complexity essentially boils down to two powerful and expressive protocols: *sumcheck* and *polynomial evaluation*, which can in turn be applied to a plethora of problems.

Yet, despite the extensive study of streaming interactive proofs over the last decade, no zero-knowledge SIPs were known prior to our work. Indeed, it is not obvious a priori whether this notion is at all possible: for instance, while traditional zero-knowledge prevents leakage of information to a polynomial-time adversary about some hard computation on an input x (e.g., about a witness that certifies x is in a language), in the streaming setting a space-bounded verifier must learn no additional information *about x itself* – even if its runtime is unbounded.

Zero-knowledge in the streaming model. Recall that in the traditional setting, which deals with *polynomial-time* algorithms, a protocol is zero-knowledge if, for every (possibly malicious) verifier $\widetilde{V}$, there exists a simulator $S_{\widetilde{V}}$ whose output cannot be told apart (either computationally or statistically) from a real interaction between P and $\widetilde{V}$ by any distinguisher D; and if this holds up to negligibly small error, the protocol can be safely repeated or composed.

In the streaming model, algorithms are restricted to *one-pass sequential access* to their input and the primary resource is *space*, rather than time. Accordingly, we say that an SIP is zero-knowledge if $\widetilde{V}$, S and D are streaming algorithms; when $\widetilde{V}$ has s bits of memory, the simulator has roughly s space and we allow the distinguisher D to have an arbitrary $\text{poly}(s)$ amount of memory. (See Section 6.1 for formal definitions.) Albeit similar, this notion is distinct to its poly-time analogue in two fundamental ways.

Negligible distinguishing bias is a robust notion of security in the setting of polynomial-time computation because it prevents polynomial-time adversaries from boosting their advantage by repeating (polynomially) many executions. However, in the data stream model, the one-pass restriction on input access precludes this strategy altogether; indeed, streaming problems often become trivial with a single additional pass. We therefore define secure protocols as those achieving $o(1)$ distinguishing bias, which ensures that the probability of information leakage tends to zero. (See Remark 6.1 for a more detailed discussion of alternative "hybrid" models and security bounds.)

The second crucial distinction is that the notion of zero-knowledge for SIPs is *unconditional*, i.e., does not rely on computational assumptions, faithfully to the nature of the data stream model. This differs markedly from past work on zero-knowledge protocols where the verifier is able to process incoming messages in a streaming fashion (e.g., [GKR15, CMT12]), whose zero-knowledge property is still with respect to the standard setting: while the honest verifier is a streaming algorithm, the protocols are only secure against polynomial-time adversaries. In this work, *adversaries are also streaming algorithms*.

Chapter 6 explores the extent to which zero knowledge streaming interactive proofs (zkSIPs) can outperform streaming algorithms: does there exist a problem they solve more efficiently? If so, can they do so for a natural problem such as INDEX, or even more ambitiously, achieve an exponential reduction in the space complexity for key problems in the data stream model?

1.3.1 Main results

The main contribution of Chapter 6 is a strong positive answer to the questions above, providing the tools to construct zero-knowledge streaming interactive proofs for essentially any problem within the reach of current (non-zero-knowledge) SIPs.

In more detail, our main results are zero-knowledge versions of the two building blocks underlying all known SIPs: the *sumcheck* and *polynomial evaluation* protocols, from which we derive zkSIPs for central streaming problems in Section 1.3.2. In doing so, we obtain *exponentially* smaller space complexity for the fundamental INDEX and frequency moment problems (among others) when compared to streaming algorithms alone.

We remark that all our zkSIPs are two-stage protocols with a *setup* and an *interactive* stage. The setup is non-interactive and consists merely of a random string (see Section 3.3.3), which can be reused in multiple interactive executions (of possibly different protocols). With this simple preprocessing step, we achieve essentially optimal time and communication complexities (i.e., subpolynomial or even polylogarithmic – as do the best non-zero-knowledge SIPs – and dramatically smaller than the complexity of streaming the input) in the interactive stage.

Sumcheck Zero-Knowledge SIP. The following theorem gives a zero-knowledge *sumcheck* SIP, which allows a streaming algorithm to decide whether the sum of evaluations of a low-degree polynomial over a large structured set (a subcube) matches some prescribed value. Sumcheck is one of the most important interactive proof protocols, and is extremely useful for SIPs in particular.

We state the following theorem (see Theorems 6.10 and 6.11 for the formal version) in generality, but note that standard parameter settings imply space complexity $s = \mathrm{polylog}(n)$ as well as $O(n^{1+\delta})$ (for any constant $\delta > 0$) and $n^{o(1)}$ communication in the setup and interactive stages, respectively. (The time complexity is of the same order as the communication in both stages.) This is the case in all of our applications.

Theorem 9 (Sumcheck zkSIP)**.** *For every dimension m, field $\mathbb{F}$ and evaluation set H, there exists a sumcheck zkSIP where the verifier streams an m-variate low-degree polynomial f and computes $\sum_{\boldsymbol{\beta} \in H^m} f(\boldsymbol{\beta})$ with $s = O(m^2 \log |\mathbb{F}|)$ bits of space. The protocol communicates $\tilde{O}(|\mathbb{F}|^m)$ bits in its setup stage and $|\mathbb{F}|^{\log\log|\mathbb{F}|} \cdot \mathrm{poly}(|\mathbb{F}|)$ bits in the interactive stage.*

The round complexity (the number of messages sent or received by each party throughout the SIP) is $m + O(1)$, a small constant larger than that of the standard sumcheck protocol.

We stress that while sumcheck is traditionally used (in the polynomial-time setting) to verify exponentially large sums in polynomial time, this is *not* the goal of the streaming variant, as sums of evaluations over a large set can be obtained incrementally for functions computable in low space (a class that includes polynomials).

Nevertheless, the sumcheck protocol achieves exponential savings in space complexity for problems that require large space without interaction: it enables efficient verification of sums of polynomials that an input defines implicitly, which require *linear space* to compute otherwise.

Polynomial Evaluation Zero-Knowledge SIP. We proceed to our second main result, a zero-knowledge streaming *polynomial evaluation* protocol. (See Theorems 6.7 and 6.8 for the formal statements.) It allows a streaming algorithm to access data that was already seen but not stored, by saving a small *fingerprint* of the stream. Similarly to sumcheck, this is a general-purpose protocol that is widely applicable to the design of SIPs.

Theorem 10 (Polynomial evaluation zkSIP)**.** *For every dimension m and field $\mathbb{F}$, there exists a polynomial evaluation zkSIP where the verifier streams an m-variate low-degree polynomial f followed by an evaluation point $\boldsymbol{\beta}$ and computes $f(\boldsymbol{\beta})$ with $O(m \log |\mathbb{F}|)$ bits of space. The communication complexity is $\tilde{O}(|\mathbb{F}|^m)$ in the setup and* $\operatorname{poly}(|\mathbb{F}|)$ *in the interactive stage.*

As in Theorem 9, standard parameter settings imply zkSIPs with polylogarithmic space, $n^{o(1)}$ time and communication complexity (in the interactive stage)[15] as well as near-linear communication in the setup. The round complexity is $O(1)$.

Streaming commitment protocols. En route to proving Theorems 9 and 10, we construct tools for the design of zkSIPs which we find of independent interest. Namely, we provide two types of *commitment protocols* for streaming algorithms.

We remark that in the polynomial-time setting, the existence of secure commitment schemes is equivalent to the existence of one-way functions [IL89, Nao91, HILL99], so it may seem surprising that our results hold *unconditionally*. However, in the incomparable model of streaming algorithms, which are not time-bounded, but are instead severely constrained with respect to space and input access, we show that no cryptographic assumption is needed.[16]

The following result shows that not only does a streaming commitment protocol exist, but that it can be made *linear*; that is, the sender may commit to a sequence of messages and decommit to a linear combination thereof, with linear coefficients of the receiver's choosing.

[15]A nontrivial security guarantee still holds with $\operatorname{polylog}(n)$ communication, but with $n^{o(1)}$ the protocol becomes secure against arbitrary $\operatorname{polylog}(n)$-space adversaries; see Remark 6.9.

[16]We refer to commitment *protocols* rather than schemes in the streaming model to avoid ambiguity with the polynomial-time analogue; see Definition 6.5.

Theorem 11 (Theorem 6.5, informally stated)**.** *There exists a commitment protocol whereby an unbounded-space sender commits a tuple $\boldsymbol{\alpha} \in \mathbb{F}^{\ell}$ to a streaming receiver and decommits to a linear combination $\boldsymbol{\alpha} \cdot \boldsymbol{\beta}$, with linear coefficients $\boldsymbol{\beta}$ chosen by the receiver. The receiver's space complexity is $O(\ell \log |\mathbb{F}|)$ and the protocol communicates $\tilde{O}\left(|\mathbb{F}|^{3\ell}\right)$ bits.*

The second component is a new notion of a streaming commitment, which we call *temporal*. This protocol allows a streaming verifier to "timestamp" its message, providing evidence that it was chosen before certain information was streamed.

Theorem 12 (Theorem 6.6, informally stated)**.** *Let Γ be an alphabet and A a space-s streaming algorithm with $s = \operatorname{polylog} |\Gamma|$. If A streams $z \sim \Gamma^{v}$ and v is large enough, the following holds:* independently of its computation *after z, with high probability A can output at most s symbol-certificate pairs $(\alpha, i) \in \Gamma \times [v]$ such that $\alpha = z_i$.*

In other words, $A(z)$ cannot remember more than s symbol-certificate pairs for the string z; and the bound is unchanged if A obtains information uncorrelated with z after reading the stream.

1.3.2 Applications

Recall that Theorems 9 and 10 provide zero-knowledge versions of the general tools that essentially underlie all known SIPs, namely, the *sumcheck* and *polynomial evaluation* protocols. We demonstrate the power and flexibility of our tools by deriving from them explicit zkSIPs for streaming problems of fundamental importance: INDEX and FREQUENCY-MOMENT, as well as POINT-QUERY, RANGE-COUNT, SELECTION and INNER-PRODUCT.

As mentioned in the previous section, while the following statements highlight space complexities, the communication complexities are $n^{o(1)}$ in the interactive stage and $O(n^{1+\delta})$ for arbitrarily small δ in the setup stage.

In the INDEX problem, a streaming algorithm reads a length-n string x followed by an index $j \in [n]$, and its goal is to output x_j. INDEX is a hard problem for streaming algorithms, requiring *linear* space to solve [RY20]. By instantiating our zkSIP for polynomial evaluation with respect to the low-degree extension of the input evaluated at the index j, we obtain the following.

Corollary 1 (Corollary 6.1, informally stated)**.** *There exists a zkSIP for* INDEX *with logarithmic verifier space complexity.*

Note that this matches the complexity of the non-zero-knowledge SIP of [CCM+15].

In the FREQUENCY-MOMENT$_k$ (or F_k) problem, an algorithm streams $x \in [\ell]^n$ and its task is to compute $F_k(x) = \sum_{i \in [\ell]} \varphi_i^k$, the k^{th} moment of the frequency vector $(\varphi_1, \dots, \varphi_\ell)$, where φ_i is the number of occurrences of i in x. This is a central problem in the streaming literature, which is well known to require linear space to compute [AMS99]; by instantiating our sumcheck protocol with respect to the low-degree extension *of the frequency vector*, we obtain a zero knowledge protocol for the exact computation of F_k.

Corollary 2 (Corollary 6.5, informally stated)**.** *For every $\ell \in [n]$ and k, there exists a zkSIP that computes F_k with* $\mathrm{polylog}(n)$ *verifier space complexity.*

Lastly, we illustrate the flexibility of our protocols by constructing additional zkSIPs for several other problems: POINT-QUERY (where the input is a stream of integer updates to an ℓ-dimensional vector y followed by an index j and the task is to output y_j); RANGE-COUNT (where the input is a sequence of points in $[\ell]$ followed by a range $R \subseteq [\ell]$ and the task is to output the number of occurrences in R); SELECTION (which generalises the computation of the median); and INNER-PRODUCT (where the task is to output the inner product between the frequency vectors of a pair of streams).

Corollary 3 (Corollaries 6.2 to 6.4 and 6.6, informally stated)**.** *There exist* $\mathrm{polylog}(n)$*-space zkSIPs for* POINT-QUERY, RANGE-COUNT, SELECTION and INNER-PRODUCT.

1.3.3 Open problems

This work opens several avenues for future research; in this short section, we highlight four particularly compelling directions.

Achieving zero-knowledge versions of the main building blocks in the SIP literature suggests a natural question: can *all* SIPs be endowed with zero-knowledge? That is, denoting by SIP (respectively, zkSIP) the class of languages that admit SIPs (respectively, zkSIPs) with $\mathrm{polylog}(n)$ space complexity, we raise the following problem.

Open Problem 9. Is SIP equal to zkSIP?

In our two-stage protocols, the communication complexity is dominated by the setup (which consists of a reusable random string of near-linear length); the remainder of the protocol is extremely efficient, with $n^{o(1)}$ (or even $\mathrm{polylog}\, n$) communication and time complexity. Making this parameter sublinear would be a major step towards practical applicability.

Open Problem 10. Can zero-knowledge SIPs achieve sublinear communication complexity?

Lastly, recall that the notion of security in this work is (unconditional and) *computational*, where streaming adversaries detect a simulation with at most $o(1)$ bias. It is natural to ask whether stronger notions are achievable – both with respect to an adversary's capabilities and feasible security bounds – which raise the following questions. (For concreteness, one may consider SIPs for INDEX.)

Open Problem 11. Are there SIPs with statistical (or even perfect) zero-knowledge?

Open Problem 12. Can security bounds of $\frac{1}{\mathrm{poly}(n)}$ or $\frac{1}{n^{\omega(1)}}$ be obtained for computational zkSIPs?

Chapter 2

Notation and standard results

This chapter introduces common notation and definitions that are used throughout this thesis, along with standard results.

For an integer $\ell \geq 1$, we denote by $[\ell]$ the set $\{1, 2, \ldots, \ell\}$. Sets S such that $|S| = q$ are called q-sets. The complement of S is denoted $\overline{S}$. An arbitrary polynomial in ℓ is denoted $\mathrm{poly}(\ell)$, and $\mathrm{polylog}(n)$ denotes $\mathrm{poly}(\log n)$.

We use lowercase Latin letters to denote positive integers (e.g., d, i, j, n) or strings (e.g., x, y), with x generally denoting a binary string that is the input of a computational problem and n its length. Our notation for matrices is the same as for strings (lowercase Latin letters), and it will be clear from context which is the case; when x is a matrix, we use x_i to refer to the i^{th} row of x.

Lowercase Greek letters usually denote (possibly constant) functions in n with codomain $[0, 1]$ (e.g., ε, δ) or elements of a finite alphabet or field (e.g., α, β); in the latter case, ρ and σ are random elements. Uppercase letters usually denote either algorithms (e.g., D, T, V) or sets; additionally, T is used as the indeterminate of a polynomial.

As integrality issues do not substantially change any of our results, equality between an integer and an expression (that may not necessarily evaluate to one) is assumed to be rounded to the nearest integer.

Multi-sets of sets. To prevent ambiguity, we call (multi-)sets comprised of objects other than points (such as sets, trees or tuples) (multi-)*collections* in this work, and denote them by the calligraphic capitals $\mathcal{D}, \mathcal{S}, \mathcal{T}$.

Vectors and matrices. We use vectors or tuples, interchangeably, to refer to elements of a vector space over a finite field $\mathbb{F}$. Such tuples are denoted with boldface (e.g., $\boldsymbol{\alpha}, \boldsymbol{\beta}$) and random tuples are denoted $\boldsymbol{\rho}, \boldsymbol{\sigma}$. We use $\boldsymbol{\alpha} \cdot \boldsymbol{\beta}$ to denote the inner product between the two vectors, and, when the dimension of $\boldsymbol{\alpha}$ matches the number of rows of a matrix x, we use $\boldsymbol{\alpha} \cdot x$ to denote the vector corresponding to the linear combination of the rows of x with coefficients $\boldsymbol{\alpha}$, i.e., $\sum_i \boldsymbol{\alpha}_i x_i$. (Equivalently, we assume vectors to be in row form.)

Asymptotic notation. For two functions $f, g \colon \mathbb{N} \to \mathbb{N}$, we say $f = O(g)$ if there exist $c > 0$ and $n_0 \in \mathbb{N}$ such that $c \cdot g(n) \geq f(n)$ for all $n \in \mathbb{N} \setminus [n_0]$.[1]

Furthermore, $f = \tilde{O}(g)$ stands for $f = O(h)$ where $h = g \cdot \operatorname{polylog}(g)$, and $f = \tilde{\Omega}(g)$ for $f = \Omega(h)$ where $h = g / \operatorname{polylog}(g)$. Lastly, $f = o(g)$ means f/g vanishes asymptotically: $\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0$.

Additionally, we sometimes use $\alpha \in f$ as a shorthand for $\alpha \in \operatorname{Im} f$ and $f_{|g}$ for $f \circ g$.

Distance measures. The distance between strings (over an alphabet Γ that is often $\mathbb{F}_2 = \{0, 1\}$) is induced by normalised *Hamming weight*: for $x, y \in \mathbb{F}^n$, the (normalised) Hamming weight of x is $|x| := \frac{|\{i \in [n] : x_i \neq 0\}|}{n}$ and the distance between x and y is $\Delta(x, y) = \frac{|\{i \in [n] : x_i \neq y_i\}|}{n}$.

For unitary matrices U, V, unless otherwise stated, we consider the distance measure to be that induced by the normalised *Hilbert-Schmidt norm*: the norm of A is $\|A\| := \sqrt{\frac{1}{n} \operatorname{Tr}(A^\dagger A)} = \sqrt{\frac{1}{n} \sum_{\lambda \in \Lambda(A^\dagger A)}^{n} \lambda^2}$, where $\Lambda(AA^\dagger)$ is the set of eigenvalues of $AA^\dagger$; and the distance between U and V is $\Delta(U, V) = \|U - V\|$.

We say two objects X, Y are *ε-close* if $\Delta(X, Y) \leq \varepsilon$, and otherwise they are *ε-far*. The ball of radius ε around X is $B_\varepsilon(X) = \{Y : Y \text{ is } \varepsilon\text{-close to } X\}$, and the ball $B_\varepsilon(S)$ around a set S is the union over $B_\varepsilon(X)$ for each $X \in S$. The notion of distance with respect to sets extends naturally: X is ε-close to a set S if $\Delta(X, Y) \leq \varepsilon$ for some $Y \in S$, and it is ε-far from S otherwise.

2.1 Probability and concentration

We use $X \sim \mu$ to denote a random variable with distribution μ, and write $X \sim S$ for the uniform distribution over a set S. All probability spaces we consider in this thesis are discrete (and finite); we denote events with square brackets, e.g., $E = [X > 0]$, with the probability of the event E denoted $\mathbb{P}[E] = \mathbb{P}[X > 0]$ and the expected value of X denoted $\mathbb{E}[X]$. We sometimes make the sources of randomness in a probabilistic expression explicit, and when we do they are assumed to be independent; e.g., only when X and Y are independent do we write $\mathbb{P}_{X \sim \mu, Y \sim \lambda}[E]$.

[1]Denoting $f \in O(g)$ would be more accurate, but we follow the widely adopted abuse of notation.

Concentration inequalities. We will make thorough use of the following versions of the Chernoff and Hoeffding inequalities (stated, e.g., in [MU05]).

Lemma 2.1 (Additive Chernoff-Hoeffding bound)**.** *Let $X_1, \ldots, X_k$ be independent Bernoulli random variables distributed as X. Then, for every $\delta \in [0,1]$,*

$$\mathbb{P}\left[\frac{1}{k}\sum_{i=1}^{k} X_i \leq \mathbb{E}[X] - \delta\right] \leq e^{-2\delta^2 k} \text{ and}$$
$$\mathbb{P}\left[\frac{1}{k}\sum_{i=1}^{k} X_i \geq \mathbb{E}[X] + \delta\right] \leq e^{-2\delta^2 k}.$$

Lemma 2.2 (Multiplicative Chernoff-Hoeffding bound)**.** *Let $X_1, \ldots, X_k$ be independent Bernoulli random variables distributed as X. Then, for every $\delta \in [0,1]$,*

$$\mathbb{P}\left[\frac{1}{k}\sum_{i=1}^{k} X_i \geq (1+\delta)\mathbb{E}[X]\right] \leq e^{-\frac{\delta^2 k \mathbb{E}[X]}{3}} \text{ and}$$
$$\mathbb{P}\left[\frac{1}{k}\sum_{i=1}^{k} X_i \leq (1-\delta)\mathbb{E}[X]\right] \leq e^{-\frac{\delta^2 k \mathbb{E}[X]}{2}}.$$

Lemma 2.3 (Hoeffding's inequality)**.** *Let $X_1, \ldots, X_k$ be independent random variables distributed as $X \in [a,b]$. Then, for every $\delta \in [0,1]$,*

$$\mathbb{P}\left[\frac{1}{k}\sum_{i=1}^{k} X_i \leq (1-\delta)\mathbb{E}[X]\right] \leq e^{-\left(\frac{\delta\mathbb{E}[X]}{b-a}\right)^2 k} \text{ and}$$
$$\mathbb{P}\left[\frac{1}{k}\sum_{i=1}^{k} X_i \geq (1+\delta)\mathbb{E}[X]\right] \leq e^{-\left(\frac{\delta\mathbb{E}[X]}{b-a}\right)^2 k}.$$

2.2 Quantum information and computation

We use calligraphics (e.g., $\mathcal{H}$) to denote arbitrary finite-dimensional Hilbert spaces. $\mathcal{U}(\mathcal{H})$ denotes the set of unitary operators on $\mathcal{H}$; the conjugate transpose of $U \in \mathcal{U}$ is denoted $U^\dagger$ and satisfies $UU^\dagger = \mathbf{I}$, the identity operator of $\mathcal{H}$.

A pure state is a unit vector in the Hilbert space $\mathcal{H}$, and is usually represented in Dirac notation, e.g., $|\psi\rangle$. We write $\dim(\mathcal{H})$ to denote the dimension of $\mathcal{H}$, and its *computational* (or canonical) basis is denoted $\{|i\rangle : i \in [\dim \mathcal{H}]\}$; when $\mathcal{H} = (\mathbb{C}^2)^{\otimes k}$, the computational basis can equivalently be labeled by $\{|x\rangle : x \in \{0,1\}^n\}$. (The states $|i\rangle$ and $|x_1\rangle \otimes \cdots \otimes |x_k\rangle$, where x is the binary representation of i, coincide.) For brevity, we also represent $|0\rangle^{\otimes n}$ as $|\mathbf{0}\rangle$.

A mixed state is a distribution on pure states $\{p_k, |\psi_k\rangle\}$; it is represented as a *density matrix* $\rho = \sum p_k |\psi_k\rangle\langle\psi_k| \in \mathbf{S}(\mathcal{H})$, where $\mathbf{S}(\mathcal{H})$ is the set of positive semi-definite operators with unit trace. Typically we divide a Hilbert space into *registers*, e.g. $\mathcal{H} = \mathcal{H}_1 \otimes \mathcal{H}_2$, and we sometimes write $\mathcal{H} \setminus \mathcal{H}_2$ to denote $\mathcal{H}_1$; we also write $\rho^{\mathcal{H}_1}$ to specify that $\rho \in \mathbf{S}(\mathcal{H}_1)$.

A *projector* Π is a Hermitian operator ($\Pi^\dagger = \Pi$) such that $\Pi^2 = \Pi$. If a (unitary U or) projector Π in a Hilbert space $\mathcal{H}_1 \otimes \mathcal{H}_2$ acts trivially (as the identity $\mathbf{I}$) in $\mathcal{H}_2$, we may write Π or $\Pi^{\mathcal{H}_1}$ to denote $\Pi \otimes \mathbf{I}^{\mathcal{H}_2}$. A collection of projectors $\mathsf{M} = (\Pi_i)_{i \in S}$ is a *projective measurement* when $\sum_{i \in S} \Pi_i = \mathbf{I}$, and a *submeasurement* when there exists a projector Π such that $\sum_{i \in S} \Pi_i = \mathbf{I} - \Pi$.

The application of M to a pure state $|\psi\rangle$ yields outcome $i \in S$ with probability $p_i = \| \Pi_i |\psi\rangle \|^2$; we denote sampling from this distribution by $i \leftarrow \mathsf{M}(\rho)$, and in this case the post-measurement state is $|\psi_i\rangle = \Pi_i |\psi\rangle / \sqrt{p_i}$. We also use $\sigma \leftarrow \mathsf{M}(\rho)$ to denote the mixture of post-measurement states $\Pi_i |\psi\rangle / \sqrt{p_i}$ with probability p_i. A two-outcome projective measurement is called a *binary projective measurement*, and is written as $\mathsf{M} = (\Pi, \mathbf{I} - \Pi)$, where Π is associated with the outcome 1, and $\mathbf{I} - \Pi$ with the outcome 0.

The set of linear operators mapping $\mathcal{H}$ to $\mathcal{K}$ is denoted by $\mathcal{L}(\mathcal{H}, \mathcal{K})$; the shorthand $\mathcal{L}(\mathcal{H})$ stands for $\mathcal{L}(\mathcal{H}, \mathcal{H})$. The set $T(\mathcal{H}, \mathcal{K})$ consists of the linear mappings from $\mathcal{L}(\mathcal{H})$ to $\mathcal{L}(\mathcal{K})$.

We say T is a completely positive (CP) map if $T \otimes \mathbf{I}_{\mathcal{K}}$ is positive for all $\mathcal{H}$, where $\mathbf{I}_{\mathcal{K}}$ is the identity of the Hilbert space $\mathcal{K}$. Furthermore, T is a completely positive trace preserving (CPTP) map if T is CP and trace preserving, i.e., such that $\mathrm{Tr}(T(\rho)) = \mathrm{Tr}(\rho)$ for all ρ. For every CPTP map $T \colon \mathbf{S}(\mathcal{H}) \to \mathbf{S}(\mathcal{H})$ there exists a *unitary dilation* U that operates on an expanded Hilbert space $\mathcal{H} \otimes \mathcal{K}$, so that, with $\mathrm{Tr}_{\mathcal{K}}$ the partial trace operator that traces out $\mathcal{K}$, we have $T(\rho) = \mathrm{Tr}_{\mathcal{K}}(U(\rho \otimes |0\rangle\langle 0|^{\mathcal{K}})U^\dagger)$. As a unitary dilation of a circuit T can be written as a circuit of size $O(|T|)$, we may assume without loss of generality that quantum computations that output a (classical) string consist of applying unitaries, performing a measurement and outputting the outcome.

For further details on quantum information and computation, see [NC16].

2.3 Algorithms and protocols

We use the same term to refer to computational problems and to protocols that solve them, but distinguish the two cases with different fonts (so that the pep and $\mathsf{sumcheck}$ protocols solve the PEP and SUMCHECK problems, respectively). We distinguish complexity classes and the algorithms they contain similarly: e.g., P, NP and BQP are complexity classes whose problems are solvable by P, NP and BQP algorithms, respectively, while IP is a complexity class characterised by IP protocols.

Query algorithms. We denote by $M^x(z)$ the output of algorithm M given direct access to input z and query access to string x (see Section 2.3.1). Probabilistic expressions that involve a randomised algorithm M generally omit the inner randomness of M; e.g., $\mathbb{P}[A(X) = 0]$ (if the distribution of X is known from context) or $\mathbb{P}_{X\sim\mu}[A(X) = 0]$ are shorthand for $\mathbb{P}_{X\sim\mu,r\sim\{0,1\}^m}[A(X;r) = 0]$, where r is A's internal randomness. The number of coin tosses A makes is its *randomness complexity*.

The minimax principle. We shall often make use of the *minimax principle*, and assume, without loss of generality, that a computationally unbounded algorithm A whose goal is to maximise some value $\mathbb{E}_{x\sim\mu}[f(A(x))]$ (e.g., the probability that $A(x)$ equals x) can be assumed to be deterministic, and thus given by a function $x \mapsto a(x)$; equivalently, A can be taken as the deterministic algorithm that maximises $\mathbb{E}[f \circ a(x)]$ for the distribution of inputs μ.

Streaming. When A is a *streaming algorithm*, x is read sequentially in one pass, from the first symbol (x_1) to the last. When $A(x, y, z)$ reads multiple inputs, $A(y)$ denotes the partial execution of A after it has read x. When the entries of a length-n string x are taken over a finite alphabet Γ, we may also use x for the equivalent bit string of length $n \log |\Gamma|$.

The *snapshot* of an algorithm is synonymous with its memory state; when A reads a sequence of more than one input, e.g., $A(x, y)$, the "snapshot of A after x" is the snapshot immediately before the first symbol of y is streamed (i.e., after A has read and processed the last symbol of x). When A is interacting in a protocol and sends a message between reading x and y, the snapshot after x is that immediately before sending the message.

Protocols. We generally use P to denote an algorithm with unbounded computational resources. In a protocol, two algorithms P and V interact by exchanging messages in a predefined order; after all messages have been exchanged, V chooses an output that we denote $\langle P, V\rangle$ and call the output of the protocol. When V rejects or P aborts midway through the interaction, we assume the algorithm proceeds until the end of the protocol with dummy messages (e.g., strings of zeroes).

2.3.1 Local algorithms and the query model

Local (or sublinear-query) algorithms are an extremely important class of sublinear algorithms, our main object of study. Such algorithms are defined in the *query model*, where (in the classical case) the input x is given as an oracle that returns x_i when input the coordinate i. (We describe the quantum case in the end of this subsection.)

The maximal number of queries M performs over all strings x and outcomes of its coin tosses is interchangeably referred to as its *query complexity* or *locality* q. When $q = o(n)$, where n is the length of the string x, we say M is a (q-)*local* algorithm. If the queries performed by M are determined in advance (so that no query depends on the result of any other query), M is *non-adaptive*; otherwise, it is *adaptive*. Finally, if M queries each coordinate independently with some probability p, we say it is a *sample-based* algorithm. Since we will want to have an absolute bound on the sample complexity (i.e., the number of coordinates sampled) of sample-based algorithms, we allow them to cap the number of coordinates they sample.

Adaptivity. Adaptive local algorithms are characterised in two equivalent manners: a standard description via decision trees and an alternative that makes more direct use of set systems. Let M be a q-local algorithm for a decision problem (i.e., which outputs an element of $\{0,1\}$) with oracle access to a string over alphabet Γ and no access to additional input (what follows immediately generalises by enumerating over explicit inputs).

The behaviour of M is characterised by a collection $\{(T_s, s) : s \in \{0,1\}^r\}$ of *decision trees*, where r is the randomness complexity of M; the trees are $|\Gamma|$-ary, have depth q, edges labeled by elements of Γ, inner nodes labeled by elements of $[n]$ and leaves labeled by elements of $\{0,1\}$. The execution of M^x proceeds as follows. It (1) flips r random coins, obtains a string $s \in \{0,1\}^r$ and chooses the decision tree T_s to execute; (2) beginning at the root, for q steps, queries the coordinate given by the label at the current vertex and follows the edge whose label is the queried value; and (3) outputs the bit given by the label of the leaf thus reached.

Quantum query algorithms. A quantum algorithm V with query access to a bit string $x \in \{0,1\}^n$ is specified by a sequence of unitary operations $V_0 \dots V_q$, that do not depend on the input x. A query to x is given by the unitary U_x on $\log n + 1$ qubits such that

$$U_x \left|i\right\rangle \left|b\right\rangle = \left|i\right\rangle \left|b \oplus x_i\right\rangle \ .$$

We denote by V^U the output of V with oracle access to a unitary U. Similarly, $V^U(n)$ denotes the case where V has access to an additional explicit input n.

The final state of an algorithm that makes q queries to the oracle, before measurement, is given by

$$V_q(U_f \otimes I)V_{q-1}(U_f \otimes I) \dots V_1(U_f \otimes I)V_0 \left|\mathbf{0}\right\rangle \ .$$

The overall Hilbert space $\mathcal{H}$ used by the algorithm is split into three registers $\mathcal{X} \otimes \mathcal{W} \otimes \mathcal{B}$, and the verifier's memory is initialised to the state $|\mathbf{0}\rangle$. The oracle acts on $\mathcal{X}$ and one additional qubit, the workspace $\mathcal{W}$ can have arbitrary size and $\mathcal{B}$ represents the single-qubit output of the algorithm. The final step of the algorithm is to measure the $\mathcal{B}$ register in the computational basis and return the outcome.

For an introduction to (classical and) quantum query complexity, see [BW02]; and for a recent summary of the progress in the last two decades, see [Aar21].

2.4 Property testing

A *property tester* is a local algorithm that solves the approximate decision problem of membership in a set Π. (For a thorough introduction to property testing, see the book [Gol17].) We choose to define property testers in the most general form we consider, namely, as parties in interactive protocols; then more restricted variants (including standard testers) follow as simple restrictions of the definition.

An *interactive proof of proximity* (IPP) for Π is a proof system that solves the problem of testing Π. The *verifier* algorithm receives as input a proximity parameter ε and has oracle access to x. It queries x in at most q coordinates and interacts with an all-powerful but untrusted prover by exchanging m messages, where the total number of communicated bits is c. The verifier must accept when $x \in \Pi$ and reject when x is ϵ-far from Π, with bounded probability of error; the outcome of such an interaction is denoted $\langle P(x), V^x\rangle$. Formally,

Definition 2.1. *An* interactive proof of proximity *(IPP) for a property* $\Pi = \cup_n \Pi_n$ *is an interactive protocol with two parties: a (computationally unbounded) prover* P *and a verifier* V*, which is a probabilistic algorithm. The parties send messages to each other in turns, with the first sent from prover to verifier, and at the end of the communication, the following two conditions are satisfied:*

1. Completeness: *For every* $\varepsilon > 0$, $n \in \mathbb{N}$, *and* $x \in \Pi_n$, *there exists a prover* P *such that*

$$\mathbb{P}\left[\langle P(x), V^x\rangle(n, \varepsilon) = 1\right] \geq 2/3 \ ,$$

where the probability is over the coin tosses of V.

2. Soundness: *For every $\varepsilon > 0$, $n \in \mathbb{N}$, $x \in \{0,1\}^n$ that is ε-far from Π_n and for every computationally unbounded (cheating) prover $\widetilde{P}$ it holds that*

$$\mathbb{P}\left[\langle \widetilde{P}(x), V^x \rangle(n, \varepsilon) = 1\right] \leq 1/3 ,$$

where the probability is over the coin tosses of V.

The query complexity *q of the protocol is the maximum number of queries the verifier makes to x in its execution; the* message complexity *m is the number of messages exchanged between prover and verifier; and the* communication complexity *c is the total number of bits communicated by these messages.*

The set of properties Π for which there exists an IPP protocol with proximity parameter ε with m messages, communication complexity c and query complexity q is denoted $\mathsf{IPP}(\varepsilon, c, q, m)$.

When the completeness condition holds with probability 1, i.e., the verifier always accepts when $x \in \Pi$, we call the protocol *one-sided.* Moreover, if the verifier does not receive ε explicitly, but rejects inputs that are ε-far from Π with *detection probability* $\rho(n, \varepsilon)$, where $\rho \colon \mathbb{N} \times [0,1] \to [0,1]$ is monotonically nondecreasing in ε, the protocol is said to be *proximity-oblivious.*

A *Merlin-Arthur proof of proximity* (MAP) for Π is an IPP where the entire communication is a single message from the prover to the verifier (i.e., an IPP with message complexity 1); the class $\mathsf{MAP}(\varepsilon, p, q)$ is thus defined as $\mathsf{IPP}(\varepsilon, p, q, 1)$. The formal definition of the *quantum* generalisation of MAPs is given in Chapter 5 (and the corresponding complexity class in Definition 5.1).

A *property tester* is an IPP with $c = m = 0$, i.e., where no communication occurs. In this case, the verifier is called a *tester*, and we define $\mathsf{PT}(\varepsilon, q) \coloneqq \mathsf{IPP}(\varepsilon, 0, q, 0)$.

2.5 Discrete algebra and coding theory

A *code* $C \colon \mathbb{F}^k \to \mathbb{F}^n$ (where $\mathbb{F}$ is a finite field) is an injective mapping from *messages* of length k to codewords of *blocklength* n. The *rate* of the code C is k/n and its *distance* is the minimum, over all distinct messages $x, y \in \mathbb{F}^k$, of $\Delta(C(x), C(y))$. We shall sometimes slightly abuse notation and use C to denote the set of all of its codewords $\left\{C(x) : x \in \mathbb{F}^k\right\} \subset \mathbb{F}^n$. If the mapping C is linear, we say C is a linear code.

Low-degree extensions. For any field $\mathbb{F}$ and integer k such that $|\mathbb{F}| \geq k$, we consider $[k] \subseteq \mathbb{F}$ via a canonical injection (e.g., taking the image of $\ell \in [k]$ as the field element whose binary representation is the same as that of ℓ). Accordingly, we write $\ell \in \mathbb{F}$ as shorthand for the field element corresponding to the image of $\ell \in [k]$ via this canonical injection.

For a string $y \in \mathbb{F}^k$, the *low-degree extension* (LDE) with *degree* d and *dimension* m where $|\mathbb{F}| \geq d+1$ and $k \leq (d+1)^m$, denoted $\hat{y}$, is the unique m-variate polynomial of individual degree d that coincides with y in $[k]$; more precisely, viewing $[k] \subseteq [d+1]^m \subseteq \mathbb{F}^m$, the LDE $\hat{y} : \mathbb{F}^m \to \mathbb{F}$ is the unique polynomial satisfying $\hat{y}(i) = y_i$ for all $i \in [k]$.[2] Our notation for the polynomial $\hat{y}$ omits the degree and dimension, as they will be clear from context.

When y is a matrix, we use $\hat{y}(\boldsymbol{\alpha}, \boldsymbol{\beta})$ to denote the linear combination of the LDEs of the rows with linear coefficients $\boldsymbol{\beta}$, i.e., $\hat{y}(\boldsymbol{\alpha}, \boldsymbol{\beta}) = \sum_i \boldsymbol{\beta}_i \hat{y}_i(\boldsymbol{\alpha})$.

2.6 Information theory

We will make use of several notions of information theory and approximations of information-theoretic quantities. The *q-ary entropy function* is defined as

$$\begin{aligned} H_q(t) &= t \log_q(q-1) - t \log_q t - (1-t) \log_q(1-t) \\ &= \frac{1}{\log q}\big(t \log(q-1) - t \log t - (1-t)\log(1-t)\big) \\ &= \frac{1}{\log q}\big(t \log(q-1) + H_2(t)\big), \end{aligned} \tag{2.1}$$

where $H_q(0) = 0$; we also use the shorthand H for H_2, which simplifies to

$$H(t) = H(1-t) = -t \log t - (1-t) \log(1-t). \tag{2.2}$$

We will make use of the following approximation for the (natural) logarithm function: for $0 \leq t \leq 1/2$,

$$-t(1+t) \leq \ln(1-t) \leq -t. \tag{2.3}$$

Recall that a *Hamming ball* of radius δ around the point $b \in \Gamma^k$ is defined as $B(b, \delta) \coloneqq \{a \in \Gamma^k : d(a,b) \leq \delta\}$. Setting $\gamma \coloneqq |\Gamma|$ for ease of notation, we will use the following approximation to the ball's volume: defining $\varepsilon = 1 - \delta$, when k is large

[2] If $(d+1)^m > k$, uniqueness is ensured by padding y with zeroes to obtain y' of length $(d+1)^m$, and embedding the latter.

enough and $\varepsilon = k^{-1}\operatorname{polylog}(k)$, then

$$\gamma^{H_\gamma(\delta)k} \geq |B(b,\delta)| = \Omega\left(\frac{\gamma^{H_\gamma(\delta)k}}{\sqrt{\varepsilon k}}\right) = \frac{\gamma^{H_\gamma(\delta)k}}{\operatorname{polylog}(k)}\ .[3] \tag{2.4}$$

The entropy of a discrete random variable X taking values in Γ is

$$H(X) = -\sum_{\alpha\in\Gamma}\mathbb{P}[X=\alpha]\log\big(\mathbb{P}[X=\alpha]\big).$$

Every such random variable satisfies

$$H(X) \in [0, \log|\Gamma|]. \tag{2.5}$$

The conditional entropy $H(X|Y)$ is the entropy of the conditional random variable, which satisfies

$$H(X|Y) \leq H(X). \tag{2.6}$$

If X, Y are independent, then

$$H(X,Y) = H(X) + H(Y). \tag{2.7}$$

The last property of entropy we will make use is the *chain rule*: for random variables $X_1, \ldots, X_n$,

$$H(X_1,\ldots,X_n) = \sum_{i=1}^{n} H(X_i|X_1,\ldots,X_{i-1}). \tag{2.8}$$

For ease of notation, when (X,Y) are jointly distributed over Γ^2 with marginals μ and λ, respectively, we denote the distribution of Y conditioned on $X=x$ as λ_x. The *KL divergence* between the distributions is

$$\mathrm{KL}(\mu\ ||\ \lambda) = \sum_{\alpha\in\Gamma}\mu(x)\log\frac{\mu(\alpha)}{\lambda(\alpha)}. \tag{2.9}$$

KL divergence upper bounds the Euclidean distance between probability

[3]The lower bound is a simplification of

$$|B(b,\delta)| \geq \gamma^{H_\gamma(\delta)k}\cdot\frac{\exp\left(\frac{1}{12k+1}-\frac{1}{12\delta k}-\frac{1}{12\varepsilon k}\right)}{\sqrt{2\pi\delta(1-\delta)k}};$$

since $\frac{1}{\varepsilon k} = o(1)$, the numerator is $1-o(1)$, and the denominator is of order $\Theta(\sqrt{\varepsilon k}) = \operatorname{polylog}(k)$. (See, e.g., [GRS12].)

vectors via *Pinsker's inequality* as follows (see, e.g., [BLM13]):

$$\|\mu - \lambda\|^2 \leq \frac{\mathrm{KL}\big(\mu \mid\mid \lambda\big)}{2 \ln 2}. \tag{2.10}$$

Finally, the *mutual information* is defined as (and equivalent to)

$$\begin{aligned} I(\mu : \lambda) &\coloneqq I(X : Y) \qquad (2.11) \\ &= I(Y : X) \\ &= H(Y) - H(Y|X) \\ &= \mathbb{E}_{X \sim \mu}[\mathrm{KL}(\lambda_X \mid\mid \lambda)]. \end{aligned}$$

Chapter 3

Technical overview

In this chapter, we provide high-level overviews of the techniques used to prove the theorems in Chapters 4 to 6. These are given in order: Section 3.1 discusses our structural theorem for local algorithms proved in Chapter 4; Section 3.2 gives an overview of quantum proofs of proximity, the object of study of Chapter 5; and lastly, Section 3.3 discusses streaming zero-knowledge, the topic of Chapter 6.

3.1 Lower bounds for local algorithms

We now proceed to an outline of the techniques used and developed in Chapter 4, where we prove Theorem 1 and its applications. Our techniques build on and simplify ideas from [FLV15, GL21], but are significantly more general and technically involved, and in particular, offer novel insight regarding adaptivity in local algorithms.

Our starting point, which we outline in Section 3.1.1, generalises the techniques of [GL21] (which are, in turn, inspired by [FLV15]) to the setting of robust local algorithms. Then, in Section 3.1.2, we identify a key technical bottleneck in previous works: *adaptivity*. We discuss the fundamental challenges that adaptivity imposes, and in Section 3.1.3 we present our strategy for meeting these challenges and the tools that we develop for dealing with them, as well as describe our construction. Subsequently, in Section 3.1.5, we provide an outline of the analysis of our construction, which relies on the Hajnal–Szemerédi theorem to sample from structured set systems we call *daisies*.

The setting. Recall, from Section 1.1.2, that our goal is to transform a robust (query-based) local algorithm into a sample-based algorithm with sublinear sample complexity. Towards this end, let M be a (ρ_0, ρ_1)-robust local algorithm for computing a function $f\colon \{0,1\}^n \to \{0,1\}$.[1] Since we also need to deal with *one-sided robustness*, assume without loss of generality that $\rho_1 = 0$ and $\rho \coloneqq \rho_0 = \Omega(1)$. Recall that the algorithm M receives query access to a string $x \in \{0,1\}^n$, flips at most r random coins, makes at most q queries to this string and outputs $f(x) \in \{0,1\}$ with probability at least $1-\sigma$.

For simplicity of exposition, we assume that the error rate is $\sigma = \Theta(1/q)$, the query complexity is constant ($q = O(1)$), and the randomness complexity r is bounded by $\log(n) + O(1)$. We remark that the analysis trivially extends to non-constant values of q, and that we can achieve the other assumptions via simple transformations, which we provide in Section 4.3.4, at the cost of logarithmic factors in q. In the following, our aim is to construct a *sample-based* local algorithm N for computing the function f, with sample complexity $O\big(n^{1-1/2q^2}\big) = n^{1-1/O(q^2)}$.

3.1.1 The relaxed sunflowers method

As a warm-up, we first suppose that the algorithm M is *non-adaptive*. (This section gives an overview of the techniques of [GL21], which suffice in the non-adaptive case.) Then we can simply represent M as a distribution μ over a collection of query sets $\mathcal{S}$, where each $S \in \mathcal{S}$ is a subset of $[n]$ of size q, and predicates $\{f_S\colon \{0,1\}^q \to \{0,1\}\}_{S \in \mathcal{S}}$, as follows. The algorithm M draws a set $S \in \mathcal{S}$ according to μ, queries S, obtains the *local view* $x_{|S}$ (i.e., x restricted to the coordinates in S), and outputs $f_S(x_{|S})$.

Consider an algorithm N that samples each coordinate of the string x independently with probability $p = 1/n^{1/2q^2}$ (and aborts in the rare event that this exceeds the desired sample complexity).[2] Naively, we would have liked N^x to emulate an invocation of the algorithm M by sampling the restriction of x to a query set $S \sim \mu$.

Indeed, if the distribution μ is "well spread", the probability of obtaining such a local view of M is high. Suppose, for instance, that all of the query sets are pairwise disjoint. In this case, the probability of N sampling any particular local view is p^q, and we expect N to obtain $\Omega(p^q n) = \Omega(n^{1-1/2q})$ local views (recall that the support size of μ, i.e., the number of query sets, is $O(n)$ by our assumption of $\log n + O(1)$ randomness complexity). However, if μ is concentrated on a small number of coordinates, it is highly unlikely that N will obtain a local view of M. For example, if M queries the first coordinate of x with probability 1, then we can obtain a local view of M with probability at most p, which is negligible.

[1]In general, the function f may depend on an explicitly given parameter (e.g., an index for decoding in the case of relaxed LDCs), but for simplicity of notation, we omit this parameter in the technical overview.

[2]This choice of p will be made clear in Section 3.1.5; see Footnote 10.

Fortunately, we can capitalise on the robustness condition to deal with this problem. We first illustrate how to do so for an easy special case, and then deal with the general setting.

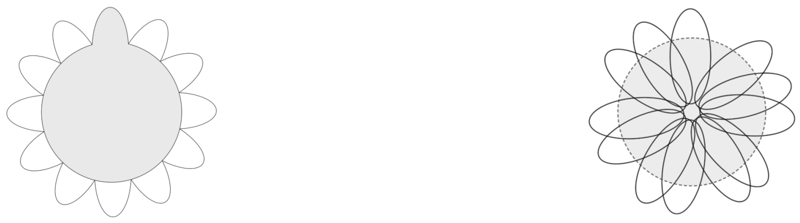
Special case: sunflower query set. Suppose that μ is concentrated on a small coordinate set K and is otherwise disjoint, i.e., the support of μ is a *sunflower* with kernel K of size at most ρn; see Figure 3.1a. Since the query sets are disjoint outside of K, by the discussion above we will sample many sets except for the coordinates in K (i.e., sample the petals of the sunflower). Recall that if x is such that $f(x) = 0$, then the $(\rho, 0)$-robust algorithm M outputs 0, with high probability, on any input y that is ρ-close to x. Thus, even if we arbitrarily assign values to K and use them to complete sampled petals into full local views, we can emulate an invocation of M that will output as it would on x.

If *all inputs* in the promise of M were robust (as is the case for LDCs, but *not* for testers, relaxed LDCs,[3] and PCPPs), then the above would suffice. However, recall that we are not ensured robustness when x is such that $f(x) = 1$. To deal with that, we can enumerate over all possible assignments to the kernel K, considering the local views obtained by completing sampled petals into full local views by using each kernel assignment to fill in the values that were not sampled. Observe that: (1) when the input x is a 1-input and N considers the kernel assignment that coincides with x, a majority of local views (a fraction of at least $1 - \sigma$) will lead M^x to output 1; and (2), when x is a 0-input, a minority of local views (a fraction of at most σ) will lead M^x to output 1 *under any kernel assignment.*

The sample-based algorithm N thus outputs 1 if and only if it sees, for some kernel assignment, a majority of local views that lead M to output 1. Recall that there is asymmetry in the robustness of M (while 0-inputs are robust, 1-inputs are not), which translates into asymmetric output conditions for N. Note, also, that correctness of this procedure for 0-inputs requires that not even a single kernel assignment would lead N to output incorrectly; but our assumption on the error rate ensures that the probability of sampling a majority of petals whose local views will lead to an error is sufficiently small to tolerate a union bound over all kernel assignments, as long as $|K|$ is small enough.

General case: extracting a heavy daisy from the query sets. Of course, the combinatorial structure of the query sets of a local algorithm is not necessarily

[3]The type of robustness that relaxed LDCs admit is slightly more subtle, since it deals with a larger alphabet that allows for outputting $\perp$. See discussion in Section 4.5.2.



(a) A *sunflower* with one of its sets shaded. The intersection of any two sets results in the same set, the kernel.

(b) A *daisy* with its kernel shaded, whose boundary is the dashed line. Outside the kernel each point is covered by a bounded number of petals.

Figure 3.1: Sunflowers and daisies.

a sunflower and may involve many complex intersections. While we could use the *sunflower lemma* to extract a sunflower from the collection of query sets, we stress that the size of such a sunflower is *sublinear*, which is not enough in our setting (as we deal with constant error rate).

Nevertheless, we can exploit the robustness of M even if its query sets only have the structure of a relaxed sunflower, referred to as a *daisy*, with a small kernel. Loosely speaking, a t-daisy is a sunflower in which the kernel is not necessarily the intersection of all petals, but is rather a small subset such that every element outside the kernel is contained in at most t petals;[4] see Figure 3.1b (and see Section 4.3.1 for a precise definition).

Using a *daisy lemma* [FLV15, GL21], we can extract from the query sets (the support of μ) of the robust local algorithm M a t-daisy $\mathcal{D}$ with t roughly equal to $n^{i/q}$ and a kernel K of size roughly $n^{1-i/q}$, where $i \in [q]$ bounds the size of the petals of $\mathcal{D}$. Moreover, the *weight* $\mu(\mathcal{D}) = \sum_{S \in \mathcal{D}} \mu(S)$ is significantly larger than the error rate σ of M (recall that we assumed a sufficiently small $\sigma = \Theta(1/q)$). Thus, even if the daisy contains all local views that lead to an error, their total weight would still be small with respect to that of local views leading to a correct decision; hence, the query sets in the daisy $\mathcal{D}$ well-approximate the behaviour of M, and we can disregard the sets in the support of μ that do not belong to $\mathcal{D}$ at the cost of a negligible increase to the error rate.

Crucially, the intersection bound t implies that *sampling a daisy is similar to sampling a sunflower*: since petals do not intersect heavily, with high probability many of them are fully queried (as is the case with sunflowers). The bound on $|K|$, on the other hand, allows us to implement the sampling-based algorithm we discussed

[4]In Definition 4.11, a t-daisy has t as a function from $[q]$ to $\mathbb{N}$ and allows for a tighter bound on the number of intersecting petals. We use the simplified definition of [GL21] in this technical overview.

for the sunflower case, except with respect to a daisy. The kernel is sufficiently small so that the output of M is unchanged under any assignment to K, and suffices to tolerate a union bound when considering all possible assignments to K.

It follows that the daisy $\mathcal{D}$ provides enough "sunflower-like" structure for the sample-based algorithm N defined previously to succeed, with high probability, when it only considers the query sets in $\mathcal{D}$ and enumerates over all assignments to its kernel.

3.1.2 The challenge of adaptivity

Let us now attempt to apply the transformation laid out in the previous section to a robust local algorithm M that makes q *adaptive* queries. In this case, M may choose to query distinct coordinates depending on the answers to its previous queries, and thus *there is no single distribution* μ that captures its query behaviour.

Observe that now, rather than inducing a distribution on sets, the algorithm M induces a distribution over *decision trees* of depth q, as the behaviour of a randomised query-based algorithm M^x can be described by choosing a decision tree according to its random string, then performing the adaptive queries according to the evaluation of that tree on the input $x \in \{0,1\}^n$. By our assumption on the randomness complexity of M, this distribution is supported on $\Theta(n)$ decision trees. Note that for any *fixed* input x, the decision tree collapses to a path, and hence the distribution over decision trees induces a distribution over query sets, which we denote μ_x (see Figure 3.2).

A naive way of transitioning from decision trees to sets is by querying all of the branches of each decision tree. Alas, doing so would increase the query complexity of M exponentially from q to (more than) 2^q, which would in turn lead to a sample-based algorithm with a much larger sample complexity than necessary. Thus, we need to deal with the far more involved structure induced by distributions over decision trees, which imposes significant technical challenges. For starters, since our technical framework inherently relies on a combinatorial characterisation of algorithms, we first need to find a method of transitioning from decision trees to (multi-)sets without increasing the query complexity of the local algorithm M.

To this end, a key idea is to enumerate over all random strings and their corresponding decision trees, and extract all q-sets (i.e., sets of size q) corresponding to each branch of each tree. This leaves us with a combinatorial *multi-set* $\mathcal{S}$ (as multiple random strings may lead to the same decision tree, and branches of distinct decision trees may query the same set) with $\Theta(2^q \cdot n) = \Theta(n)$ query sets, of size q each, corresponding to all possible query sets induced by all possible input strings.[5]

[5]We remark that this treatment of multi-sets allows us to significantly simplify the preparation

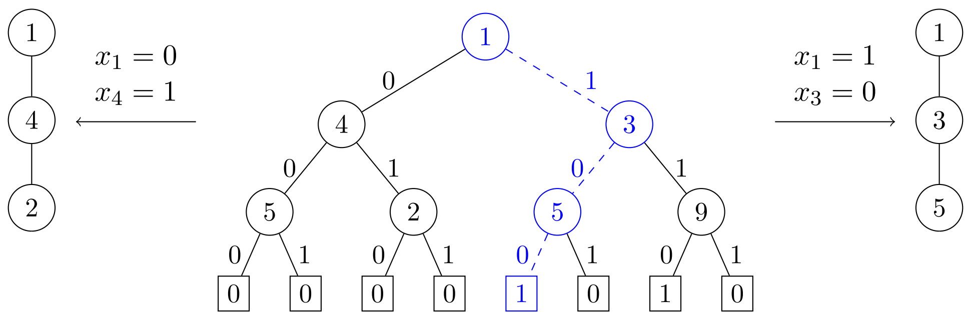


Figure 3.2: Decision tree of a 3-local algorithm. When the input x is such that $x_1 = 1$, $x_3 = 0$ and $x_5 = 0$, the branch highlighted in blue (and dashed) queries $\{1, 3, 5\}$ and outputs 1. When $x_1 = 0$ and $x_4 = 1$, this tree *induces* the query set $\{1, 2, 4\}$; when $x_1 = 1$ and $x_3 = 0$, it induces the set $\{1, 3, 5\}$. This "collapsing" of the query behaviour is illustrated on either side of the tree.

Note that $\mathcal{S}$ contains the elements of the support of μ_x for all inputs $x \in \{0, 1\}^n$ and that, for any fixed input x, the vast majority of these query sets may not be relevant to this input: each $S \in \mathcal{S} \setminus \text{supp}(\mu_x)$ corresponds to a branch of a decision tree that the bits of x would have not led to query.

This already poses a significant challenge to our approach, as we would have liked to extract a heavy daisy $\mathcal{D}$ from the collection $\mathcal{S}$ which well-approximates the query sets of M *independently of any input*. However, it could be the case that the sets that are relevant to an input x (i.e., $\text{supp}(\mu_x)$) induce a completely different daisy (with potentially different kernels over which we'll need to enumerate) than the relevant sets for a different input y that differs from x on the values in the kernel, and so it is not clear at all that there exists a single daisy that well-approximates the query behaviour of the adaptive algorithm M for all inputs.

Furthermore, the above also causes problems with the kernel enumeration process. For each assignment κ to the kernel K, denote by $x_\kappa \in \{0, 1\}^n$ the word that takes the values of κ in K and the values of x outside of K. Recall that the crux of our approach is to simulate executions of M^{x_κ}, for each kernel assignment κ, using the values of the sampled petals and plugging in the kernel assignment to complete these petals into local views (assignments to full query sets). Hence, since relevant sets corresponding to different kernel assignments may be distinct, it is unclear how to rule according to the local views that each of them induce.

We overcome these challenges in the next section with a more sophisticated extraction of daisies that, crucially, *does not discard any query sets* of the adaptive algorithm M. Specifically, we will partition the (multi)-collection of all possible query

for combinatorial analysis that was used in previous works involving sunflowers and daisies.

sets into a collection of daisies and *simultaneously analyse all daisies in the partition* to capture the adaptive behaviour of the algorithm.

3.1.3 Capturing adaptivity in daisy partitions

Relying on techniques from [FLV15, GL21], we can not only extract a single heavy daisy, but rather *partition* a (multi-)collection of query sets into a family of daisies, with strong structural properties on which we can capitalise. This allows us to apply our combinatorial machinery without dependency on a particular input, and *analyse all daisies simultaneously.*

Daisy partition lemma. A refinement of the daisy lemma in [GL21], which we call a *daisy partition lemma* (Lemma 4.1), partitions a multi-set $\mathcal{S}$ of q-sets into $q+1$ daisies $\{\mathcal{D}_i : 0 \leq i \leq q\}$ (see Figure 3.3) with the following structural properties.

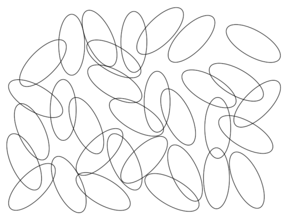
1. $\mathcal{D}_1$ is a $n^{1/q}$-daisy, and for $i > 1$, each $\mathcal{D}_i$ is a t-daisy with $t = n^{(i-1)/q}$;
2. The kernel K_0 of $\mathcal{D}_0$ coincides with that of $\mathcal{D}_1$, and, for $i > 0$, the kernel K_i of $\mathcal{D}_i$ satisfies $|K_i| \leq q|\mathcal{S}| \cdot n^{-i/q}$;
3. The petal $S \setminus K_i$ of every $S \in \mathcal{D}_i$ has size exactly i.

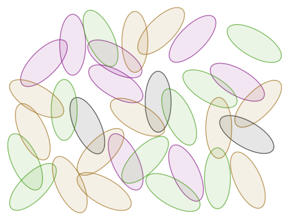
Moreover, the kernels form an incidence chain $K_q = \varnothing \subseteq K_{q-1} \subseteq \cdots \subseteq K_1 = K_0$. Note that $\mathcal{D}_0$ is vacuously a t-daisy for any t, since its petals are empty; and that our assumption on the randomness complexity of M implies $|K_i| = O(n^{1-i/q})$ when $i > 0$.

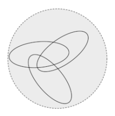
We may thus apply the daisy partition lemma to $\mathcal{S}$ and assert that, *for any input* x, there exists *some* $i \in \{0, \ldots, q\}$ such that $\mu_x(\mathcal{D}_i)$ is larger than $1/q$ (recall that, for all x, the support of μ_x is contained in $\mathcal{S}$); that is, each input may lead to a different heavy daisy, but there will always be at least one daisy that well-approximates the behaviour of the algorithm on input x. Alas, with only a local view of the input word, we are not able to tell which daisies are heavy and which are not.

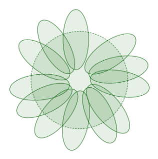
It is clear, then, that a sample-based algorithm that makes use of the daisy partition has to rule not only according to a single daisy, but rather according to all of them. But *how* exactly it should do so is a nontrivial question to answer, given that there are multiple daisies (and kernels) potentially interfering with one another.

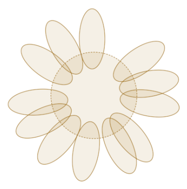
Adaptivity in daisy partitions. A natural approach for dealing with multiple daisies simultaneously is by enumerating over every assignment to *all* kernels (i.e., to


(a) A collection $\mathcal{S}$ of 3-sets before being partitioned.


(b) The collection $\mathcal{S}$ partitioned into 4 daisies: $\mathcal{D}_0$ (shaded in grey), $\mathcal{D}_1$ (green), $\mathcal{D}_2$ (yellow) and $\mathcal{D}_3$ (purple).


(c) $\mathcal{D}_0$, with sets entirely contained in the kernel K_0.


(d) $\mathcal{D}_1$ with $K_1 = K_0$, where each $S \in \mathcal{D}_1$ has a petal $S \setminus K_1$ of size 1.


(e) $\mathcal{D}_2$ with $K_2 \subseteq K_1$, where each $S \in \mathcal{D}_2$ has a petal $S \setminus K_2$ of size 2.

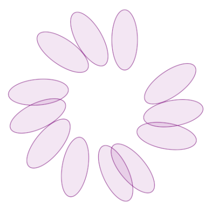

(f) $\mathcal{D}_3$, with kernel $K_3 = \varnothing$. Each $S \in \mathcal{D}_3$ has a petal $S \setminus K_3 = S$ of size 3.

Figure 3.3: Daisy partition.

$\cup_i K_i$) and, for each such assignment, obtaining local views from all daisies and ruling according to the aggregated local views. Note that the incidence chain structure implies that enumerating over assignments to K_0 suffices, since each assignment to K_0 induces assignments to K_i for all i.

However, this approach leads to fundamental difficulties. Recall that correctness of the sample-based algorithm on 0-inputs depends on *no kernel assignment* causing an output of 1. Although for any assignment to K_i this happens with sufficiently small probability to ensure it is unlikely to happen on all $2^{|K_i|}$ assignments simultaneously, this does not hold true for assignments to larger kernels. More precisely, since $|K_{i-1}|$ may be larger than $|K_i|$ by a factor of $n^{1/q}$, an error rate that is preserved by $2^{|K_i|}$ assignments becomes unbounded if the number of assignments increases to $2^{|K_{i-1}|}$. This leads us to only consider, for query sets in $\mathcal{D}_i$, assignments to K_i rather than to the union of all kernels.

Put differently, we construct an algorithm that deals with each daisy independently, and whose correctness follows from a delicate analysis that aggregates local views taken from all daisies, which we outline in Section 3.1.5. We begin by considering a sample-based local algorithm N that extends the strategy we used for a single daisy as follows. On input $x \in \{0,1\}^n$, it:

(1) samples each coordinate of the string x independently with probability $p = 1/n^{1/2q^2}$;
(2) for each $i \in \{0, \ldots, q\}$ and each assignment κ to the kernel K_i of the daisy $\mathcal{D}_i$, outputs 1 if a majority of local views leads M to output 1; and
(3) outputs 0 if no such majority is ever found.

First, note that since the algorithm N is constructed in a *white-box* manner, it has access to the description of all decision trees induced by the query-based algorithm M. Hence N^x is able to determine which local views correspond to a valid execution of M. Denoting by Q the set of coordinates that were sampled, an assignment κ to K_i induces, for each query set $S \subset Q \cup K_i$, the assignment $x_{\kappa|S}$; the sample-based algorithm N can check whether each such S is a relevant query set (i.e., belongs to the support of μ_{x_κ}) by verifying it arises from some branch of a decision tree of M that x_κ would have led to query. This allows N to ignore the non-relevant query sets and overcome the difficulty pointed out in the previous section.[6]

However, we remain with the issue that motivated searching for heavy daisies in the first place: there is no guarantee that every $\mathcal{D}_i$ well-approximates the algorithm M on all inputs. This is due to the use of relative estimates: if x is a 0-input and $\mu_x(\mathcal{D}_i)$ is smaller than the error rate σ, even when N considers the correct kernel assignment $x_{|K_i}$ with respect to x, it may find a majority of local views that leads M^x to output 1; indeed, nothing prevents all the "bad" query sets, which lead M^x to erroneously output 1, from being placed in the same daisy $\mathcal{D}_i$.

The solution is to use a simpler decision rule: absolute rather than relative. We count the number of local views leading to an output of 1, outputting 1 if and only if it crosses a threshold. The upper bound σ on the weight of "bad" query sets limits their number, and a large enough threshold prevents them from causing an incorrect output even if *no local view* leads to the correct one. Note that a different threshold τ_i is needed for each daisy $\mathcal{D}_i$, since the probability of sampling petals decreases as i increases. The thresholds τ_i must thus be carefully set to take this into account.

Finally, note that whenever the daisy $\mathcal{D}_0$ leads to an output of 1, this happens (almost) *independently of the input*: the assignment to every $S \in \mathcal{D}_0$ is determined solely by the assignment to K_0, because $S \subset K_0$. Therefore, the sample-based algorithm N disregards $\mathcal{D}_0$ in its execution.

[6]We remark that in the accurate description of our construction (see Section 4.4.1), we capture all the information contained in the decision trees via *tuples* that contain, besides the query set, the assignment that led to it being queried as well as the output of the algorithm when it does so. The daisy partition lemma then allows to partition these tuples based on the structure of the sets they contain.

The algorithm. By the discussion above, we obtain the following description for the sample-based algorithm N^x (with some parameters that we will set later).

1. Sample each coordinate of x independently with probability $p = 1/n^{1/2q^2}$. If the number of samples exceeds the desired sample complexity, abort.
2. For every $i \in [q]$ and every assignment κ to K_i, perform the following steps.
   (a) Count the number of sets in $\mathcal{D}_i$ with local views that lead M to output 1, which are *relevant for the assignment* κ and the queried values. If $i = 1$, discard the sets whose petals are shared by at least α local views.[7]
   (b) If the number is larger than the threshold τ_i, output 1.
3. If every assignment to every kernel failed to trigger an output of 1, then output 0.

In the next section we will present key technical tools that we develop and apply to analyse this algorithm, as well as discuss the parameters $\tau_i = \gamma_i \cdot np^i$ (where $\gamma_i = \Theta(1)$) and $\alpha = \Theta(1)$, and show it indeed suffices for the problem we set out to solve.

3.1.4 Two technical lemmas

To establish the correctness of the aforementioned sample-based algorithm, we shall first need two technical lemmas about sampling daisies. We will then proceed to provide an outline of the analysis of our algorithm.

We sketch the proofs of two simple, yet important technical lemmas that will be paramount to our analysis: (1) a lemma that allows us to transition from arguing about probability mass to arguing about combinatorial volume; and (2) a lemma that allows us to efficiently analyse sampling petals of daisies with complex intersection patterns.

The volume lemma. We start by showing how to derive from the probability mass of query sets (i.e., the probability under μ_x when the input is x) a bound on the volume that the union of these query sets cover. This is provided by the following *volume lemma*, which captures what is arguably *the defining structural property of robust local algorithms.*

Recall that the sample-based algorithm N uses the query sets of a $(\rho, 0)$-robust algorithm M with error rate σ, which comprise the support of the distributions μ_x

[7]The extra condition for $i = 1$ is necessary to deal with the looser intersection bound $t = n^{1/q} > n^{(i-1)/q}$ on $\mathcal{D}_1$. We discuss this in the next section.

for all inputs x. Intuitively, these sets cannot be too concentrated (i.e., cover little volume), as otherwise slightly corrupting a word (in less than ρn coordinates) could require M to output differently, a behaviour that is prevented by the robustness of M. This intuition is captured by the following *volume lemma.*

Lemma 3.1 (Lemma 4.3, informally stated)**.** *Let $x \in \{0,1\}^n$ be a non-robust input (a 1-input in our case) and $\mathcal{S}$ be a subcollection of query sets in the support of μ_x. If $\mathcal{S}$ covers little volume (i.e., $\cup\mathcal{S} < \rho n$), then it has small weight (i.e., $\mu_x(\mathcal{S}) < 2\sigma$).*

We stress that the *robustness of the* 0*-inputs yields the volume lemma for* 1*-inputs.*[8] Note that the contrapositive of the volume lemma yields a desirable property for our sample-based algorithm: for any (non-robust) 1-input x, the query sets in $\operatorname{supp}(\mu_x)$ must cover a large amount of volume, so that we can expect to sample many such sets.

The Hajnal–Szemerédi theorem. Once we establish that a daisy covers a large volume, it remains to argue how this affects the probability of sampling petals from this large daisy, which is a key component of our algorithm. Recall that sampling the petals of a sunflower is trivial to do. However, with the complex intersection patterns that the petals of a daisy could have, we need a tool to argue about sampling petals of daisies.

First, recall that the daisy partition lemma ensures that each $\mathcal{D}_i$ is a t-daisy where $t = n^{\max\{1,i-1\}/q}$, for all i. Observe that if $\mathcal{D}_i$ is a 1-daisy (which we call a *simple* daisy), that is, each point outside the kernel K_i is contained in at most one set $S \in \mathcal{D}_i$, then the sets in $\mathcal{D}_i$ have pairwise disjoint petals, so sampling them is *exactly* like sampling petals of a sunflower: these petals are sampled independently from one another, and we expect their number to be concentrated around the expectation of $p^i|\mathcal{D}_i|$ (recall that all petals have size i).

Of course, there is no guarantee that $\mathcal{D}_i$ is a simple daisy, though we expect it to *contain* a simple daisy if it is large enough. Indeed, greedily removing intersecting sets yields a simple daisy of size $\Theta(|\mathcal{D}_i|/t)$, but this does not suffice for our purposes because most of the sets in $\mathcal{D}_i$ are discarded.

Instead, we rely on the *Hajnal–Szemerédi theorem* to obtain a "lossless" transition from a t-daisy to a collection of simple daisies, from which sampling petals is easy. The Hajnal–Szemerédi theorem shows that for every graph G with m vertices and maximum degree $\Delta(G)$, and for any $k \geq \Delta(G) + 1$, there exists a k-colouring of the

[8]This is a rather subtle consequence of adaptivity; in the nonadaptive setting a symmetric volume lemma for b-inputs can be shown using robustness on b-inputs, for $b \in \{0,1\}$.

vertices of G such that every colour class has size either $\lfloor m/k \rfloor$ or $\lceil m/k \rceil$. By applying this theorem to the incidence graph G of the *petals* of query sets (i.e., the graph with vertex set $\mathcal{D}_i$ where we place an edge between S and S' when $(S \cap S') \setminus K_i \neq \varnothing$), which satisfies $\Delta(G) \leq 2t$ (see Claim 4.5) we obtain a partition of $\mathcal{D}_i$ into t simple daisies of the same size (up to an additive error of 1), and hence obtain stronger sampling bounds.

3.1.5 Analysis

With the two tools of Section 3.1.4, we proceed to the analysis proper. Note that the probability that N samples too many coordinates (thus aborts) is exponentially small, hence we assume hereafter that this event did not occur.

We proceed to sketch the high-level argument of the correctness of the sampled-based algorithm N, described in the previous section, making use of tools above. This follows from two claims that hold with high probability: (1) *correctness on non-robust inputs*, which ensures that when x is a 1-input (i.e., is non-robust), there exists $i \in [q]$ such that when N considers the kernel assignment $x_{|K_i}$ (which coincides with the input), the number of local views that lead to output 1 crosses the threshold τ_i; and (2) *correctness on robust inputs*, which, on the other hand, ensures that when x is a 0-input (i.e., is robust), for *every kernel* K_i and *every kernel assignment*, the number of local views that lead to output 1 *does not cross* the threshold τ_i.

In the following, we remind that when the sample-based algorithm N considers a particular assignment κ to a kernel and counts the number of local views that lead to output 1, the algorithm only considers views that are *relevant* to x_κ (the input x where the values of its kernel are replaced by κ); that is, local views that arise from some branch of a decision tree of the adaptive algorithm M that would have led it to query these local views. While N does not know all of x, after collecting samples from x and considering the kernel assignment κ, it can check which local views are relevant to x_κ (see discussion in Section 3.1.3).

Correctness on non-robust inputs. We start with the easier case, where x is a non-robust input (in our case, $f(x) = 1$). We show that there exists $i \in [q]$ such that when N considers the kernel assignment $x_{|K_i}$, the number of local views that lead to output 1 crosses the threshold $\tau_i = \gamma_i \cdot np^i$. We begin by recalling that N disregards the daisy $\mathcal{D}_0$, whose query sets are entirely contained in the kernel K_0, and arguing that this leaves sufficiently many query sets that lead to output 1. Indeed, while we could not afford this if $\mathcal{D}_0$ was heavily queried by M given the 1-input x (i.e., if $\mu_x(\mathcal{D}_0)$ is close to $1 - \sigma$), an application of the volume lemma shows this is not the case: since $|K_0| = o(n)$, this volume is smaller than ρn, implying $\mu_x(\mathcal{D}_0) < 2\sigma$ *for all* 1*-inputs* x.

Apart from $\mathcal{D}_0$, the query sets in the daisy $\mathcal{D}_1$ whose petals are shared by at least α local views (for a parameter α to be discussed shortly) are also discarded, and we need to show that the loss incurred by doing so is negligible as well. This is accomplished with a slightly more involved application of the volume lemma: since the sets of $\mathcal{D}_1$ have petals of size 1, the subcollection $\mathcal{C} \subseteq \mathcal{D}_1$ of sets that are discarded covers a volume of at most $|K_1| + |\mathcal{C}|/\alpha$. For a sufficiently large choice of a constant $\alpha > 0$, we have $|\mathcal{C}|/\alpha \leq \rho n/2$ (recall that $\mathcal{C} \subseteq \text{supp}(\mu_x)$ and $|\text{supp}(\mu_x)| = \Theta(n)$ by the assumption on the randomness complexity of M). Since $|K_1| = o(n)$ and in particular $|K_1| < \rho n/2$, applying the volume lemma to $\mathcal{C}$ shows that $\mu_x(\mathcal{C}) < 2\sigma$.

Finally, the total weight of all query sets in $\text{supp}(\mu_x)$ that lead to output 0 is at most σ (by definition of the error rate σ). This implies that the subcollection of $\text{supp}(\mu_x)$ that leads to output 1 *and is not disregarded* has weight at least $1 - 2\sigma - 2\sigma - \sigma = 1 - 5\sigma$, and, for a sufficiently small value of σ (recall that $\sigma = \Theta(1/q)$), we have $1 - 5\sigma \geq 1/2$.

We now shift perspectives, and in effect use the volume lemma in the contrapositive direction: large weights imply large volumes. By a simple averaging argument, it follows that at least one daisy $\mathcal{D}_i$ has weight at least $(1-5\sigma)/q \geq 2\sigma$, and thus, by the volume lemma, *covers at least* ρn *coordinates.* Therefore, since $|\text{supp}(\mu_x)| = \Theta(n)$ and μ_x *is uniform over a multi-collection of query sets*, this daisy contains $\Theta(n)$ "good" sets (that lead to output 1 and were not discarded). For the analysis, using the Hajnal–Szemerédi theorem, we partition the t-daisy $\mathcal{D}_i$ into t *simple* daisies of size $\Theta(n/t)$. Each such simple daisy has *disjoint* petals of size i, so that $\Omega(np^i/t)$ petals will be sampled except with probability $\exp\bigl(-\Omega(np^i/t)\bigr)$. Finally, this implies that, by setting $\gamma_i = \Theta(1)$ small enough, *when* N *considers the kernel assignment* $x_{|K_i}$ *to* K_i, at least $\tau_i = \gamma_i \cdot np^i$ petals are sampled except with probability

$$O(t) \cdot \exp\left(-\Omega\left(\frac{np^i}{t}\right)\right) = \exp\left(-\Omega\left(n^{1-\frac{\max\{1,i-1\}}{q}-\frac{i}{2q^2}}\right)\right) = o(1).$$

(Recall that $t = n^{\max\{1,i-1\}/q}$ and the sampling probability is $p = 1/n^{1/2q^2}$.)

Correctness on robust inputs. It remains to show the harder case: when x is a robust input (in our case, $f(x) = 0$), then *all kernel assignments to all daisies* will make the local views that lead to output 1 fail to cross the threshold. This case is harder since, by the asymmetry of N with respect to robust and non-robust inputs, here we need to prove a claim for all kernel assignments to all daisies, whereas in the non-robust case above we only had to argue about the existence of a single assignment to a single kernel.

We begin with a simple observation regarding $\mathcal{D}_0$, then analyse the daisies $\mathcal{D}_i$ for $i > 1$, and deal with the more delicate case of $\mathcal{D}_1$ last. Recall that $\mathcal{D}_0$ is disregarded by the algorithm N, and that by the asymmetry of N with respect to 0- and 1-inputs, this only makes the analysis on robust inputs *easier*. Indeed, N^x is correct when no kernel assignment to any of the $\mathcal{D}_i$'s leads to crossing the threshold τ_i of local views on which the query-based algorithm M outputs 1. Thus, by discarding the query sets in $\mathcal{D}_0$, we only increase the probability of not crossing these thresholds.

Fix $i > 0$ and an arbitrary kernel assignment κ to K_i. Then, the relevant sets that N may sample are those in the support of μ_{x_κ} (recall that x_κ is the word obtained by replacing the bits of x whose coordinates lie in K_i by κ). Since $|K_i| = o(n)$, it follows by the robustness of x that x_κ is ρ-close to x, and thus the weight of the collection $\mathcal{O} \subseteq \text{supp}(\mu_{x_\kappa})$ of query sets that lead to output 1 is at most σ. For the sake of this technical overview, we focus on the worst-case scenario, where all of these "bad" sets are in the daisy $\mathcal{D}_i$ (i.e., $\mathcal{O} \subseteq \mathcal{D}_i$) and $|\mathcal{O}|$ is as large as possible (i.e., $|\mathcal{O}| = \Theta(n)$), and show that even that will not suffice to cross the threshold τ_i.

By the randomness complexity of the algorithm M, the size of $\mathcal{O}$ is $\Theta(n)$. We apply the Hajnal–Szemerédi theorem and partition $\mathcal{O}$ into $\Theta(t)$ simple daisies of size $\Theta(n/t)$. Recall that the petals of query sets in $\mathcal{D}_i$ have size i and are disjoint; therefore, each of these simple daisies has $\gamma_i \cdot np^i/t$ sampled petals with probability only $\exp\bigl(-\Omega(np^i/t)\bigr)$.[9] By an averaging argument, the total number of sampled petals crosses $\tau_i = \gamma_i \cdot np^i$ with probability at most

$$O(t) \cdot \exp\left(-\Omega\left(\frac{np^i}{t}\right)\right) = \exp\left(-\Omega\left(n^{1-\frac{i-1}{q}-\frac{i}{2q^2}}\right)\right) = \exp\left(-\Omega\left(n^{1-\frac{i}{q}+\frac{1}{2q}}\right)\right) \ ;$$

recall that $t = n^{(i-1)/q}$ and the sampling probability is $p = 1/n^{1/2q^2}$, so $p^i \geq 1/n^{1/2q}$. Since the daisy partition lemma yields a bound of $O(n^{1-i/q})$ for the size of the kernel K_i, a union bound over all $2^{|K_i|}$ kernel assignments ensures the threshold is crossed with probability $o(1)$.[10]

[9]We stress that *the expected number of sampled petals is smaller* in the robust case than in the non-robust one. This is what allows us to show the total number of queried petals is at least $\tau_i = \gamma_i np^i$ with probability $\exp\bigl(-\Omega(np^i/t)\bigr)$ in the robust case but $1 - \exp\bigl(-\Omega(np^i/t)\bigr)$ in the non-robust, for the same constant γ_i.

[10]Recall that we set the sampling probability to be $p := 1/n^{1/\beta}$ with $\beta = 2q^2$. This choice is justified as follows: the union bound requirement that $2^{|K_i|}$ multiplied by the probability of crossing the threshold be small translates into $1/q - i/\beta > 0$ for all $i \in [q-1]$. Then $i = q-1$ requires $\beta = \Omega(q^2)$.

We now analyse $\mathcal{D}_1$ and stress that the need for a separate analysis arises from the looser intersection bound on this daisy: $\mathcal{D}_1$ is a t-daisy with $t = n^{1/q}$, whereas for all other i the bound is $t = n^{(i-1)/q}$. This implies that there is no "gap" between the expected number of queried petals in each simple daisy $\Theta(np/t) = \Theta(n^{1-1/q-1/(2q^2)}) = o(n^{1-1/q})$ and the size of the kernel $|K_1| = O(n^{1-1/q})$, so a union bound as in the case $i > 1$ does not suffice.

This is precisely what the "capping" performed by N on $\mathcal{D}_1$ is designed to address: the query sets $\mathcal{O} \subseteq \operatorname{supp}(\mu_{x_\kappa})$ that lead to output 1 will only be counted by N *if their petals are shared by at most α query sets.* Then, by the Hajnal–Szemerédi theorem, we partition $\mathcal{O}$ into $\alpha = \Theta(1)$ simple daisies of size $\Theta(n)$. Each simple daisy will have more than $\tau_i/\alpha = \Theta(np)$ queried petals with probability at most $\exp(-\Omega(np))$, so that the total number of such petals across all simple daisies exceeds τ_j with probability at most $\exp(-\Omega(np))$. This provides the necessary gap: as $\Theta(np) = \Omega(n^{1-1/2q^2})$ and $|K_1| = O(n^{1-1/q})$, a union bound over all $2^{|K_1|}$ assignments to K_1 shows the threshold is crossed with probability $o(1)$.

This concludes our high-level proof of correctness, and thus of Theorem 1 (see Section 4.4.2 for the full proof). For an overview of how to derive our applications from Theorem 1, see Section 4.5.

3.2 Quantum proofs of proximity

We now proceed to a discussion of the high-level ideas of Chapter 5, which contains proofs of the theorems stated in Section 1.2.2. Our discussion is divided into lower bounds and algorithmic techniques.

In Section 3.2.1, we introduce some of the lower bound techniques that we use in charting the complexity landscape of quantum proofs of proximity. En route, we extend the framework of Blais, Brody and Matulef [BBM12] to show lower bounds for *quantum* property testers. To the best of our knowledge, this is the first quantum testing lower bound proved via a reduction from quantum communication complexity, an open question raised by Montanaro and de Wolf [MW16, Question 4]. In addition, we show how to prove lower bounds on QMAP algorithms via an argument about the threshold degree of boolean functions.

In Section 3.2.2 we show how to construct quantum proofs of proximity for properties that can be decomposed into sub-problems, and we prove that these QMAP protocols outperform both quantum testers as well as classical proof of proximity protocols. Moreover, we give an overview of an efficient QMAP protocol for a natural property of bounded-degree graphs, *bipartiteness*, which does not fall into the decomposability paradigm.

3.2.1 Lower bounds

We highlight two techniques that we exploit to prove complexity separations and limitations on QMAPs: (1) proving quantum testing lower bounds via reductions from quantum communication complexity [BBM12], which we use to show a separation between MAPs and quantum testers; and (2) proving QMAP lower bounds by studying the threshold degree of boolean functions.

Quantum testing lower bounds via reductions from communication complexity. The methodology of [BBM12] has proven very successful for showing *classical* property testing lower bounds. However, extending this methodology to the quantum setting poses an inherent difficulty that we expand upon next. Following the exposition of [MW16], we illustrate the methodology and the difficulty in the quantum setting by considering the problem of testing whether a function $f\colon \{0,1\}^n \to \{0,1\}$ is k-linear, i.e., a Fourier character of weight k.

We can obtain query complexity lower bounds on testers via a reduction from the randomised *communication complexity* problem of disjointness, as follows. Recall that, in the disjointness problem, Alice receives $x \in \{0,1\}^n$ and Bob receives $y \in \{0,1\}^n$ (for lower bound purposes, we may assume without loss of generality that both bit strings are promised to have Hamming weight $k/2$ for some known $k \in [n]$), and their goal is to decide whether or not there exists an index $i \in [k]$ such that $x_i = y_i = 1$, while communicating a minimal number of bits.

Suppose that there exists a property tester for k-linearity with query complexity q. We will use this tester to construct a communication complexity protocol for disjointness (i.e., deciding if, for every $i \in [n]$, either $x_i = 0$ or $y_i = 0$) as follows. First, Alice and Bob use shared randomness and simulate the tester on the input f, interpreted as a function mapping $\{0,1\}^n$ to $\{0,1\}$ defined as $f(z) = \bigoplus_{i\in[n]} z_i \cdot (x_i \oplus y_i)$. To simulate a query $f(z)$, Alice computes $A(z) = \bigoplus_{i\in[n]} z_i \cdot x_i$ and sends it to Bob, while Bob computes $B(z) = \bigoplus_{i\in[n]} z_i \cdot y_i$ and sends it to Alice. Since $f(z) = A(z) \oplus B(z)$, each query to f incurs 2 bits of communication. Moreover, if x and y are disjoint, then f is k-linear; and if they are not disjoint, f is ℓ-linear for some $\ell < k$, and is in particular $1/2$-far from every k-linear function. Therefore, the simulated tester indeed solves the communication problem, so that the $\Omega(k)$ lower bound for the latter implies an $\Omega(k)$ lower bound for testing k-linearity.

An attempt to extend this to *quantum* testers, however, reveals a severe bottleneck in the reduction. Note that, classically, the fact that Alice and Bob can use shared randomness to fix a *deterministic* tester to simulate is crucial: at every step, both parties know which query z the tester will make next *without* the need to communicate it. The problem is that there is no way to fix the "quantumness" using shared randomness. Details follow.

While disjointness is still hard in the quantum communication complexity model, communicating the query (which may be in a superposition) that the quantum tester requires will incur a *linear* overhead, rendering the reduction useless. Namely, to simulate a query to f *in superposition*, the parties need to exchange all n qubits at each round: Alice would apply the unitary (on $n+1$ qubits) $|z\rangle|b\rangle \mapsto |z\rangle|b \oplus A(z)\rangle$, and send the $(n+1)$-qubit state to Bob, who applies $|z\rangle|b \oplus A(z)\rangle \mapsto |z\rangle|b \oplus A(z) \oplus B(z)\rangle = |z\rangle|b \oplus f(z)\rangle$ and returns them to Alice. Simulating a single query then requires the communication of $2n+2$ qubits, rather than the 2 needed by a classical tester. Thus, the reduction can only prove a degenerate $\Omega(1)$ testing lower bound. This is, in fact, not surprising, since k-linearity is testable with $O(1)$ queries by the Bernstein-Vazirani [BFNR08] algorithm!

While the discussion above might suggest that communication complexity can only yield trivial quantum testing lower bounds, we show this is not the case; indeed, the absence of nontrivial (quantum) applications of the technique thus far points to a conceptual barrier that is clarified by a coding-theoretic perspective similar to [Gol20], which reveals that the linear overhead is not inherent to any quantum reduction. Observe that testing k-linearity is a special case of *testing a subset of a code*: namely, a k-linear function f where $f(z) = w \cdot z$ corresponds to the *Hadamard encoding* of the string w with Hamming weight k (which maps $w \in \{0,1\}^n$ into the codeword $C(w) = (w \cdot z : z \in \{0,1\}^n)$ with blocklength $n' = 2^n$). Note that the aforementioned quantum simulation strategy is efficient, in the sense that it requires only $O(\log n')$ qubits to communicate a representation of the length-n' encoding; the issue is the Hadamard code's exponential blocklength $n' = 2^n$, which renders the simulation's efficiency moot. As we see next, however, *the same reduction* yields nontrivial bounds if we choose the code appropriately.

Given a linear code $C\colon \{0,1\}^n \to \{0,1\}^{n'}$ with $n' = \mathrm{poly}(n)$, only $\log n' = O(\log n)$ qubits are necessary to represent $C(x)$ and $C(y)$. More precisely, Alice can apply the $O(\log n)$-qubit unitary $|i\rangle|b\rangle \mapsto |i\rangle|b \oplus C(x)_i\rangle$ and send all $O(\log n)$ qubits to Bob, who applies $|i\rangle|b\rangle \mapsto |i\rangle|b \oplus C(y)_i\rangle$ and returns them. This composition of unitaries is

$$|i\rangle|b\rangle \mapsto |i\rangle|b \oplus C(x)_i \oplus C(y)_i\rangle = |i\rangle|b \oplus C(x \oplus y)_i\rangle\ ,$$

simulating a query with *logarithmic*, rather than linear, overhead.

Therefore, the $\Omega(\sqrt{n})$ quantum communication lower bound for disjointness [Raz03] implies an $(n')^{\Omega(1)}$ lower bound for the problem of testing a subset of C. Indeed, we show that, for a linear code $C\colon \mathbb{F}^n \to \mathbb{F}^{n'}$ (over a larger field of odd characteristic), the property $\{C(z) : z \in \{0,1\}^n\}$, of *booleanity,*[11] which may be of interest in PCP constructions, has a quantum testing lower bound of $\Omega(\sqrt{n}/\log n)$ via a reduction from disjointness (see Section 5.2.2 for details). We remark that since this technique is used to show a separation between quantum testers and MA proofs of proximity, we use codes that are *locally testable* and *relaxed locally decodable*, which allow for efficient testing by a MAP. Since there exist such codes with a nearly-linear blocklength [BGH+06, AS21], the lower bound we obtain is only slightly worse than a square root.

QMAP lower bounds via threshold degree. We prove lower bounds for QMAPs via the *threshold degree* of related functions. A function $f\colon \{0,1\}^n \to \{0,1\}$ is said to have threshold degree (at most) d if there exists a degree-d polynomial $P(X_1, \ldots, X_n)$ over $\mathcal{R}$ such that $f(x) = 1$ if $P(x) > 0$ and $f(x) = 0$ if $P(x) < 0$; in other words, the threshold degree of f is the smallest degree of a polynomial that *sign-represents* f.

As a first step, we show that the inclusion $\mathsf{QMA} \subseteq \mathsf{PP}$ [MW05] (in the *polynomial-time* setting, implied by the technique known as Marriott-Watrous amplification) carries over to the property testing setting, implying $\mathsf{QMAP} \subseteq \mathsf{UPP}$.[12] Next, we show that the query complexity of a UPP algorithm that computes f is exactly the threshold degree of f (this result is folklore, but we provide a proof for completeness). Since a property Π induces the (partial) function f_Π such that $f_\Pi(x) = 1$ when $x \in \Pi$ and $f_\Pi(x) = 0$ when x is ε-far from Π, the query complexity of a UPP algorithm that "tests" Π (i.e., computes f_Π) is a lower bound on the product pq of the proof and query complexities of any QMAP protocol for testing Π. Finally, we show that if Π is *k-wise independent* (i.e., looks perfectly random on any subset of k coordinates) and not too large, the threshold degree of f_Π is at least k, so that $pq = \Omega(k)$ (see Section 5.3 for details).

In particular, any code with linear dual distance and small enough rate is an example of a hard property for QMAPs, requiring proof and query complexities that

[11] In fact, we show (and use to prove the separation $\mathsf{QPT} \not\subseteq \mathsf{MAP}$) a lower bound for *non*-booleanity; but the symmetry of the model of communcation complexity implies the same lower bound holds for booleanity as well.

[12] UPP is the query model version of the class PP of unbounded-error randomised algorithms, where in particular the amount of randomness available to the algorithm is unbounded. Since PP algorithms run in polynomial time, they may access at most a polynomial number of random bits; this restriction does not hold for UPP.

satisfy $pq = \Omega(n)$ for proximity parameter $\varepsilon = \Omega(1)$.

3.2.2 Algorithmic techniques

As a warm-up, consider the *exact decision* problem of verifying that an n-bit string x has even parity. This is maximally hard for both IP algorithms and quantum query algorithms, requiring $\Omega(n)$ queries to the bit string, and thus asymptotically no better than trivially querying every coordinate. As we will see next, however, QMA algorithms can capitalise on having a proof *and* quantum processing power to break the linear barrier.

We rely on the technique of *amplitude amplification* [BHMT02] to obtain such an algorithm with sublinear proof and query complexities. Loosely speaking, amplitude amplification takes a (randomised) decision algorithm that always accepts yes-inputs and rejects no-inputs with probability ρ, and produces an algorithm with rejection probability $2/3$ (for no-inputs) using the former algorithm only $O(1/\sqrt{\rho})$ times as a subroutine.

We can thus obtain a QMA (query) algorithm for the parity problem as follows. The proof string specifies the purported parities of each block of an equipartition of the input $x \in \{0,1\}^n$ into p blocks of length n/p. The verifier first checks that the proof string has even parity, rejecting immediately otherwise. Then, the verifier performs amplitude amplification on the following subroutine: sample $i \in [p]$ uniformly at random, read the entire block of n/p bits and check that its parity coincides with that claimed by the proof; if so, accept, and reject otherwise.

Note that the aforementioned subroutine always accepts if x has even parity and the proof corresponds to the parity of every block. On the other hand, if a string has odd parity and the proof has even parity, *at least one bit of the proof* disagrees with the corresponding block, so that the subroutine rejects with probability at least $1/p$. Since we need only repeat $O(\sqrt{p})$ times, each of which queries n/p bits, the query complexity of our algorithm is $q = O(n/\sqrt{p})$; in particular, if $p = n^{2/3}$ then $q = O(n^{2/3})$.

This is a special case of a more general phenomenon, which holds for all *decomposable* properties (see Section 5.5.3 for a discussion of how exact decision follows as a special case). Since amplitude amplification can only be applied to one-sided algorithms (i.e., those that always accept a valid input), we restrict our attention to this type of algorithm hereafter.

Decomposable properties. A property Π is (k,s)-decomposable if a "specification" of length s efficiently reduces testing Π to testing k smaller properties $\Lambda^{(1)}, \dots, \Lambda^{(k)}$.

More precisely, Π is (k, s)-decomposable if,

1. there exists some s-bit string that specifies a set of k properties $\Lambda^{(i)}$ as well as k strings $x^{(i)} \in \{0,1\}^{m_i}$ whose bits are determined by a small number of bits of x; and

2. ε-testing $x \in \{0,1\}^n$ with respect to Π reduces to testing $x^{(i)}$ with respect to $\Lambda^{(i)}$ in the following sense: when $x \in \Pi$ then $x^{(i)} \in \Lambda^{(i)}$ for all $i \in [k]$, whereas when x is ε-far from Π, then $x^{(i)}$ is ε_i-far from $\Lambda^{(i)}$ for some ε_i satisfying $\mathbb{E}_i[\varepsilon_i] = \Omega(\varepsilon)$, where the expectation over i means that i is sampled with probability proportional to m_i.

If the specification is short (i.e., $s = O(k \log n)$), we say Π is *succinctly* k-decomposable (see Section 5.5 for details). Decomposable properties generalise the notion of *parametrised k-concatenation properties* introduced in [GR18], which corresponds to the special case of a $(k, 0)$-decomposition that is an equipartition of the input string.

Our simplest example of a decomposable problem is that of testing the set of k-monotone functions $f : [n] \to \{0,1\}$, i.e., functions that change from non-decreasing to non-increasing and vice-versa at most $k-1$ times. A natural decomposition of this property is to specify the set of at most $k-1$ "critical points", which induce a set of at most k subfunctions f_i that are monotone and overlap with f_{i-1} and f_{i+1} at their endpoints; then, it suffices to test for (1-)monotonicity of each subfunction. More precisely, this property is $(k, (k-1)\log n)$-decomposable (thus succinctly k-decomposable), and given the (alleged) critical points $n_1 < n_2 < \cdots < n_{k-1}$, the subproperty $\Lambda^{(i)}$ for odd (resp. even) i is the set of non-decreasing (resp. non-increasing) functions on $[m_i]$, where $m_i = n_i - n_{i-1} + 1$ (with $n_0 = 1$ and $n_k = n$). Note, moreover, that if f is ε-far from k-monotone, then its absolute distance from all functions specified by the critical points is at least εn; thus, denoting by ε_i the distance of f_i to $\Lambda^{(i)}$, we have $\sum_i \varepsilon_i m_i \geq \varepsilon n$, implying $\mathbb{E}_i[\varepsilon_i] = \Omega(\varepsilon)$. We remark that decomposing other properties (e.g., branching programs, context-free languages, and Eulerian graph orientations) is much less straightforward and often allows for breaking the property into any desired number of sub-properties, which in turn admits proof length versus query complexity tradeoffs. See Section 5.5 for details.

Given a (k, s)-decomposable property that admits MAPs for the subproperties $\Lambda^{(i)}$, a natural protocol for Π is to sample $i \in [k]$ uniformly at random and execute the verifier for $\Lambda^{(i)}$. Note that, if these MAPs have proof complexity p and query complexity q, the protocol for Π has proof length $s + kp$. Moreover, $\mathbb{E}_i[\varepsilon_i] = \Omega(\varepsilon)$ means that a randomly chosen $i \in [k]$ is (in expectation) at distance roughly ε from $\Lambda^{(i)}$, so it is reasonable to expect that $O(1/\varepsilon)$ classical repetitions of the base protocol would ensure a rejection with high probability, and that a QMAP protocol can make do with only $O(1/\sqrt{\varepsilon})$ repetitions using amplitude amplification.

The above outline glosses over the fact that we have no information on *the distribution of errors* $(\varepsilon_1, \dots, \varepsilon_k)$. For example, it may be that most ε_i are of the same order of magnitude (in which case a random $i \in [k]$ is likely to point to a mildly corrupted $x^{(i)}$), or it may be that a few ε_i are very large while all other ε_i are small or even zero (in which case $x^{(i)}$ is unlikely to be corrupted for a random $i \in [k]$, but when it is, the amount of corruption is large). Fortunately, this issue can be addressed by the technique of *precision sampling* [Lev87], incurring a merely logarithmic overhead. We thus obtain a QMAP protocol for ε-testing Π with proof complexity $s + kp$ and query complexity comparable to q (and often smaller; see Theorem 5.14 for details).

Bipartiteness testing. Consider the problem of testing whether a bounded-degree graph G (given as an oracle to its adjacency list) is bipartite or far from any bipartite graph. (Note that this is not a decomposable property.) There exists a MAP protocol for a promise variant of this problem, where graphs are *rapidly-mixing* [GR18]. We will show that it is possible to combine quantum speedups obtained by amplitude amplification *and* by replacing a classical subroutine with a more efficient quantum analogue.

Let us first consider the (classical) MAP verifier for bipartiteness, which receives a subset of vertices S of size k, allegedly on the same side of a bipartition, as a proof. To test with respect to proximity parameter ε, the verifier repeats the following procedure: sample a uniformly random vertex v, take roughly $n/(k\varepsilon)$ short (lazy) random walks starting from v, recording whether the walk ended at a vertex in S as well as the parity of the walk (i.e., the parity of the number of non-lazy steps). If two walks start from the same vertex v and end in S with different parities, then reject; otherwise, accept. Setting $m \coloneqq n/k$, the query complexity of (one iteration of) the verifier is m/ε (ignoring constants and polylogarithmic factors).

If the graph is bipartite and the proof S is indeed on the same side of a bipartition, there cannot exist two paths from the same vertex into S with different parities (as that would imply a path of odd length with both endpoints on the same side). Therefore, the verifier always accepts in this case. If the graph is ε-far from bipartite, however, each iteration finds evidence to this effect with probability $\Omega(\varepsilon)$. Thus, the classical verifier samples a new vertex roughly $1/\varepsilon$ times, for a total query complexity of m/ε^2.

Now, one immediate way to improve this algorithm is to perform amplitude amplification: the resulting algorithm repeats the procedure $1/\sqrt{\varepsilon}$ times, improving the query complexity to $m/\varepsilon^{3/2}$. A second (and less straightforward) strategy is to use the quantum *collision-finding* algorithm [Amb07] to reduce the number of random walks taken from each vertex to $(m/\varepsilon)^{2/3}$, as in [ACL11].[13] This strategy reduces the required number of queries to $m^{2/3}/\varepsilon^{5/3}$, improving the dependency on m but achieving a worse one on ε.

Of course, this begs the question: why not apply both optimisations? Indeed, we show how to tweak the classical MAP verifier in order to do so, and thus simultaneously obtain the speedups from each of them.[14]

More precisely, sample a uniformly random vertex v and let g_v denote the mapping $r \mapsto (a, b) \in \{0,1\}^2$ obtained by executing a random walk starting from v with r as its inner randomness, where $a = 1$ if the walk stops at a vertex in S and b is the parity of the walk. The collision-finding algorithm is capable of finding a pair r_0, r_1 such that $g_v(r_i) = (1, i)$ for $i \in \{0,1\}$, if such a pair exists. The query complexity of the collision-finding algorithm is the domain size to a $2/3$ power, and, since we take m/ε walks from v, the number of queries is $(m/\varepsilon)^{2/3}$. Although such a collision is not guaranteed to exist for all starting vertices v, it is for a fraction of roughly ε of them. By applying amplitude amplification to the procedure described in this paragraph, we obtain a QMAP protocol for bipartiteness with proof length $O(k \log n)$ and query complexity $\tilde{O}((m/\varepsilon)^{2/3} \cdot 1/\sqrt{\varepsilon}) = \tilde{O}((n/k)^{2/3}/\varepsilon^{5/6})$ (see Theorem 5.18 for details).

3.3 Streaming zero-knowledge proofs

For concreteness, we focus on the construction of zero-knowledge SIPs for one of the most fundamental problems in the data stream model: INDEX.

We begin with a bird's eye view of our ideas and the challenges that arise in their implementation. The starting point of our efforts is Section 3.3.1, where we describe the *polynomial evaluation protocol* (`pep`), from which a (non zero-knowledge) SIP for the INDEX problem follows. An attempt to make this protocol zero-knowledge faces two fundamental challenges, which we address in Sections 3.3.2 and 3.3.3 via the construction of two types of *streaming commitment protocols*.

[13]Here and throughout, we use the term collision-finding to refer to Ambainis's algorithm that, for any f with 1-certificate complexity at most 2, uses $\Theta(n^{2/3})$ queries and with constant probability outputs a 1-certificate when run on any input $x \in f^{-1}(1)$.

[14]Amplitude amplification requires that the algorithm be *invertible*, i.e., be given by a unitary A, as the technique repeatedly applies A and A^{-1}. For this reason, it is often said to apply to quantum algorithms without intermediate measurements (as these make an algorithm non-invertible), which is not the case for collision-finding. However, the (standard) *principle of deferred measurement* (see, e.g., [NC16]) allows us to transform any quantum algorithm A into a reversible A' with the same query complexity, and apply amplitude amplification to the latter.

In Section 3.3.4, we apply the foregoing protocols to obtain a streaming interactive proof for INDEX and briefly discuss the proof of its zero-knowledge property, which requires an involved simulator argument. Finally, Section 3.3.5 sketches another application of this framework that obtains an additional powerful and flexible tool: a *zero-knowledge streaming sumcheck* protocol.

3.3.1 A starting point: the polynomial evaluation protocol

Recall that in the INDEX problem, a streaming algorithm with s bits of memory reads a length-n string x over an alphabet Γ (one symbol at a time), followed by a coordinate $j \in [n]$, and its goal is to output $x_j \in \Gamma$. It is well-known that INDEX is maximally hard for streaming algorithms, requiring $s = \Omega(n)$ space for the output to be correct with nontrivial probability.

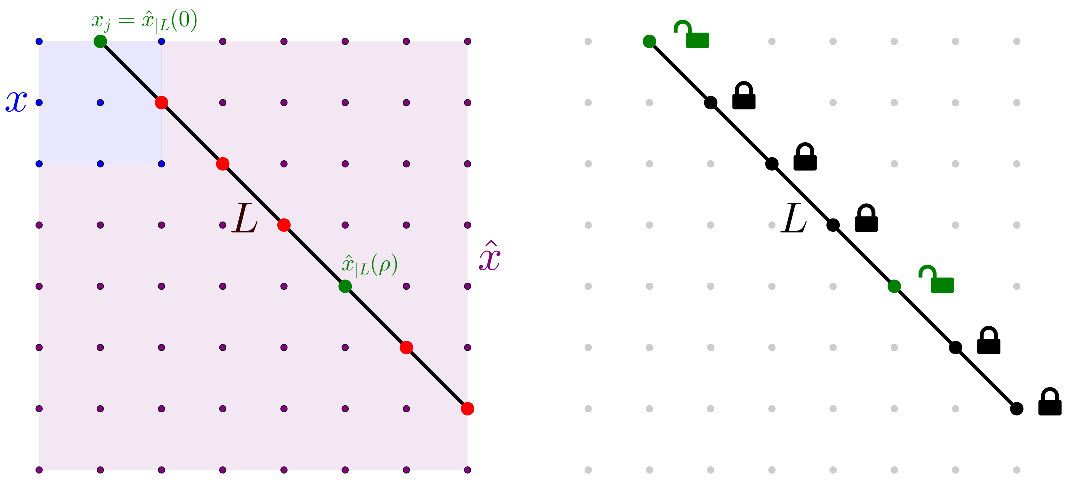
First, note that obtaining an efficient SIP for INDEX is non-trivial even without zero-knowledge. Indeed, the naive approach of having the prover P reveal the index j before V streams x, allowing the verifier to only store x_j, fails: both parties observe *the same* stream of information (recall Section 1.3), so P only learns j long after V has seen x_j. Any communication in an SIP before the input stream must therefore be *independent* of it.

Remarkably, an exponential reduction in space complexity is possible despite both prover and verifier not knowing the index j before it appears in the stream. We recall the SIP in [CCM+15, CCM+19], upon which we build, and argue why it is *not* zero-knowledge to begin with. Their SIP is an application of pep, the *polynomial evaluation protocol* (Protocol 6.1), which enables a small-space algorithm to recover any element that was streamed but not stored, using only a small fingerprint of the stream.

We embed the input stream into an object with algebraic structure in a space of size much larger than n, namely, by viewing $x_i \in \mathbb{F}$, for a large enough finite field $\mathbb{F}$, and considering an m-variate *low-degree polynomial* $\hat{x}$ that interpolates across all x_i (recall Section 2.5); we call the polynomial $\hat{x} : \mathbb{F}^m \to \mathbb{F}$ of individual degree $d = d(m,n)$ the low-degree extension (LDE) of x. (Usual parameter settings satsify $d, m \leq \log n$ and $|\mathbb{F}| = \operatorname{polylog} n$.)

The protocol proceeds as follows. The verifier samples a random evaluation point $\boldsymbol{\rho} \sim \mathbb{F}^m$ and computes the *fingerprint* $\hat{x}(\boldsymbol{\rho})$, which can be evaluated in low space via standard online Lagrange interpolation. After V learns j, it enlists P in the recovery of x_j: it sends P a line $L : \mathbb{F} \to \mathbb{F}^m$ incident to j (viewing this index as an element of $\mathbb{F}^m$) and $\boldsymbol{\rho}$, where $L(0) = j$ and $L(\rho) = \boldsymbol{\rho}$ for a random $\rho \sim \mathbb{F}$, whereupon P replies with the (low-degree) univariate polynomial $\hat{x}_{|L} = \hat{x} \circ L$.

If P is honest, then V can easily recover $x_j = \hat{x}(j) = \hat{x}_{|L}(0)$. However, P could easily cheat if V made no further checks: the prover could just as well pick $\alpha \in \mathbb{F}$ arbitrarily and send any low-degree polynomial g such that $g(0) = \alpha$ to (falsely) convince V that $x_j = \alpha$. By having V only accept the prover's claim that $x_j = g(0)$ *if g also agrees with the fingerprint*, i.e., if $g(\rho) = \hat{x}_{|L}(\rho) = \hat{x}(\boldsymbol{\rho})$, the verifier thwarts this (and any other) attack: since both $\boldsymbol{\rho}$ and ρ are unknown to the prover, to convince the verifier of an incorrect answer $g(0) \neq \hat{x}_{|L}(0)$, the prover must send a polynomial $g \neq \hat{x}_{|L}$ that agrees with $\hat{x}_{|L}$ at a random point; and if $\mathbb{F}$ is sufficiently large, the probability of this event (ρ being a root of the nonzero polynomial $g - \hat{x}_{|L}$) is arbitrarily small.



(a) V streams x (in blue), learns $\hat{x}(\boldsymbol{\rho}) = \hat{x}_{|L}(\rho)$ and sends L. The prover replies with $\hat{x}_{|L}$, revealing x_j and $\hat{x}(\boldsymbol{\rho})$ (in green) along with evaluations of $\hat{x}$ that V cannot learn on its own (in red).

(b) A first attempt at preventing leakage: sending the evaluation table of $\hat{x}_{|L}$ in "locked boxes" and only unlocking the points checked by the verifier.

Figure 3.4: Leakage in the SIP for INDEX via evaluation of the bivariate polynomial $\hat{x} : \mathbb{F}^2 \to \mathbb{F}$, and an (unsuccessful) attempt to prevent it.

The protocol outlined above is, however, *not* zero-knowledge: after all, V learns not only x_j, but the restriction of $\hat{x}$ to an entire line L *through* j (see Figure 3.4a). Note that learning the restriction of $\hat{x}$ to (say) a random line R does not necessarily constitute leakage: V could simply compute a few evaluations (rather than only one) of $\hat{x}_{|R}$, which fully determine the polynomial. The issue is that L is a function of the coordinate j, which V does not know prior to streaming x.

In the next section we will take our first steps towards making the protocol zero-knowledge, i.e., ensuring that the verifier learns nothing beyond the value x_j. Note that the honest V only evaluates $\hat{x}_{|L}$ at two points, ρ and 0; what if P could send the evaluations of $\hat{x}_{|L}$ in “locked boxes” and only open the pair that the verifier needs?

3.3.2 Curtailing leakage with commitments

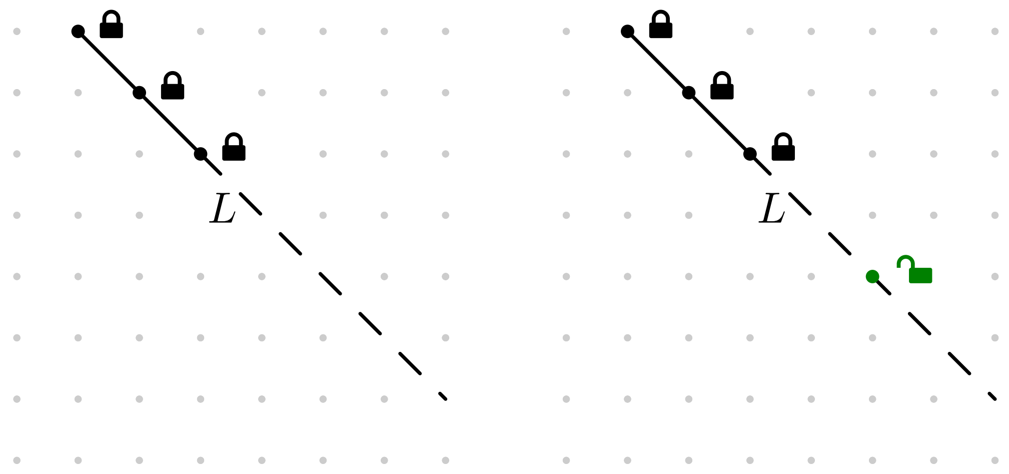
To make the foregoing approach more precise, let us first assume the existence of a *commitment protocol* that allows P to transmit any field element α to V in two steps: sending a string $\mathsf{commit}(\alpha)$, from which V is unable to extract any information about α; and later, upon the verifier's request, revealing a field element β such that, if $\beta \neq \alpha$, then V can detect that the P is being dishonest.

With such a commitment protocol in hand, a natural attempt to prevent the pep protocol from leaking information is to have the prover P send a commitment to $\hat{x}_{|L}$, the restriction of the input's LDE to the line chosen by V (rather than sending the polynomial in the clear). That is, the prover would commit to the evaluation table of $\hat{x}_{|L}$, sending $\big(\mathsf{commit}(\hat{x}_{|L}(\rho')) : \rho' \in \mathbb{F}\big)$, after which V can reveal its random evaluation point ρ and P decommits *only* to the evaluations of 0 and ρ (see Figure 3.4b). This does indeed reveal less information (2 rather than $|\mathbb{F}|$ evaluations of $\hat{x}$), but is still far from what we set out for.

There are two severe shortcomings with this idea; we shall tackle one now and defer the other to Section 3.3.3. First we need to ask: what is to prevent a cheating prover from committing to a function g that is inconsistent with $\hat{x}_{|L}$? Indeed, since V is (by design) unable to learn the field elements that were committed to, it cannot detect whether the function is a low-degree polynomial; then a cheating prover may commit to any $\alpha \neq x_j = \hat{x}_{|L}(0)$ as the claimed evaluation at 0, while committing to the correct evaluations elsewhere. The resulting function is not a low-degree polynomial anymore, but V is oblivious to this fact.

Therefore, we require a scheme that allows not only to commit to a function, but to also ensure it is a low-degree polynomial. We solve this problem by constructing an *algebraic* commitment protocol, whereby P commits to a set of field elements and can decommit to *any linear combination* of them. Then P may commit to $d+1$ points – which uniquely determine a degree-d polynomial g – and V requests a decommitment to the linear combination that coincides with $g(\rho)$ (see Figure 3.5). We next present the basic commitment protocol, and then extend it to be algebraic.

The basic protocol. Recall that our goal is to construct a commitment protocol between asymmetric parties, allowing a computationally unbounded P to send and later reveal a message $\alpha \in \mathbb{F}$ to a low-space verifier V. We focus on the first step, where P sends a hidden message, and deal with how to reveal it later. A natural



(a) Commitments to an interpolating set of $\hat{x}_{|L}$.

(b) Decommiting to a point outside the interpolating set.

Figure 3.5: Preventing leakage by committing to $\hat{x}_{|L}$ as an interpolating set for the polynomial. To decommit to an evaluation outside the set, the scheme must be algebraic.

attempt is to play the prover's strength against the verifier's weakness: we know, from the hardness of INDEX, that the space limitation of V prevents it from recalling an item from a long stream whose position is only revealed later; we can thus have P send a long stream y with the message hidden at a coordinate k that is revealed at the end.

While the idea seems intuitively sound, there are nontrivial issues to address. For example, the string-coordinate pair (y, k) should not have any structure from which V could extract information, which we can ensure by sampling both uniformly at random; but to prove security for this strategy, INDEX must be hard to solve *on average*. Luckily, reductions from one-way communication complexity enable us to prove this fact: one-way protocols where Alice receives $x \sim \{0,1\}^n$ and sends an s-bit message to Bob, who receives $j \sim [n]$ and attempts to output x_j, succeed with probability at most $\frac{1}{2} + O(\sqrt{s/n})$ [RY20]. We show that the bound extends to larger alphabets, carrying over to space-s streaming algorithms (see Proposition 6.1 and Lemma 6.2).

In short, we have P encode its message $\alpha \in \mathbb{F}$ *as the solution to a random* INDEX instance, exploiting the problem's average-case hardness to ensure that V is unable to extract α; more precisely, P sends a uniformly random string-coordinate pair (y, k) and then the "correction" $\gamma = \alpha - y_k$.[15] Of course, the discussion thus

[15]We remark that while replacing y_{ik} with α (rather than sending a random element and a

far only shows how P can commit; but we also need a decommitment protocol whereby V can check that P is being honest when it reveals β (which may or may not coincide with the message α). Fortunately, we already have a tool V can use to solve INDEX with an untrusted prover's assistance! The decommitment thus consists of an execution of pep by P and V with respect to the instance (y, k): this allows V to learn y_k and check that $\gamma + y_k = \beta$, i.e., that the correction γ sent earlier matches the (alleged) message.

Recall that we are building technical tools towards a zkSIP for INDEX, so we ultimately *exploit the hardness of a problem to solve an instance of the same problem.* Should we not expect, then, that the same leakage issues should arise with respect to the "virtual" instance (y, k) as they did with the "real" instance (x, j)? While this may appear to be circular reasoning, we stress that revealing evaluations of $\hat{y}$ leaks no information whatsoever about the input; indeed, (y, k) is a uniform random variable that is independent of (x, j). Put differently, V only obtains information about uniformly random strings that are completely uncorrelated with the input. See Section 6.2.2 for details.

Making the scheme algebraic. We now extend the foregoing idea into an *algebraic* protocol, which allows P to commit to a *tuple* of field elements $\boldsymbol{\alpha} = (\boldsymbol{\alpha}_1, \dots, \boldsymbol{\alpha}_\ell)$ and decommit to a linear combination $\boldsymbol{\alpha} \cdot \boldsymbol{\beta}$. (Committing to a polynomial and decommitting to an evaluation follows as a special case; see Section 6.2.1.) Note that such an extension seems to follow if linear combinations "commute" with commitments; that is, by showing that linear combinations of a fingerprint (as defined in Section 3.3.1) match a fingerprint of the linear combinations, we should be able to use essentially the same strategy of the basic scheme: committing with a random INDEX instance and decommitting with pep. Details follow.

Consider a trivial extension of the scheme that allows P to transmit a pair of messages $\alpha, \alpha' \in \mathbb{F}$: sending two independent commitments $(y, k, \alpha - y_k)$ and $(y', k', \alpha' - y'_{k'})$. The key observation is that, if V saves two fingerprints *at the same evaluation point* $\boldsymbol{\rho}$, then linear combinations and low-degree extensions do commute: for any $\beta, \beta' \in \mathbb{F}$, defining $z \coloneqq \beta y + \beta' y'$, we have $\hat{z}(\boldsymbol{\rho}) = \beta \hat{y}(\boldsymbol{\rho}) + \beta' \hat{y}'(\boldsymbol{\rho})$; in short, low-degree extending is a linear operation.

A problem still remains, however: since $k \neq k'$ with overwhelming probability, an execution of the pep protocol enables V to learn $z_k = \beta y_k + \beta' y'_k$; but the correction for y' refers to another coordinate $k' \neq k$ (with overwhelming probability). We address

correction later) looks simpler, then (y, k) ceases to be a random INDEX instance, and it is not clear how to show a reduction from INDEX.

this issue by *hiding both messages at the same index*, i.e., setting $k' = k$ and only revealing the coordinate after both y and y' are sent; see Section 6.2.3 for details.

3.3.3 From honest to malicious verifiers: temporal commitments

Recall that a source of leakage in the INDEX protocol of Section 3.3.1 is the prover P sending the restriction of $\hat{x}$ (the LDE of the input) to a line L in the clear. In the previous section, we constructed a prover-to-verifier scheme that enables P to commit to a low-degree polynomial and decommit to a single evaluation of it. We may then use it to modify the original protocol, having P instead *commit* to $\hat{x}_{|L}$ and decommit to the points inspected by V.

While this modification amounts to significant progress – indeed, it achieves an *honest-verifier* SIP for INDEX– there is a second major challenge to address. The issue is that *if a verifier* $\widetilde{V}$ *cheats*, it can use the protocol to extract information that it could not have learned on its own, as we will see next. The goal of this section is to describe a strategy that prevents leakage of information *without* requiring that $\widetilde{V}$ behave honestly; in other words, we would like to make the protocol *malicious-verifier* zero-knowledge.

Concretely, consider the (cheating) verifier $\widetilde{V}$ that ignores the input string x, reads j and requests the line through j and $j+1$ from the prover. P then commits to the restriction of $\hat{x}$ to this line and decommits to the evaluation of the LDE at both j and $j+1$. This reveals x_j and x_{j+1} to $\widetilde{V}$, which shows clearly that the modified protocol still leaks: x_j is *the only information the verifier should learn* that it could not have computed on its own, but the protocol also reveals x_{j+1} (which is just as hard to compute as the j^{th} coordinate).

An idealised scenario: V-to-P commitments. Let us assume, for the moment, that there also exists a commitment protocol in the reverse direction, allowing V to commit and later reveal a message to P. We will show how, in this idealised setting, we can prevent information leakage altogether. Note that the difficulty posed by a malicious verifier $\widetilde{V}$ is the usage of an allegedly random evaluation point $\boldsymbol{\rho}$ that is, in reality, a function of the input.

If $\widetilde{V}$ *proves that* $\boldsymbol{\rho}$ *is indeed random*, however, we may conclude that $\widetilde{V}$ could have computed $\hat{x}(\boldsymbol{\rho})$ alone – and thus that no leakage occurs. The idealised scheme allows $\widetilde{V}$ to do (almost) that, by having it commit to $\boldsymbol{\rho}$ *before reading the input stream* and decommit to it at a later step (after the prover's commitment). While this does not ensure $\boldsymbol{\rho}$ is random, the fact that $\widetilde{V}$ cannot decommit to anything other than $\boldsymbol{\rho}$ constrains its evaluation point to be *chosen before the input stream*, so that it cannot be a function of the input.

Of course, it is not at all clear that such a commitment protocol, allowing a weak computational party to commit to a computationally unbounded one, even exists; after all, the commitment step generally exploits their very difference to hide the message, as we did in the previous section. Is this just wishful thinking?

The solution: a temporal commitment. We will now see that, perhaps surprisingly, we can once again exploit the space limitation of $\widetilde{V}$ to accomplish this goal. What we obtain in fact falls short of a full-fledged commitment protocol: roughly speaking, the *temporal commitment* will enable a space-s verifier $\widetilde{V}$ to reveal not one, but s messages. But this collection is still determined before the input, so that it remains fit for purpose (incurring a small overhead in the simulator algorithm that we discuss in the following section).

As discussed above, we cannot expect $\widetilde{V}$ to be able to send a hidden message to P: however $\widetilde{V}$ may try to hide it, P can simply store the entirety of the communication and extract the message itself. Since sending is out of the picture, could $\widetilde{V}$ instead commit by *receiving* a message? Note that, while somewhat counter-intuitive, this would allow $\widetilde{V}$ to play what is essentially its only strength, its private randomness, against P. Recall, moreover, that there is a temporal aspect to the positions of a long stream z that $\widetilde{V}$ can remember: if it remembers z_i, this can be seen as evidence that *i was determined no later than when z was seen.*

Let us now make the idea more precise, and construct our verifier-to-prover temporal commitment protocol. The main idea is to impose some cost onto the ability of $\widetilde{V}$ to "unlock" the decommitment from P, without overly constraining the honest verifier V. Note that after P sends the commitment to a low-degree polynomial, having V reveal the point $\boldsymbol{\rho} = L(\rho)$ at which it computed $\hat{x}$ is not a problem (as opposed to revealing $\boldsymbol{\rho}$ *before* P sends the polynomial, which allows the prover to cheat easily). Therefore, we will have $\widetilde{V}$ reveal its alleged evaluation point $\boldsymbol{\rho}$ *along with a certificate* $c(\boldsymbol{\rho})$ that shows $\widetilde{V}$ selected the point before seeing the input stream. P will only proceed with the protocol if the certificate is valid; if not, it aborts to prevent $\widetilde{V}$ from learning information beyond its reach.

Given that the verifier's scarce resource is space, we design this certificate to require a number of bits that is not too large and yet not negligible; then the honest V should have no trouble, as it only needs to remember one piece of information, whereas the malicious $\widetilde{V}$ described before would need to store a certificate for the evaluation point $j+1$, which it does not know before reading x.

We thus prepend our INDEX protocol with a step where P sends $\widetilde{V}$ a long string z containing all possible evaluation points (i.e., the entire domain) of the low-degree extension $\hat{x}$.[16] Now, if $\widetilde{V}$ wants the prover, in the future, to decommit to a polynomial evaluation at the point $\boldsymbol{\rho}$, it must offer evidence that $\boldsymbol{\rho}$ is uncorrelated with the input stream: $\widetilde{V}$ does so by revealing $\boldsymbol{\rho}$ *along with the coordinate i that contains $\boldsymbol{\rho}$ in z*; i.e., the certificate for $\boldsymbol{\rho}$ is $c(\boldsymbol{\rho}) = i$, the coordinate satisfying $z_i = \boldsymbol{\rho}$.

The temporal commitment indeed achieves what we set out for: regardless of what $\widetilde{V}$ does, as long as its space is bounded we are able to extract the points it may ask P for *in advance of its streaming of x* (see Section 6.2.4). Note that the commitment is non-interactive (consisting of a single message from P to V) and need not be rerun if the verifier streams multiple inputs; we shall use it as the setup stage of our protocol. Its analysis is subtle and involved: it begins with a study of a variant of INDEX in the one-way communication model that we call RECONSTRUCT, where, upon receipt of a message from Alice, Bob outputs a guess for every coordinate of the input string rather than for only one. Using tools from information theory, we obtain an upper bound on the expected number of correct coordinates, which we call the protocol's *score*.

Next, we use the expected score bound of RECONSTRUCT to prove a related upper bound for a problem we call PAIR: a variant of INDEX where Bob, rather than receiving the coordinate to be recovered as part of the input, is free to choose it. The implication is that any protocol for PAIR has a small number C of indices such that the output of the protocol is outside C and yet correct (i.e., a pair (i, z_i) with $i \notin C$) with arbitrarily small probability. This will underpin the *simulator argument* that ultimately shows our protocol is zero-knowledge, which we sketch in the next section.

3.3.4 A sketch of the zero-knowledge INDEX protocol

We now have all of the components necessary to sketch a zero-knowledge streaming interactive proof for INDEX. Recall that we constructed a prover-to-verifier *algebraic* commitment protocol in Section 3.3.2 and a verifier-to-prover *temporal* commitment in Section 3.3.3. We will now compose them in the appropriate order, using the temporal commitment to constrain V to choose its inner randomness before reading the input stream; and the algebraic commitment to ensure P only reveals what the verifier needs. The protocol follows.

[16]In fact, any given point has a small probability of being absent from the string. We ignore this issue in the technical overview.

Parameters. Without loss of generality, we consider the alphabet over which the input string is defined to be a field of size $|\mathbb{F}| = q$; that is, $x \in \mathbb{F}^n$. We also fix two additional parameters, d and m, which characterise the low-degree extension $\hat{x} : \mathbb{F}^m \to \mathbb{F}$ as an m-variate polynomial of individual degree d. We assume all parameters are known to P and V in advance.

Setup: verifier-to-prover temporal commitment. P sends V a permutation of $\mathbb{F}^m$ as a string z (of length $v = q^m$). Before receiving the string, V samples $\boldsymbol{\rho} \sim \mathbb{F}^m$ and then streams z. When it sees $\boldsymbol{\rho}$ at the ℓ^{th} coordinate of z, the verifier stores ℓ.

Step 1: input streaming. V streams the input string x and records the fingerprint $\hat{x}(\boldsymbol{\rho})$ as well as the target index j.

Step 2: prover-to-verifier algebraic commitment. V samples $\rho \sim \mathbb{F}$ and sends P the line $L : \mathbb{F} \to \mathbb{F}^m$ through j and $\boldsymbol{\rho}$ (satisfying $L(0) = j$ and $L(\rho) = \boldsymbol{\rho}$).

P sends $x_j = \hat{x}_{|L}(0)$ (in the clear) and an algebraic commitment $(y, \boldsymbol{\gamma}, k)$ to the remainder of an interpolating set of the degree-dm polynomial $\hat{x}_{|L} : \mathbb{F} \to \mathbb{F}$, i.e., to the field elements $\hat{x}_{|L}(i)$ for all $i \in [dm]$. The commitment consists of a random matrix $y \sim \mathbb{F}^{dm \times p}$ with dm rows and a large enough number p of columns; a random (column) coordinate $k \sim [p]$; and the correction tuple $\boldsymbol{\gamma}$ satisfying $\boldsymbol{\gamma}_i = \hat{x}_{|L}(i) - y_{ik}$.

V computes the fingerprint $y(\boldsymbol{\sigma}, \boldsymbol{\beta}) = \sum_i \boldsymbol{\beta}_i \hat{y}_i(\boldsymbol{\sigma})$ after sampling (another) evaluation point $\boldsymbol{\sigma}$ and setting $\boldsymbol{\beta}$ as the tuple that satisfies $\sum_i \boldsymbol{\beta}_i \hat{x}_{|L}(i) = \hat{x}(\boldsymbol{\rho})$;[17] it also computes $\gamma = \sum_i \boldsymbol{\beta}_i \boldsymbol{\gamma}_i$ and stores k.

Step 3: temporal decommitment. V reveals its fingerprint's evaluation point $\boldsymbol{\rho}$ along with the index ℓ where it appeared in z. The prover checks that $z_\ell = \boldsymbol{\rho}$, and only continues to the final step if the check passes.

Step 4: algebraic decommitment. P and V engage in the decommitment of the k^{th} coordinate of the string $y' = \boldsymbol{\beta} \cdot y$ (the linear combination of the rows y_i with coefficients $\boldsymbol{\beta}_i$).[18] V outputs the (alleged) x_j if the decommitment is consistent with $\hat{x}(\boldsymbol{\rho})$, and rejects otherwise.

[17]Note that $\boldsymbol{\beta}_i$ is determined solely by i and ρ: it is the evaluation $\chi_i(\rho)$ of the i^{th} Lagrange polynomial.

[18]This requires P to know the linear coefficients $\boldsymbol{\beta}$, and, while we could have the verifier send them, this is not necessary: P learns $\boldsymbol{\rho}$ in step 3, which allows it to determine $\rho = L^{-1}(\boldsymbol{\rho})$ and thus $\boldsymbol{\beta} = \boldsymbol{\beta}(\rho)$ as well.

In an honest execution of the above protocol, the final decommitment reveals

$$\begin{aligned} y'_k &= \sum_i \boldsymbol{\beta}_i y_{ik} \\ &= \sum_i \boldsymbol{\beta}_i \big(\hat{x}_{|L}(i) - \boldsymbol{\gamma}_i\big) \\ &= \hat{x}(\boldsymbol{\rho}) - \gamma, \end{aligned}$$

so that V, having stored $\hat{x}(\boldsymbol{\rho})$ and γ, can indeed perform this consistency check (which shows the protocol is complete). The protocol's soundness follows from that of `pep`, noting that none of the mechanisms we add harm soundness (indeed, the last check relies, as does `pep`, on a random evaluation of the low-degree extension), while zero-knowledge, which we discuss next, follows from the correctness of our commitment protocols.

Proving the zero knowledge property. We conclude with a discussion of the simulator argument for the protocol laid out in this section. Recall that proving zero-knowledge for the foregoing protocol entails the construction of a *simulator* S, a streaming algorithm with knowledge of x_j and roughly the same memory as $\widetilde{V}$, which is able to interact with $\widetilde{V}$ without it being able to tell whether it is communicating with S or P.

Roughly speaking, S is does the following: after the temporal commitment step, it inspects the memory state of $\widetilde{V}$ and records (almost) all the points to which $\widetilde{V}$ can decommit; as shown in the last section, this is a relatively small set C. It then streams the input and records $\hat{x}(\boldsymbol{\rho})$ *for all* $\boldsymbol{\rho} \in C$.[19] Upon receipt of a line L from $\widetilde{V}$, the simulator computes and commits to an arbitrary low-degree polynomial g that interpolates across the points in $L \cap C$. When $\widetilde{V}$ requests the algebraic decommitment to obtain an evaluation of g, the simulator checks that the evaluation point ρ is contained in C (in which case $g(\rho)$ matches a fingerprint $\hat{x}(\boldsymbol{\rho})$ known to S), proceeds with the decommitment if that is the case, and otherwise aborts.

We note that implementing the strategy above raises yet another challenge, namely, extracting the set C of evaluation points from the description and memory state of $\widetilde{V}$. This is accomplished via a form of *white-box access* to $\widetilde{V}$, see Section 6.1.

The simulator S is thus able to generate the transcript of an interaction where the message $\hat{x}_{|L}$ of the algebraic commitment is replaced with another low-degree polynomial g whose evaluations match $\hat{x}_{|L}$ *at all points where* $\widetilde{V}$ *is able to temporally*

[19]We note that storing C is the most space-intensive task of S, which implies a small overhead to its space complexity as compared to $\widetilde{V}$; see Theorem 6.8.

decommit. Then, distinguishing between a real and a simulated transcript amounts to distinguishing an INDEX instance whose solution is $\hat{x}_{|L}$ from one whose solution is g.

We prove that any streaming algorithm that does so with nontrivial bias implies a one-way communication protocol for INDEX with a small message, contradicting the known hardness of the problem. We remark that the reduction is rather nontrivial, as we must insert an INDEX instance into the algebraic commitment $(y, \boldsymbol{\gamma}, k)$ while ensuring the decommitment can be simulated without any knowledge about the instance. See Theorem 6.8 for details.

Remark 3.1 (Superpolynomial to near-linear communication)**.** We stress that, while we may prove zero-knowledge with the strategy above, the natural reduction from INDEX is over a large alphabet $\Gamma = \mathbb{F}^{dm}$. But then, for indistinguishability to follow, the length p of the temporal commitment must be q^{dm}, which implies *superpolynomial* communication complexity.

We avoid this blowup via Lemma 6.2, which shows that an INDEX (one-way) protocol for large alphabets implies another protocol for the binary alphabet with only a mild loss to its success probability; this restricts our ambient field to be an extension of $\mathbb{F}_2$, but reduces the superpolynomial complexity to *barely superlinear*.

3.3.5 A general-purpose zero-knowledge SIP: sumcheck

Lastly, we briefly mention how the commitment protocols developed in Sections 3.3.2 and 3.3.3 can be used not only to solve INDEX (and, more generally, the polynomial evaluation problem), but also to construct another widely applicable tool: a streaming zero-knowledge *sumcheck* protocol.

As before, we start with an SIP that is clearly not zero-knowledge: the standard sumcheck protocol leaks hard-to-compute sums over subcubes. By carefully using the algebraic and temporal commitment protocols, we can also endow the sumcheck protocol with zero-knowledge in the data stream model. However, we note that doing so is considerably more involved than in the case of INDEX, owing to, among other reasons, several rounds of interaction with nontrivial dependencies of messages on past communication.

More precisely, we consider a slight variation of the standard sumcheck protocol: while in the latter every round is followed by a (random) consistency check, we instead defer all such checks to the end. It is clear that this variant is equivalent to the standard protocol; however, without the modification, the zero-knowledge property seems to require a strengthening of the chained commit-decommit strategy we follow. Moreover, rather than a single algebraic commitment followed by a (single) decommitment, the sumcheck protocol requires many decommitments; indeed, for an m-variate polynomial f, the prover commits to m partial sums of f, and each partial sum is involved in two decommitments (for a total of $m+1$ decommitments).

Therefore, by extending the techniques that underpin our approach for the INDEX problem to a *multi-round* setting, we are able to construct a zero-knowledge sumcheck SIP. Such a protocol can then be used to compute frequency moments and inner products, problems known to require linear space without a prover's assistance [AMS99]. See Section 6.4 for details.

Chapter 4

A structural theorem for local algorithms

Overview

We prove a general structural theorem for a wide family of local algorithms, which includes property testers, local decoders, and PCPs of proximity. Namely, we show that the structure of every algorithm that makes q adaptive queries and satisfies a natural robustness condition admits a sample-based algorithm with $n^{1-1/O(q^2 \log^2 q)}$ sample complexity, following the definition of Goldreich and Ron (TOCT 2016). We prove that this transformation is nearly optimal. Our theorem also admits a scheme for constructing privacy-preserving local algorithms.

Using the unified view that our structural theorem provides, we obtain results regarding various types of local algorithms, including the following.

- We strengthen the state-of-the-art lower bound for relaxed locally decodable codes, obtaining an *exponential* improvement on the dependency in query complexity; this resolves an open problem raised by Gur and Lachish (SICOMP 2021).
- We show that any (constant-query) testable property admits a sample-based tester with sublinear sample complexity; this resolves a problem left open in a work of Fischer, Lachish, and Vasudev (FOCS 2015), bypassing an exponential blowup caused by previous techniques in the case of adaptive testers.
- We prove that the known separation between proofs of proximity and testers is essentially maximal; this resolves a problem left open by Gur and Rothblum (ECCC 2013, Computational Complexity 2018) regarding sublinear-time delegation of computation.

Our techniques strongly rely on relaxed sunflower lemmas and the Hajnal-Szemerédi theorem.

Organisation

This chapter is organised as follows. In Section 4.1, we briefly discuss preliminaries for the technical sections (besides those in Chapter 2). In Section 4.2, we present our definition of robust local algorithms and show how to cast various types of algorithms in this framework. In Section 4.3, we provide an arsenal of technical tools, including relaxed sunflower lemmas and a sampling lemma that builds on the Hajnal–Szemerédi theorem. In Section 4.4, we use the foregoing tools to prove Theorem 1. Finally, in Section 4.5, we derive our applications to coding theory, property testing, and proofs of proximity.

4.1 Preliminaries

The description of adaptive algorithms in terms of decision trees provided in Section 2.3.1 is complete; however, we choose to use an alternative that is amenable to *daisy lemmas* (see Section 4.3.1). This is obtained by describing each decision tree with the collection of its branches. From $\{(T_s, s) : s \in \{0,1\}^r\}$ we construct $\{(S_{st}, a_{st}, b_{st}, s, t) : s \in \{0,1\}^r, t \in [|\Gamma|^q]\}$, where t identifies which branch the tuple is obtained from. S_{st} is the q-set queried by the t^{th} branch of T_s, while a_{st} is the assignment to S_{st} defined by the edges of this branch and $b_{st} \in \{0,1\}$ is the output at its leaf. We remark that the decision trees may be reconstructed from their branches, so that this is description is indeed equivalent (though we will not need this fact).

Nonstandard notation. We shall use the following notation to study local algorithms (in Chapter 4). An *assignment* to a set S (over alphabet Γ) is a function $a\colon S \to \Gamma$, which may be equivalently seen as a vector in $\Gamma^{|S|}$ whose coordinates correspond to elements of S in increasing order. Its restriction to $P \subseteq S$ is denoted $a_{|P}$. If x is an assignment to S and κ is an assignment to $P \subseteq S$, the *partially replaced assignment* $x_\kappa \in \Gamma^n$ is that which coincides with κ in P and with x in $S \setminus P$ (i.e., $x_{\kappa|P} = \kappa$ and $x_{\kappa|S\setminus P} = x_{S\setminus P}$).

We now define the distribution of an algorithm, as well as its distribution under a fixed input.

Definition 4.1 (Induced distribution)**.** *Let M be a q-local algorithm with randomness complexity r described by the collection of decision trees $\{T_s : s \in \{0,1\}^r\}$. The* distribution $\tilde{\mu}^M$ *of M is given by sampling $s \in \{0,1\}^r$ uniformly at random and taking T_s.*

Fix an arbitrary input x to M and, for all $s \in \{0,1\}^r$, let $(S_{st}, x_{|S_{st}}, b_{st}, s, t)$ be the unique tuple defined by the branch of T_s followed on input x. We may thus *discard t and the tuple $(S_s, x_{|S_s}, b_s, s)$ is well defined. The distribution μ_x^M is given by sampling $s \in \{0,1\}^r$ uniformly at random and taking the set S_s (the first element of the tuple $(S_s, x_{|S_s}, b_s, s)$).*

We note that the contents of the tuple $(S_s, x_{|S_s}, b_s, s)$ describe exactly how M will behave on input x and random string s.

4.2 Robust local algorithms

We now formally introduce *robust local algorithms*, which capture a wide class of sublinear algorithms, ranging from *property testing* to *locally decodable codes.* Our main result (Theorem 1) holds for any robust local algorithm, and indeed, we obtain our results for coding theory, testing, and proofs of proximity as direct corollaries.

While local algorithms are very well studied, their definition is typically context-dependent, where they are required to perform different tasks (e.g., test, self-correct, decode, perform a local computation) under different promises (e.g., proximity to encoded inputs, being either "close" to or "far" from sets). However, structured promises on the input are (with the exception of degenerate cases) necessary for algorithms that only make a sublinear number of queries. This feature leads naturally to the notion of robustness, which, loosely speaking, a local algorithm satisfies if its output is stable under small perturbations.

In the next subsection, we provide a precise definition of robust local algorithms. Then, in the subsequent subsections, we show how this notion captures property testing, locally testable codes, locally decodable and correctable codes, PCPs of proximity, and other local algorithms.

4.2.1 Definition

We begin by defining *local algorithms*, which are probabilistic algorithms that receive query access to an input x and explicit parameter z and are required to compute a partial function $f(z, x)$ (which represents a promise problem) by only making a small number of queries to the input x.

Definition 4.2 (Local algorithms)**.** *Let Γ be a finite alphabet, Z a finite set and $\{\mathcal{P}_z : z \in Z\}$ a family of sets $\mathcal{P}_z \subseteq \Gamma^n$ indexed by Z. Let $\mathcal{P} := \{(z, x) : z \in Z, x \in \mathcal{P}_z\}$ and $f \colon \mathcal{P} \to \{0,1\}$ be a partial function.*[1] *A q-*local algorithm *M* for computing *f*

[1]We remark that allowing only rectangles $\mathcal{P} = Z \times \mathcal{Q}$ as the domain of f suffices for most of our applications (e.g., testers and local decoders), but not all. For example, in a MAP for a property Π, there may be inputs $x \in \Pi$ that are only contained in $\mathcal{P}_z$ for a single $z \in Z$. (See Section 4.5.3.)

with error rate σ *receives explicit access to* $z \in Z$*, query access to* $x \in \mathcal{P}_z$*, makes at most* q *queries to* x *and satisfies*

$$\Pr[M^x(z) = f(z,x)] \geq 1 - \sigma.$$

The parameter q is called the *query complexity* of M (recall Section 2.3.1), to which we also refer as *locality*. Throughout, when we refer to a *local algorithm*, we mean a q-local algorithm with $q = o(n)$. Another important parameter is the *randomness complexity* of M, defined as the maximal number of coin tosses it makes over all $(z,x) \in Z \times \Gamma^n$ (note that an execution $M^x(z)$ is well-defined even if $x \notin \mathcal{P}_z$).

The following definition formalises the aforementioned natural notion of *robustness*, which is the structural property that underlies local computation.

Definition 4.3 (Robustness)**.** *Let* $\rho > 0$*. A local algorithm* M *for computing* $f \colon \mathcal{P} \to \{0,1\}$ *is* ρ-robust *at the point* $(z,x) \in \mathcal{P}$ *if* $\Pr[M^w(z) = f(z,x)] \geq 1 - \sigma$ *for all* $w \in B_\rho(x)$*. We say that* M *is* (ρ_0, ρ_1)-robust *if, for all* $z \in Z$ *and* $b \in \{0,1\}$*,* M *is* ρ_b*-robust at every* x *such that* $f(z,x) = b$*.*

If a local algorithm M is (ρ_0, ρ_1)-robust and $\max\{\rho_0, \rho_1\} = \Omega(1)$ (a constant independent of n), we simply call M *robust*. Note that non-trivial robustness is only possible because f is a partial function; that is, the local algorithm M solves a *promise problem* where, for every parameter z, the algorithm is promised to receive an input from $\mathcal{P}_{z,0} := f^{-1}(z,0)$ on which it should output 0, or an input from $\mathcal{P}_{z,1} := f^{-1}(z,1)$ on which it should output 1.

Remark 4.1 (One-sided robustness)**.** For our main result (Theorem 1), it suffices to have *one-sided robustness*, i.e., (ρ_0, ρ_1)-robustness where only one of ρ_0, ρ_1 is non-zero. For example, in the setting of property testing with proximity parameter ε we only have $(\varepsilon, 0)$-robustness (see Section 4.2.2 for details). To simplify notation, we refer to $(\rho, 0)$-robust local algorithms as ρ-robust.

Remark 4.2 (Larger alphabets)**.** The definition of local algorithms can be further generalised to a constant-size output alphabet Σ, in which case the partial function is $f \colon \Gamma^n \to \Sigma$; we assume $\Sigma = \{0,1\}$ for simplicity of exposition, but note that our results extend to larger output alphabets in a straightforward manner.

We proceed to show how to capture various well-studied families of sublinear algorithms (such as testers, local decoders, and PCPs) using the notion of robustness.

4.2.2 Property testing

Recall, from Definition 2.1 (and the following discussion in Section 2.4) that property testers are probabilistic algorithms that solve approximate decision problems by making a small number of queries to their input. We restate the definition below.

Definition 4.4 (Testers)**.** *An ε-*tester *with* error rate σ *for a* property $\Pi \subseteq \Gamma^n$ *is a probabilistic algorithm T that receives query access to a string $x \in \Gamma^n$. The tester T performs at most $q = q(\varepsilon, n)$ queries to x and satisfies the following two conditions.*

1. *If $x \in \Pi$, then* $\Pr[T^x = 1] \geq 1 - \sigma$.
2. *For every x that is ε-far from Π (i.e., $x \in \overline{B_\varepsilon(\Pi)}$), then* $\Pr[T^x = 0] \geq 1 - \sigma$.

We are interested in the regime where $\varepsilon = \Omega(1)$ (i.e., ε is a fixed constant independent of n), and assume it to be the case in the remainder of this discussion.

Note that testers are *not* robust with respect to inputs in the property Π, as changing a single coordinate of an input $x \in \Pi$ could potentially lead to an input outside Π. Moreover, an ε-tester does not immediately satisfy one-sided robustness, as inputs that are on the boundary of the ε-neighbourhood of Π are not robust (see figure Figure 1.1b).

However, by increasing the value of the proximity parameter by a factor of 2, we can guarantee that every point that is 2ε-far from Π satisfies the robustness condition. The following claim formalises this statement and shows that testers can be cast as robust local algorithms.

Claim 4.1. *An ε-tester T for property $\Pi \subseteq \Gamma^n$ is an $(\varepsilon, 0)$-robust local algorithm, with the same parameters, for computing the function f defined as follows.*

$$f(x) = \begin{cases} 1, & \text{if } x \in \Pi \\ 0, & \text{if } x \text{ is } 2\varepsilon\text{-far from } \Pi. \end{cases}$$

Proof. By definition, the tester T is a local algorithm for computing f; denote its error rate by σ. We show it satisfies (one-sided) robustness with respect to f. Let $x \in \Gamma^n$ be an input that is 2ε-far from Π, and consider $y \in B_\varepsilon(x)$. By the triangle inequality, we have that y is ε-far from Π. Thus, $\Pr[T^y = 0] \geq 1 - \sigma$, and so T is an $(\varepsilon, 0)$-robust local algorithm for f. □

Remark 4.3 (Robustness vs proximity tradeoff)**.** The notion of a tester with proximity parameter ε and that of an ε-robust tester with proximity parameter 2ε coincide. Moreover, there is a tradeoff between the size of the promise captured by the partial function f and the robustness parameter ρ: taking any $\varepsilon' > \varepsilon$, the tester T is a ρ-robust local algorithm with $\rho = \varepsilon' - \varepsilon$ for computing the function

$$f(x) = \begin{cases} 1, & \text{if } x \in \Pi \\ 0, & \text{if } x \text{ is } \varepsilon'\text{-far from } \Pi. \end{cases}$$

As ε' increases, the robustness parameter ρ increases and the size of the domain of definition of f decreases. In particular, taking $\varepsilon' = \varepsilon$ makes T a $(0, 0)$-robust algorithm (i.e., an algorithm that is not robust).

4.2.3 Local codes

We consider error-correcting codes that admit local algorithms for various tasks, such as testing, decoding, correcting, and computing functions of the message. Recall, from Section 2.5, that a *code* $C\colon \{0,1\}^k \to \{0,1\}^n$ is an injective mapping from *messages* of length k to codewords of *blocklength* n, whose *rate* is defined as k/n and whose *relative distance* is the minimum, over all distinct messages $x, y \in \Gamma^k$, of $\Delta(C(x), C(y))$. Note that hereafter we focus on *binary* codes, but remind that the extension to larger alphabets is straightforward. In the following, we show how to cast the prominent notions of local codes as robust local algorithms.

4.2.3.1 Locally testable codes

Locally testable codes (LTCs) [GS06] are codes that admit algorithms that distinguish codewords from strings that are far from being valid codewords, using a small number of queries.

Definition 4.5 (Locally Testable Codes (LTCs))**.** *A code $C\colon \{0,1\}^k \to \{0,1\}^n$ is locally testable, with respect to proximity parameter ε and error rate σ, if there exists a probabilistic algorithm T that makes q queries to a purported codeword w such that:*

1. *If $w = C(x)$ for some $x \in \{0,1\}^k$, then* $\Pr[T^w = 1] \geq 1 - \sigma$.
2. *For every w that is ε-far from C, we have* $\Pr[T^x = 0] \geq 1 - \sigma$.

Note that the algorithm T that an LTC admits is simply an ε-tester for the property of being a valid codeword of C. Thus, by Claim 4.1, we can directly cast T as a robust local algorithm.

4.2.3.2 Locally decodable and correctable codes

Locally decodable codes (LDCs) [KT00] are codes that admit algorithms for decoding each individual bit of the message of a moderately corrupted codeword by only making a small number of queries to it. We recall their formal definition below.

Definition 4.6 (Definition 5.3, restated). *A code $C\colon \{0,1\}^k \to \{0,1\}^n$ is locally decodable with* decoding radius δ *and error rate σ if there exists a probabilistic algorithm D that, given index $i \in [k]$, makes q queries to a string w promised to be δ-close to a codeword $C(x)$, and satisfies*

$$\Pr[D^w(i) = x_i] \geq 1 - \sigma.$$

Note that local decoders are significantly different from local testers and testing in general. Firstly, decoders are given a promise that their input is *close* to a valid codeword (whereas testers are promised to either receive a perfectly valid input, or one that is far from being valid). Secondly, a decoder is given an index as an explicit parameter and is required to perform a different task (decode a different bit) for each parameter (see Figure 1.1a).

Nevertheless, local decoders can also be cast as robust local algorithms. In fact, unlike testers, they satisfy *two-sided* robustness (i.e., both 0-inputs and 1-inputs are robust). In the following, note that since inputs near the boundary of the decoding radius are *not* robust, we reduce the decoding radius by a factor of 2.

Claim 4.2. *A local decoder D with decoding radius δ for the code $C\colon \{0,1\}^k \to \{0,1\}^n$ is a $(\delta/2, \delta/2)$-robust local algorithm for computing the function f defined as follows.*

$$f(z, w) = x_z, \text{ if } x \in \{0,1\}^k \text{ is such that } w \text{ is } \delta/2\text{-close to } C(x).$$

Proof. Take any $w \in \{0,1\}^n$ that is $\delta/2$-close to $C(x)$. Then, (w, z) is in the domain of definition of f for all explicit inputs $z \in [k]$. Now let $w' \in B_{\delta/2}(w)$ and note that w' is still within the decoding radius of D. Hence, the decoder D outputs x_z with probability $1 - \sigma$, as required. Moreover, this holds regardless whether $x_z = 0$ or $x_z = 1$, and so D is $(\delta/2, \delta/2)$-robust. □

Remark 4.4 (Robustness vs decoding radius tradeoff). A local decoder has decoding radius δ if and only if it is $\delta/2$-robust with decoding radius $\delta/2$, and a tradeoff between promise size and robustness parameter likewise holds in this case: for any $\delta' < \delta$, the decoder D is a $(\delta - \delta', \delta - \delta')$-robust algorithm for the restriction of f to the δ'-neighbourhood of the code C. In particular, D is a (δ, δ)-robust algorithm with the domain of f defined to be the code C.

Locally correctable codes. The notion of locally correctable codes (LCCs) is closely related to that of LDCs, except that rather than admitting an algorithm that can decode any individual *message* bit, LCCs admit an algorithm that can correct any corrupted *codeword* bit of a moderately corrupted codeword.

Definition 4.7 (Locally Correctable Codes (LCCs))**.** *A code $C\colon \{0,1\}^k \to \{0,1\}^n$ is locally correctable with* correcting radius δ *and error rate σ if there exists a probabilistic algorithm D that, given index $j \in [n]$, makes q queries to a string w promised to be δ-close to a codeword $C(x)$ and satisfies*

$$\Pr[D^w(j) = C(x)_j] \geq 1 - \sigma.$$

A straightforward adaptation of Claim 4.2 yields the following claim.

Claim 4.3. *A local corrector D with correcting radius δ for the code $C\colon \{0,1\}^k \to \{0,1\}^n$ is a $(\delta/2, \delta/2)$-robust local algorithm for computing the function f defined as follows.*

$$f(z, w) = C(x)_z, \text{ if } x \in \{0,1\}^k \text{ is such that } w \text{ is } \delta/2\text{-close to } C(x).$$

4.2.3.3 Relaxed locally decodable codes

Relaxed locally decodable codes (relaxed LDCs) [BGH+06] are codes that admit a natural relaxation of the notion of local decoding, in which the decoder is allowed to output a special abort symbol $\perp$ on a small fraction of indices, indicating it detected an inconsistency, but never erring with high probability.

Definition 4.8 (Definition 5.4, restated)**.** *A code $C\colon \{0,1\}^k \to \{0,1\}^n$ whose distance is δ_C is a q-local relaxed LDC with success rate ρ, decoding radius $\delta \in (0, \delta_C/2)$ and error rate $\sigma \in (0, 1/3]$ if there exists a randomised algorithm D, known as a* relaxed decoder*, that, on input $i \in [k]$, makes at most q queries to an oracle w and satisfies the following conditions.*

1. *Completeness: For any $i \in [k]$ and $w = C(x)$, where $x \in \{0,1\}^k$,*

$$\Pr[D^w(i) = x_i] \geq 1 - \sigma\ .$$

2. *Relaxed Decoding: For any $i \in [k]$ and $w \in \{0,1\}^n$ that is δ-close to a (unique) codeword $C(x)$,*

$$\Pr[D^w(i) \in \{x_i, \perp\}] \geq 1 - \sigma\ .$$

3. *Success Rate: There exists a constant $\xi > 0$ such that, for any $w \in \{0,1\}^n$ that is δ-close to a codeword $C(x)$, there exists a set $I_w \subseteq [k]$ of size at least ξk such*

that for every $i \in I_w$,

$$\Pr[D^w(i) = x_i] \geq 2/3 \ .$$

Note that strictly speaking, the special abort symbol makes it so that relaxed local decoders do not fully fit Definition 4.2, as the input-output mapping f becomes one-to-many. Nevertheless, a simple generalisation of local algorithms, which allows an additional abort symbol, enables us to capture relaxed LDCs as robust local algorithms as well. We show this in Section 4.5.2.

4.2.3.4 Universal locally testable codes

Universal locally testable codes (universal LTCs) [GG18] are codes that admit local tests for membership in numerous possible subcodes, allowing for testing properties of the encoded message.

Definition 4.9 (Universal LTCs)**.** *A universal LTC $C\colon \{0,1\}^k \to \{0,1\}^n$ for a family of functions $\mathcal{F} = \left\{f_i : \{0,1\}^k \to \{0,1\}\right\}_{i\in[M]}$ is a code such that for every $i \in [M]$ the subcode $\{C(x) \ : \ f_i(x) = 1\}$ is locally testable.*

Note that ULTCs trivially generalise LTCs, as well as generalise relaxed LDCs (see details in [GG18, Appendix A]). Since universal testers can be viewed as algorithms that receive an explicit parameter $i \in [M]$ and invoke an ε-tester for the property $\{C(x) \ : \ f_i(x) = 1\}$, then by applying Claim 4.1 to each value of the parameter i they can be cast as robust local algorithms.

4.2.4 PCPs of proximity

PCPs of proximity (PCPPs) [BGH+06] are probabilistically checkable proofs wherein the verifier is given query access not only to the proof, but also to the input. The PCPP verifier is then required to probabilistically check whether the statement is correct by only making a constant number of queries to both input and proof.

Definition 4.10. *A PCP of proximity (PCPP) for a language L with proximity parameter ε, error rate σ and query complexity q, consists of a probabilistic algorithm V, called the verifier, that receives query access to* both *an input $x \in \Gamma^n$ and a proof $\pi \in \{0,1\}^m$. The verifier V is allowed to make q queries to (x, π) and satisfies the following:*

1. *for every $x \in L$ there exists a proof π such that* $\Pr\left[V^{(x,\pi)} = 1\right] \geq 1 - \sigma$*; and*
2. *for every x that is ε-far from L and every proof π, it holds that* $\Pr\left[V^{(x,\pi)} = 0\right] \geq 1 - \sigma$.

We observe that PCPs of proximity with canonical proofs [GS06] (i.e., such that the verifier rejects statement-proof pairs that are far from being the concatenation of a valid statement with a valid proof for it) admit verifiers that are robust local algorithms. Using the tools of [DGG19], who show that PCPPs can be endowed with the canonicity property at the cost of polynomial blowup in proof length, we can obtain robust local algorithms for general PCPPs.

Claim 4.4. *A PCPP for a language $L \subseteq \Gamma^n$ with proximity parameter $\varepsilon > 0$, error rate σ and query complexity q can be transformed into a PCPP for L with proximity parameter 2ε, whose verifier is an $(\varepsilon, 0)$-robust local algorithm with the same query complexity and error rate.*

Sketch of proof. Let V be a PCPP verifier with proximity parameter ε and error rate σ for $L \subseteq \Gamma^n$, that makes at most q queries to its input-proof pair $(x, \pi) \in \Gamma^n \times \{0,1\}^m$. By [DGG19, Section 3], there exists a PCPP verifier V' for L with $\mathrm{poly}(m)$ proof length (as well as proximity parameter ε, error rate σ and query complexity q) that satisfies the following strengthening of the conditions in Definition 4.10: there is a set of *canonical proofs* $\{\pi_x\}_{x \in L}$ such that

1. for every $x \in L$, it holds that $\Pr\left[V^{(x,\pi_x)} = 1\right] \geq 1 - \sigma$; and
2. if (x, π) is ε-far from (y, π_y) for all $y \in L$, it holds that $\Pr\left[V^{(x,\pi)} = 0\right] \geq 1 - \sigma$.

In other words, V' is an ε-tester for the property $\Pi \coloneqq \{(x, \pi_x) : x \in L\}$, and we invoke Claim 4.1. □

Non-interactive proofs of proximity. MA proofs of proximity (MAPs) [GR18, FGL14] are proof systems that can be viewed as a property testing analogue of NP proofs. The setting of MAPs is very similar to that of PCPPs, with the distinction that the purported proof is of sublinear size and is given explicitly, i.e., the MAP verifier can read the entire proof. With the equivalent description of a MAP as a covering by partial testers (Claim 5.1), *every fixed proof string* defines a tester, and Claim 4.1 applies. We cover this in Section 4.5.3.

4.3 Technical lemmas

In the section we provide an arsenal of technical tools for analysing robust local algorithms, which we will then use to prove our main result in Section 4.4. The order in which we present the tools is according to their importance, starting with the most central lemmas.

Specifically, in Section 4.3.1 we discuss the notion of relaxed sunflowers that we shall need, called daisies, then state and prove a daisy partition lemma for multi-collections of sets. In Section 4.3.2, we apply the Hajnal-Szemerédi theorem to derive a sampling lemma for daisies. In Section 4.3.3, we prove a simple yet vital volume lemma for robust local algorithms, which will be used throughout our analysis. Finally, in Section 4.3.4 we adapt generic transformations (amplification and randomness reduction) to our setting of robust local algorithms.

4.3.1 Relaxed sunflowers

We discuss the central technical tool used in the transformation to sample-based algorithms, which is a relaxation of combinatorial sunflowers, referred to as *daisies* [FLV15, GL21]. We extend the definition of daisies to multi-sets, then state and prove the particular variant of a daisy lemma that we shall need.

Definition 4.11 (Daisy)**.** *Suppose $\mathcal{S}$ is a multi-collection of subsets of $[n]$ (i.e., subsets may repeat). $\mathcal{S}$ is an h-daisy (where $h\colon \mathbb{N} \to \mathbb{N}$) with petals of size j and* kernel $K \subseteq [n]$ *if the following holds: every $S \in \mathcal{S}$ has a* petal $S \setminus K$ *with $|S \setminus K| = j$ and, for every $k \in [j]$, there exists a subset $P_k \subseteq S \setminus K$ with $|P_k| \geq k$ whose elements are contained in at most $h(k)$ sets from $\mathcal{S}$.*
A daisy with pairwise disjoint petals (1-daisy) is referred to as a simple daisy*.*

We remark that the notion of a daisy relaxes the standard definition of a sunflower in two ways: (1) the kernel is not required to equal the pairwise intersection of all sets in the collection, its structure is unconstrained; and (2) the *petals* $\mathcal{P} = \{S \setminus K : S \in \mathcal{D}\}$ need not be pairwise disjoint, but rather, each point outside of the kernel can be contained in at most $h(j)$ sets of $\mathcal{D}$; see Figure 3.1b. Note that Definition 4.11, in contrast to sunflowers (for which pairwise disjointness disallows multiple copies of a same set), applies to *multi-sets*.

These relaxations, unlike in the case of sunflowers, allow us to arbitrarily partition any collection of subsets into a collection of daisies with strong structural properties, as Lemma 4.1 shows.

Lemma 4.1 (Daisy partition lemma for multi-collections)**.** *Let $\mathcal{S}$ be a multi-collection of q-sets of $[n]$, and define the function $h\colon \mathbb{N} \to \mathbb{N}$ as follows:*

$$h(k) = n^{\frac{\max\{1,k-1\}}{q}} .$$

Then, there exists a collection $\{\mathcal{D}_j : 0 \leq j \leq q\}$ such that

1. *$\{\mathcal{D}_j\}$ is a partition of $\mathcal{S}$, i.e., $\bigcup_{j=0}^{q} \mathcal{D}_j = \mathcal{S}$ and $\mathcal{D}_j \cap \mathcal{D}_k = \emptyset$ when $j \neq k$.*

2. *For every $0 \leq j \leq q$, there exists a set $K_j \subseteq [n]$ of size $|K_j| \leq q|\mathcal{S}| \cdot n^{-\max\{1,j\}/q}$ such that $\mathcal{D}_j$ is an h-daisy with kernel K_j and petals of size j. Moreover, the kernels form an incidence chain $\varnothing = K_q \subseteq K_{q-1} \subseteq \cdots \subseteq K_1 = K_0$.*

Proof. We construct the collections $\{\mathcal{D}_j : 0 \leq j \leq q\}$ in a greedy iterative manner, as follows.

1. Define $\mathcal{S}_0 \coloneqq \mathcal{S}$.
2. Inductively define, for each $0 \leq j \leq q-1$:
   (a) *Kernel construction:* Define K_j as the set of points in $[n]$ that are contained in at least $h(j+1)$ sets from $\mathcal{S}$.
   (b) *Daisy construction:* Set $\mathcal{D}_j$ to be all the sets $S \in \mathcal{S}_j$ such that $|S \setminus K_j| = j$.
   (c) Set $\mathcal{S}_{j+1}$ to be $\mathcal{S}_j \setminus \mathcal{D}_j$.
3. Finally, set $\mathcal{D}_q = \mathcal{S}_q$ and $K_q = \emptyset$.

We now prove that this construction yields daisies with the required properties. For ease of notation, let d_i be the number of sets in $\mathcal{S}$ containing i for each $i \in [n]$. By definition, $\mathcal{S}_q \subseteq \mathcal{S}_{q-1} \subseteq \cdots \subseteq \mathcal{S}_0$ and $\mathcal{D}_j \subseteq \mathcal{S}_j$ for all j. Since $\mathcal{D}_{j-1} \cap \mathcal{S}_j = \varnothing$ for all $j \in [q]$, it follows that $\mathcal{D}_j \cap \mathcal{D}_k = \varnothing$ when $j \neq k$. Also, since $\mathcal{S} = \mathcal{S}_q \cup \bigcup_{0 \leq j < q} \mathcal{D}_j$ and $\mathcal{S}_q = \mathcal{D}_q$, we have $\mathcal{S} = \bigcup_{0 \leq j \leq q} \mathcal{D}_j$.

Since $\mathcal{S}$ is comprised of q-sets,

$$\sum_{i \in [n]} d_i = q|\mathcal{S}|.$$

By the kernel construction, for each $j \in \{0, 1, \ldots, q\}$, K_j is the set of all $i \in [n]$ such that $d_i \geq h(j+1)$, which implies $\sum_{i \in K_j} d_i \geq |K_j| \cdot h(j+1)$. Therefore,

$$q|\mathcal{S}| = \sum_{i \in [n]} d_i \geq \sum_{i \in K_j} d_i \geq |K_j| \cdot h(j+1)$$

and thus $|K_j| \leq q|\mathcal{S}|/h(j+1) = q|\mathcal{S}| \cdot n^{-\max\{1,j\}/q}$. Note, also, that the kernel construction ensures not only $K_j \subseteq K_{j-1}$ when $j \in [q]$, but also $K_0 = K_1$ because $h(1) = h(2)$.

Since the petals of each $S \in \mathcal{D}_j$ have size exactly j by construction, all that remains to be proven is the intersection condition on the petals that makes $\mathcal{D}_j$ an h-daisy; namely, that for every $k \in [j]$, there exists a subset $P_k \subseteq S \setminus K$ with $|P_k| \geq k$ whose elements are contained in at most $h(k)$ sets from $\mathcal{D}_j$. Assume for the sake of contradiction that this condition does not hold.

Let $j \in [q]$ be the smallest value for which $\mathcal{D}_j$ is not an h-daisy and $S \in \mathcal{D}_j$ be a set that violates the intersection condition; then take $k \leq j$ to be the smallest subset size such that every $P_k \subseteq S \setminus K_j$ with size k has an element $i \in P_k$ with $d_i > h(k)$ (equivalently said, j and k are minimal such that the subset $L \subset S \setminus K_j$, comprised of all i with $d_i \leq h(k)$, has size at most $k-1$).

Suppose first that $k = 1$. Then, every $i \in S \setminus K_j$ is such that $d_i > h(2) = h(1)$ and thus $S \setminus K_j \subseteq K_0$. But this implies $S \in \mathcal{D}_0$ (since $|S \setminus K_0| = 0$), a contradiction because the intersection condition holds by vacuity on empty petals.

Suppose now that $k > 1$. The subset

$$L \coloneqq \{i \in S \setminus K_j : d_i \leq h(k)\}$$

contains at most $k-1$ points; however, by minimality of k, at least $k-1$ distinct points $i \in L$ satisfy $d_i \leq h(k-1) \leq h(k)$. Therefore, $|L| = k-1$ and

$$L = \{i \in S \setminus K_j : d_i \leq h(k-1)\} \ .$$

By the definition of L, every $i \in S \setminus (K_j \cup L)$ satisfies $d_i > h(k)$, so that $i \in K_{k-1}$; therefore, $S \subseteq K_{k-1} \cup K_j \cup L$. Since the kernels form an incidence chain, $K_j \subseteq K_{k-1}$ and thus $S \setminus K_{k-1} = L$. But then $|S \setminus K_{k-1}| = |L| = k-1$, so that $S \in \mathcal{D}_{k-1}$, contradicting the fact that $S \in \mathcal{D}_j$ (because $k-1 < j$ and $\{\mathcal{D}_j\}$ is a partition). $\square$

The following claim shows an upper bound on the total number of sets in an h-daisy that may intersect a given petal. It will be useful in order to partition a daisy into *simple* daisies, as the next section will show.

Claim 4.5. *Let $\mathcal{S}$ be a multi-collection of q-sets and $\{\mathcal{D}_j : 0 \leq j \leq q\}$ be a daisy partition obtained by an application of Lemma 4.1. Then, for every $j \in [q]$ and $S \in \mathcal{D}_j$, the number of sets in $\mathcal{D}_j$ whose petals intersect $S \setminus K_j$ (including S itself) is at most $2h(j) = 2n^{\max\{1,j-1\}/q}$.*

Proof. Let S be an arbitrary set in $\mathcal{D}_j$. We name the elements in $S \setminus K_j$ by $u_1, u_2, \ldots, u_j$ (by Lemma 4.1, every $S \in \mathcal{D}_j$ satisfies $|S \setminus K_j| = j$). For every $k \in [j]$, let d_k be the number of sets of $\mathcal{D}_j$ that u_k is a member of.

Assume without loss of generality that $d_k \leq d_\ell$ for every k and ℓ in $[j]$ such that $k < \ell$, as otherwise we can rename $u_1, u_2, \ldots, u_j$ so that this holds.

By the definition of an h-daisy, for every $\ell \in [j]$, there exists a set of ℓ elements $k \in [j]$ that satisfy $d_k \leq h(\ell)$. Thus, $[\ell]$ is such a set and we know that $d_\ell \leq h(\ell)$. As the number of petals that intersect $S \setminus K_j$ is at most $\sum_{k=1}^{j} d_k$, we get that

$$\sum_{k=1}^{j} d_k \leq \sum_{k=1}^{j} h(k) = \sum_{k=1}^{j} n^{\frac{\max\{1,k-1\}}{q}} \leq 2h(j).$$

The last equality follows directly from the value of $h(k)$. □

4.3.2 Sampling daisies and the Hajnal-Szemerédi theorem

Concentration of measure is an essential ingredient in our proofs, which we first illustrate via a simplified example. Consider a collection of singletons that comprise the petals of a combinatorial sunflower: sets $P_1, \ldots P_k$, all disjoint and of size 1, contained in the ground set $[n]$. If we perform binomial sampling of the ground set (sampling each $i \in [n]$ independently with probability p), the Chernoff bound ensures that the number of sampled petals is close to its expectation. Defining X_i as the random variable that indicates whether P_i was sampled, we have lower and upper tail bounds that guarantee the number of queried petals is concentrated around pn except with exponentially small probability. Note, too, that the same holds for larger petals: if P_i is a j-set for all i, the number of queried petals is concentrated around $p^j n$.

Now consider the case where $P_1, \ldots, P_k$ are petals of a *daisy*. In this case the Chernoff bound does not apply, since the indicator random variables X_i are no longer independent; however, the structure of a daisy ensures there is not too much intersection among these petals, which gives means to control the correlation between these random variables. It is thus reasonable to expect that *sampling a daisy is similar to sampling a sunflower.* This intuition is formalised by making use of the Hajnal-Szemerédi theorem [HS70], which we state next.

Theorem 4.5. *Let G be a graph with m vertices and maximum degree Δ. Then, for any $k \geq \Delta + 1$, there exists a k-colouring of the vertices of G such that every colour class has size either $\lfloor m/k \rfloor$ or $\lceil m/k \rceil$.*

We remind that integrality does not cause issues in our analyses, and we thus assume all colour classes have size n/k. By encoding the sets of a daisy as the vertices of an "intersection graph", the fact that petals have bounded intersection translates into a graph with bounded maximum degree. Applying the Hajnal-Szemerédi theorem to this graph, we are able to partition the original daisy into a small number of large *simple* daisies.

Lemma 4.2. *A daisy $\mathcal{D}$ with kernel K, such that each one of its petals has a non-empty intersection with at most $t-1$ other petals, can be partitioned into t simple daisies with the same kernel, each of size $|\mathcal{D}|/t$.*

Proof. Construct a graph G with vertex set $\mathcal{D}$ by placing an edge between vertices S and S' when $(S \cap S') \setminus K \neq \varnothing$. By definition, the maximum degree of G is $\Delta(G) \leq t-1$. The Hajnal-Szemerédi theorem implies G is colourable with t colours, where each colour class has size $|G|/t$. This partition of the vertex set corresponds to a partition of the daisy $\mathcal{D}$ into simple daisies $\{\mathcal{S}_j : j \in [t]\}$, each of size $|\mathcal{D}|/t$. $\square$

4.3.3 The volume lemma

This section proves a key lemma that makes use of daisies to prove a certain structure on the sets that a *robust* local algorithm may query. Loosely speaking, the volume lemma ensures that in order for a collection of sets to be queried with high enough probability, it must cover a sufficiently large fraction of the input's coordinates.

Let M be a q-local algorithm that computes a partial function f with error rate σ (we assume the explicit input to be fixed and omit it). Recall that, for each input $x \in \Gamma^n$, the algorithm M queries according to a distribution μ_x over a multi-collection of q-sets, as defined in Definition 4.1.

Lemma 4.3 (Volume lemma). *Fix $x \in \Gamma^n$ in the domain of f. If there exists a ρ-robust $y \in \Gamma^n$ such that $f(y) \neq f(x)$, then every collection $\mathcal{S} \subseteq \operatorname{supp}(\mu_x)$ such that $|\cup\mathcal{S}| = |\cup_{S\in\mathcal{S}} S| < \rho n$ satisfies $\mu_x(\mathcal{S}) \leq 2\sigma$.*

Proof. Suppose, by way of contradiction, that there exists $\mathcal{S} \subseteq \operatorname{supp}(\mu_x)$ such that $\mu_x(\mathcal{S}) > 2\sigma$ and $|\cup\mathcal{S}| < \rho n$.

For notational simplicity, assume without loss of generality that $f(x) = 1$, and take a ρ-robust $y \in \Gamma^n$ such that $f(y) = 0$. Define w to match x in the coordinates covered by $\cup\mathcal{S}$, and to match y otherwise. Then w is ρ-close to y, so that M outputs 0 when its input is w with probability at least $1 - \sigma$.

When the algorithm samples a decision tree that makes it query $S \in \mathcal{S}$, then M behaves *exactly* as it would on input x, which happens with probability at least $\mu_x(\mathcal{S}) > 2\sigma$. But the algorithm outputs 1 on input x with probability at most σ, and thus outputs 0 on input w with probability greater than σ, in contradiction with the robustness of y. $\square$

Remark 4.6. The volume lemma requires an arbitrary ρ-robust y with $f(y) \neq f(x)$. It thus suffices that *a single* such ρ-robust point exists for the volume lemma to hold *for every x' such that $f(x') = f(x)$*.

4.3.4 Generic transformations

This section provides two standard transformations that improve parameters of an algorithm: error reduction (Claim 4.6) and randomness reduction (Claim 4.7), which, applied in conjunction, imply Lemma 4.4. These apply generally to randomised algorithms for decision problems, and, when applied to robust local algorithms, *both transformations compute the same function and preserve robustness.*

The following claim is an adaptation of a basic fact regarding randomised algorithms: performing independent runs and selecting the output via a majority rule decreases the error probability exponentially.

Claim 4.6 (Error reduction). *Let M be a (ρ_0, ρ_1)-robust algorithm for $f\colon \mathcal{P} \to \{0,1\}$ (where $\mathcal{P} \subset Z \times \Gamma^n$) with error rate $\sigma \leq 1/3$, query complexity q and randomness complexity r.*

For any $\sigma' > 0$, there exists a (ρ_0, ρ_1)-robust algorithm N for computing the same function with error rate σ', query complexity $O(q \log(1/\sigma')/\sigma)$ and randomness complexity $O(r \log(1/\sigma')/\sigma)$.

Proof. Define N as the algorithm that makes $t = 108 \log(1/\sigma')/\sigma$ independent runs of M and outputs the most frequent symbol, resolving ties arbitrarily. The query and randomness complexities of N clearly match the statement, and we must now prove that the error rate is indeed σ' and that N is (ρ_0, ρ_1)-robust.

Fix $z \in Z$ and $x \in \Gamma^n$ in the domain of f and let $b := f(z, x)$. As M is ρ_b-robust at x, the algorithm satisfies $\Pr[M^y(z) = b] \geq 1 - \sigma$ for all $y \in B_{\rho_b}(x)$. By the Chernoff bound (Lemma 2.2),

$$\Pr\left[M^y(z) \neq b \text{ for at least } \left(\sigma + \frac{1}{6}\right) t \text{ runs}\right] \leq e^{-\frac{\sigma t}{3 \cdot 36}} = e^{-\log \frac{1}{\sigma'}} < 2^{-\log \frac{1}{\sigma'}} = \sigma'.$$

The majority rule will thus yield outcome b with probability at least $1 - \sigma'$, since at least $1 - \left(\sigma + \frac{1}{6}\right) t \geq t/2$ runs output b (except with probability at most σ'). As x, z and $y \in B_{\rho_b}(x)$ are arbitrary, the result follows. □

Next, we state a transformation that yields an algorithm with twice the error rate and significantly reduced randomness complexity. This, in turn, provides an upper bound on the number of q-sets queried by the algorithm, such that an application of Lemma 4.1 to this multi-collection yields daisies with kernels of sublinear size. Such a bound on the size of the kernels is crucial to ensure correctness of the sample-based algorithm we construct in Section 4.4.1. Our proof adapts the technique of Goldreich and Sheffet [GS10], which in turn builds on the work of Newman [New91].

Claim 4.7 (Randomness reduction). *Let M be a (ρ_0, ρ_1)-robust algorithm for $f \colon \mathcal{P} \to \{0,1\}$ (where $\mathcal{P} \subset Z \times \Gamma^n$) with error rate σ, query complexity q and randomness complexity r.*

There exists a (ρ_0, ρ_1)-robust algorithm N for computing the same function with error rate 2σ and query complexity q, whose distribution $\tilde{\mu}^N$ has support size $3n \ln |\Gamma|/\sigma$. In particular, the randomness complexity of N is bounded by $\log(n/\sigma) + \log\log|\Gamma| + 2$.

Proof. Fix any explicit input $z \in Z$. Let $\{x_j\}$ be an enumeration of the inputs in Γ^n such that $\Pr[M^{x_j}(z) = b_j] \geq 1 - \sigma$ for some $b_j \in \{0,1\}$. Note that this includes points in the neighbourhood of a point at which M is robust which are not necessarily in the domain of f, so it suffices to show $\Pr[N^{x_j}(z) = b_j] \geq 1 - 2\sigma$ to prove the claim for N with the required query complexity and distribution.

Define the $2^r \times |\{x_j\}|$ matrix E with entries in $\{0,1\}$ as follows. Denote by $b_{ij} \in \{0,1\}$ the output of $M^{x_j}(z)$ when it executes according to the decision tree indexed by (the binary representation of) $i \in [2^r]$. Then,

$$E_{i,j} = \begin{cases} 1, & \text{if } b_{ij} \neq b_j \\ 0, & \text{otherwise.} \end{cases}$$

Note that $E_{i,j}$ simply indicates whether $M^{x_j}(z)$ outputs incorrectly on input when the outcome of the algorithm's coin flips is (the binary representation of) i. By construction, for each fixed j, a fraction of at most σ indices $i \in [2^r]$ are such that $E_{i,j} = 1$.

Let $t = 3n \ln |\Gamma|/\sigma$ and $I_1, \ldots, I_t$ be independent random variables uniformly distributed in $[2^r]$. For each fixed $j \leq |\{x_j\}| \leq |\Gamma|^n$ and $k \leq t$, we have $\mathbb{E}[E_{I_k,j}] \leq \sigma$. By the Chernoff bound,

$$\Pr\left[\sum_{k=1}^{t} E_{I_k,j} \geq 2\sigma t\right] \leq e^{-\frac{\sigma t}{3}} = e^{-n \ln |\Gamma|} < |\Gamma|^{-n}.$$

Applying the union bound over all $j \leq |\{x_j\}| \leq |\Gamma|^n$, we obtain

$$\Pr\left[\sum_{k=1}^{t} E_{I_k,j} \geq 2\sigma t \text{ for some j}\right] < 1.$$

We have thus shown, via the probabilistic method, the existence of a multi-set R_z of size $3n \ln |\Gamma|/\sigma$ such that

$$\Pr[N^{x_j}(z) \neq b_j] \leq 2\sigma,$$

where N samples its random strings uniformly from R_z (rather than from $\{0,1\}^r$), using the corresponding decision trees of M. The size of S_z is thus $\left|\tilde{\mu}^N\right| = 3n \ln|\Gamma|/\sigma$, and this sampling can be performed with $\log(n/\sigma) + \log\log|\Gamma| + 2$ random coins.

Since the decision trees of N are simply a subcollection of those of M, the query complexity of N is q and the claim follows. □

In the next section, we need a combination of error reduction and randomness reduction, which the following lemma provides.

Lemma 4.4. *Assume there exists a ρ-robust algorithm M for computing f with query complexity ℓ, error rate $1/3$ and arbitrary randomness complexity. Then there exists a ρ-robust q-local algorithm M' for f with error rate*

$$\sigma = \frac{1}{4q}$$

such that $\frac{q}{\log 8q} = O(\ell)$, or, equivalently,

$$q = O(\ell \log \ell).$$

Moreover, the distribution of M' is uniform over a multi-collection of decision trees of size $6n \ln|\Gamma|/\sigma$.

Proof. We apply both transformations in order, omitting mention of parameters that are left unchanged. Recall that M may have arbitrarily large randomness complexity.

1. Apply Claim 4.6 (error reduction) to M, obtaining M'' with error rate $\sigma'' = 1/8q$ and query complexity $q = O(\ell \log(1/\sigma'')) = O(\ell \log(8q))$ (as well as larger randomness complexity).
2. Apply Claim 4.7 (randomness reduction) to M'', thereby obtaining a new algorithm M' with error rate $\sigma = 2\sigma'' = \frac{1}{4q}$ and support size $3n \ln|\Gamma|/\sigma'' = 6n \ln|\Gamma|/\sigma$ on its distribution over decision trees. □

4.4 Proof of Theorem 1

This section contains the main technical contribution of our work: a proof that every robust local algorithm with query complexity q can be transformed into a *sample-based* local algorithm with sample complexity $n^{1-1/O(q^2 \log^2 q)}$. We begin by providing a precise statement of Theorem 1. In the following, we remind that when the error rate of an algorithm is not stated, it is assumed to be $1/3$.

Theorem 4.7 (Theorem 1, restated)**.** *Suppose there exists a (ρ_0, ρ_1)-robust local algorithm M for the function $f \colon \mathcal{P} \to \{0,1\}$ (where $\mathcal{P} \subset Z \times \Gamma^n$) with query complexity ℓ and* $\max\{\rho_0, \rho_1\} = \Omega(1)$*. Then, there exists a* sample-based *algorithm N for f with sample complexity $\gamma \cdot n^{1-1/2q^2}$, where $q = O(\ell \log \ell)$ and $\gamma = O(|\Gamma|^q \ln |\Gamma|)$.*

Note that when $q = \Omega(\sqrt{\log n})$ or $|\Gamma|^q = \Omega\big(n^{1/2q^2}\big)$, the algorithm we obtain samples linearly many coordinates, and the statement becomes trivial. Therefore, hereafter we assume that the query complexity of M satisfies $\ell \leq \sqrt[5]{\log n}$ (so $q = \Theta(\sqrt[5]{\log n} \log\log n) = o(\sqrt{\log n})$) and the alphabet size satisfies $|\Gamma| \leq 2^{\sqrt[6]{\log n}}$ (so $|\Gamma|^q \leq n^{1/q^3}$).

We proceed to prove Theorem 4.7. Specifically, in Section 4.4.1 we construct a sample-based local algorithm N from the (ρ_0, ρ_1)-robust local algorithm M in the hypothesis of Theorem 4.7; in Section 4.4.2, we analyse our sample-based algorithm N; and in Section 4.4.3 we conclude the proof by showing the lemmas proved throughout the analysis indeed imply correctness of N.

4.4.1 Construction

Hereafter, let $f \colon \mathcal{P} \to \{0,1\}$ be the function in the hypothesis of Theorem 4.7. As the following treatment is the same for all explicit inputs $z \in Z$, we assume it to be fixed and omit it from the notation. We also assume without loss of generality that ρ_0 is a constant strictly greater than 0 (if this is not the case we simply exchange the 0 and 1 values in the truth table of f). We set $\rho = \rho_0$.

Let M be the algorithm in the hypothesis of Theorem 4.7. We apply Lemma 4.4 and obtain a (ρ_0, ρ_1)-robust local algorithm M' for the same problem, with query complexity $q = O(\ell \log \ell)$ and error rate $\sigma = 1/4q$. The algorithm N we describe below has white box access to the local algorithm M'. We next explain how it extracts information from it.

Upon execution, M' chooses a decision tree uniformly at random according to the outcome of its coin flips; this uniform distribution is denoted $\tilde{\mu} = \tilde{\mu}^{M'}$, whose support size is $|\tilde{\mu}|$. For every decision tree and every one of its branches, define a *description tuple* (S, a_S, b, s), where s is the random string that will cause the use of this tree, S is the set of all the queries in this branch, a_S is the assignment to these queries that will result in M' using this specific branch and b is the value M' returns when this occurs (as per Definition 4.1).

We assume that for every description-tuple (S, a_S, b, s) the size of S is exactly q. This can be assumed without loss of generality since it is possible to convert M' into an algorithm such that every decision tree and every one of its branches makes q

distinct queries: if the same query appears more than once on a branch of a tree, all but the first appearance can be removed by choosing the continuation that follows the (already known) value that leads to the algorithm using this branch. In addition, a tree can be expanded by adding queries, so that every branch has exactly q distinct queries. Both of these changes do not change any of the parameters of the algorithm beyond ensuring that it will query exactly q coordinates.

The algorithm N we describe next only makes use of description tuples (S, a_S, b, s) such that $b = 1$. To this end we set

$$\mathcal{T} = \{(S, a_S, b, s) : (S, a_S, b, s) \text{ is a description tuple such that } b = 1\}.$$

Algorithm N also requires the multi-collection $\mathcal{S}$ defined as follows:

$$\mathcal{S} = \{S : (S, a_S, b, s) \in \mathcal{T}\}.$$

Specifically, it applies Lemma 4.1 to get a daisy partition of $\mathcal{S}$. When the algorithm extracts $\mathcal{T}$ and $\mathcal{S}$ from M' and computes a daisy partition for $\mathcal{S}$, it preserves the information that allows it to associate the set of a tuple of (S, a_S, b, s) to the unique daisy S is contained in.

The construction proceeds in two stages: *preprocessing* and *execution*. Recall that, for any input $x \in \Gamma^n$ and assignment κ to a subset $S \subseteq [n]$, we denote by x_κ the word that assigns the same values as κ on S and the same values as x on $[n] \setminus S$.

Preprocessing. N has access to M', with which it computes $\mathcal{T}$ and $\mathcal{S}$. Applying Lemma 4.1 to $\mathcal{S}$, the algorithm obtains the *daisy partition*

$$\mathcal{D} = \{\mathcal{D}_j : 0 \leq j \leq q\},$$

so that each tuple in $\mathcal{T}$ is associated with $\mathcal{D}_j$ for exactly one $j \in \{0, \dots, q\}$. Set

$$p := \gamma \cdot n^{-1/2q^2},$$

the *sampling probability*, where $\gamma = 24|\Gamma|^q \ln |\Gamma|$; and, for every $j \in [q]$, set

$$\tau_j := \frac{|\tilde{\mu}|}{2q} \cdot p^j,$$

the *thresholds*, which will be used in the execution stage.

Execution. When N receives query access to a string $x \in \Gamma^n$, it performs the following sequence of steps.

1. *Sampling:* Select each element in $[n]$ independently with probability p. Denote by Q the set of all coordinates thus obtained. If $|Q| \geq 2pn$, then N outputs arbitrarily. Otherwise, N queries all the coordinates in Q.
2. *Enumeration:* For every $j \in [q]$ and kernel assignment κ to K_j,[2] perform the following steps. Set a counting variable v to 0 before each iteration.
   (a) *Local view generation and vote counting:* For every tuple $(S, a_S, 1, s) \in \mathcal{T}$ such that $S \in \mathcal{D}_j$, increment v if $S \subset Q \cup K_j$ and a_S assigns on S the same values as x_κ does.
   In the case $j = 1$, if $12 \ln |\Gamma|/(\rho \cdot \sigma)$ sets have the same point outside K_1, disregard them in the count.[3]
   (b) *Threshold check:* If $v \geq \tau_j$, output 1.
3. If the condition $v \geq \tau_j$ was never satisfied, then output 0.

We proceed to analyse this construction.

4.4.2 Analysis

We remind that the explicit input z is assumed to be fixed and is omitted from the notation. For the analysis we are interested in the behaviour of the algorithm M' on a fixed input x. For this purpose, we use the distribution μ_x from Definition 4.1.

For $x \in \Gamma^n$ we define μ_x to be the uniform distribution over the multi-collection of sets

$$\left\{ S : (S, a_S, b, s) \text{ is a description tuple such that } a_S = x_{|S} \right\}, \tag{4.1}$$

where a description tuple is as appears in Section 4.4.1. We note that this implies that $\mathrm{supp}(\mu_x)$ has exactly one set for each decision tree M' may use, since when both the randomness and the input are fixed exactly one branch of the decision tree is used by M'. Therefore,

$$|\mu_x| = |\tilde{\mu}| \,.$$

We now list the relevant parameters in the analysis with reference to where they are obtained. By Lemma 4.4,

$$\sigma = \frac{1}{4q} \,, \tag{4.2}$$

[2]Note that the algorithm *does not* iterate over the case $j = 0$. We will show in Section 4.4.2 that this has a negligible effect.

[3]This is required for technical purposes when dealing with K_1.

and, for every $x \in \Gamma^n$,

$$|\mu_x| = |\tilde{\mu}| = \frac{6n \ln |\Gamma|}{\sigma} . \tag{4.3}$$

The construction of N in the previous section sets the parameters

$$\gamma = 24|\Gamma|^q \ln |\Gamma| , \tag{4.4}$$

$$p = \gamma \cdot n^{-1/2q^2} , \tag{4.5}$$

and, for all $j \in [q]$,

$$\tau_j = \frac{|\tilde{\mu}|}{2q} \cdot p^j = \frac{|\mu_x|}{2q} \cdot p^j . \tag{4.6}$$

(The second equality holds for all $x \in \Gamma^n$ by Eq. 4.3.) Finally, the size of the collection of tuples $\mathcal{T}$, which by the construction in Section 4.3.4 is the same as that of $\mathcal{S}$, is bounded by the total number of branches over all decision trees in $\mathrm{supp}(\tilde{\mu})$. Thus,

$$|\mathcal{S}| = |\mathcal{T}| \leq |\Gamma|^q \cdot |\tilde{\mu}| = |\Gamma|^q \cdot |\mu_x| , \tag{4.7}$$

for every $x \in \Gamma^n$.

For our result we need an upper bound on the sizes of the kernels that algorithm N enumerates over, which we show next.

Claim 4.8. *Let $\{K_i : 0 \leq i \leq q\}$ be the kernels of the daisy partition $\{\mathcal{D}_i\}$ of $\mathcal{S}$ used by the algorithm N. For every $i \in \{0, 1, \ldots, q\}$, the kernel K_i is such that $|K_i| \leq \gamma \cdot q^2 \cdot n^{1-\max\{1,i\}/q}$ and, for n sufficiently large, $|K_i| < \rho n/2$.*

Proof. By Lemma 4.1, for every $i \in \{0, 1, \ldots, q\}$,

$$\begin{aligned}
|K_i| &\leq q|\mathcal{S}|n^{-\max\{1,i\}/q} \\
&\leq q|\Gamma|^q|\mu_x| \cdot n^{-\max\{1,i\}/q} && (\text{by Eq. 4.7, } |\mathcal{S}| \leq |\Gamma|^q \cdot |\mu_x|) \\
&\leq q|\Gamma|^q \cdot \frac{6n \ln |\Gamma|}{\sigma} \cdot n^{-\max\{1,i\}/q} && \left(\text{by Eq. 4.3, } |\mu_x| \leq \frac{6n \ln |\Gamma|}{\sigma}\right) \\
&= 24|\Gamma|^q \cdot \ln |\Gamma| \cdot q^2 \cdot n^{-\max\{1,i\}/q} && \left(\text{by Eq. 4.2, } \sigma = \frac{1}{4q}\right) \\
&= \gamma \cdot q^2 \cdot n^{1-\max\{1,i\}/q} && (\text{by Eq. 4.4, } \gamma = 24|\Gamma|^q \ln |\Gamma|)
\end{aligned}$$

It remains to prove the second part of the claim. By the calculation above, since ρ is constant and $|\Gamma|^q \ln |\Gamma| \cdot q^2 = o(n^{-1/q})$ (recall that $|\Gamma| \leq n^{1/q^4}$ and q is sub-logarithmic), for sufficiently large n,

$$|K_0| \leq \gamma \cdot q^2 \cdot n^{1-1/q} = \left(24|\Gamma|^q \ln |\Gamma| \cdot q^2 \cdot n^{-1/q}\right) n \leq \rho n/2. \tag{4.8}$$

By Lemma 4.1, $K_q \subseteq K_{q-1}, \ldots, K_1 = K_0$, and hence the claim follows. $\square$

Next, we provide a number of definitions emanating from algorithm N. We define, for every $x \in \Gamma^n$, the multi-collection

$$\mathcal{O}^x \coloneqq \left\{ S : (S, a_S, 1, s) \in \mathcal{T} \text{ and } x_{|S} = a_S \right\},$$

where $\mathcal{T}$ is defined as in Section 4.4.1. Note that the definition of this collection depends only on the algorithm M and not on the function f it computes. Hence, it is well-defined for every x, and in particular for points that are ρ-close to a ρ-robust point of the domain (where f may not be defined). We note that, since μ_x is defined over the collection in Eq. 4.1 we know that

$$\mathcal{O}_x \subseteq \operatorname{supp}(\mu_x)\,. \tag{4.9}$$

Since the "capping parameter" $12 \ln |\Gamma| / (\rho \cdot \sigma)$ is used numerous times, we set

$$\alpha = \frac{12 \ln |\Gamma|}{\rho \cdot \sigma}\,. \tag{4.10}$$

We refer to the act of incrementing v as counting a vote. For each $j \in [q]$, we define the *vote counting function* $v_j \colon \Gamma^n \to \mathbb{N}$ to be a random variable (determined by Q) as follows. If $j > 1$,

$$v_j(x) \coloneqq |\{S \in \mathcal{O}^x \cap \mathcal{D}_j : S \subseteq Q \cup K_j\}|\,,$$

and $v_1(x)$ is defined likewise, with the exception that, when more than α sets intersect in a point outside K_1, they are discarded.

Claim 4.9. *Let $x \in \Gamma^n$, $j \in [q]$ and κ an assignment to K_j. Then $v_j(x_\kappa)$ is equal (as a function of Q) to the maximum value of the counter v computed by N on input x with kernel K_j and the kernel assignment κ to K_j.*

Proof. Fix $x \in \Gamma^n$. Recall that when algorithm N computes v for a $j \in \{2, 3, \ldots, q\}$ and a kernel assignment κ to K_j in Step 2a, it only increases v if it encounters a tuple $(S, a_S, 1, s)$ where $S \in \mathcal{D}_j$, $S \subset Q \cup K_j$ and a_S assigns on S the same values as x_κ does. Thus, by the definition of $\mathcal{O}_x$, the algorithm N counts exactly all the tuples $(S, a_S, 1, s)$ such that $S \in \mathcal{O}^x$ and $S \subset Q \cup K_j$. These are precisely the sets that comprise the collection whose cardinality is $v_j(x_\kappa)$. Note that the same holds when $j = 1$ due to the additional condition in Step 2a and the corresponding restriction in the definition of $v_1(x_\kappa)$. $\square$

We now proceed to the main claims. The algorithm N only counts votes for output 1, i.e. tuples with 1 as their third value, and hence it suffices to prove that: (1) *when* $f(x) = 1$ *and the kernel assignment is* $\kappa = x_{|K_j}$ (the value of x on the indices in K_j) *for some daisy* $\mathcal{D}_j$, the number of votes is high enough to cross the threshold τ_j; and that (2) *when* $f(x) = 0$*, then every kernel assignment* κ is such that the number of votes is smaller than the threshold. These conditions are shown to hold with high probability in Sections 4.4.2.1 and 4.4.2.2, respectively, and we show how the theorem follows from them in Section 4.4.3.

4.4.2.1 Correctness on non-robust inputs

Claim 4.10. *Let Q be the coordinates sampled by N and fix $x \in \Gamma^n$ such that $f(x) = 1$. There exists $j \in [q]$ such that, with the kernel assignment $\kappa = x_{|K_j}$, the vote counting function satisfies $v_j(x_\kappa) = v_j(x) \geq \tau_j$ with probability at least $9/10$.*

Proof. For ease of notation, let us fix x as in the statement and denote $\mathcal{O} := \mathcal{O}^x = \mathcal{O}^{x_\kappa}$. When $j > 1$, define the subcollection of $\mathcal{O}$ in $\mathcal{D}_j$ by $\mathcal{O}_j := \mathcal{O}^x \cap \mathcal{D}_j$; when $j = 1$, define $\mathcal{O}_1 := (\mathcal{O}^x \cap \mathcal{D}_j) \setminus \mathcal{C}$, where $\mathcal{C} \subseteq \mathcal{O}^x \cap \mathcal{D}_1$ contains every $S \in \mathcal{O}^x \cap \mathcal{D}_1$ for which there exist at least $\alpha - 1$ other sets in $S' \in \mathcal{O}^x \cap \mathcal{D}_1$ that have the same petal as S, i.e., such that $S \setminus K_1 = S' \setminus K_1$. We also take n to be sufficiently large when necessary for an inequality to hold.

For the claim to hold we require the existence of $j \in [q]$ such that $\mathcal{O}_j$ is a sufficiently large portion of $\mathcal{O}$. Since $\bigcup_{0<j\leq q} \mathcal{O}_j = \mathcal{O} \setminus (\mathcal{O}_0 \cup \mathcal{C})$, in order to achieve this goal, we only need to bound the sizes of both $\mathcal{O}_0$ and $\mathcal{C}$. As a first step, we bound $\mu_x(\mathcal{O}_0)$ and $\mu_x(\mathcal{C})$, which give us a lower bound on $\mu_x\left(\bigcup_{0<j\leq q} \mathcal{O}_j\right)$, which we then use in order to lower bound $\left|\bigcup_{0<j\leq q} \mathcal{O}_j\right|$.

We start with $\mu_x(\mathcal{O}_0)$. All the sets in $\mathcal{D}_0$ are subsets of K_0, and $|K_0| < \rho n/2$ by Claim 4.8. This implies that the cardinality of $\cup\mathcal{O}_0$ (the union of all sets in $\mathcal{O}_0$) is strictly less than ρn, and consequently, by the volume lemma (Lemma 4.3, which applies because $f(x) = 1$), we have $\mu_x(\mathcal{O}_0) \leq 2\sigma$.

We now proceed to bound $\mu_x(\mathcal{C})$. As $\mathcal{C} \subseteq \mathcal{D}_1$, every set in $\mathcal{C}$ has exactly one element in $[n] \setminus K_1$ and repeats at least α times, the cardinality of $\cup\mathcal{C}$ is at most

$|K_1| + \frac{|\mathcal{C}|}{\alpha}$. By Claim 4.8, $|K_1| < \rho n/2$, and it follows that

$$\begin{aligned}
|K_1| + \frac{|\mathcal{C}|}{\alpha} &< \frac{\rho n}{2} + \frac{|\mathcal{O}|}{\alpha} \\
&\leq \frac{\rho n}{2} + \frac{|\mu_x|}{\alpha} && (\text{by Eq. 4.9, } \mathcal{O} = \mathcal{O}_x \subseteq \mathrm{supp}(\mu_x)) \\
&\leq \frac{\rho n}{2} + |\mu_x| \cdot \frac{\rho \cdot \sigma}{12 \ln |\Gamma|} && \left(\text{by Eq. 4.10, } \alpha^{-1} = \frac{\rho \cdot \sigma}{12 \ln |\Gamma|}\right) \\
&= \frac{\rho n}{2} + \frac{6n \ln |\Gamma|}{\sigma} \cdot \frac{\rho \cdot \sigma}{12 \ln |\Gamma|} && \left(\text{by Eq. 4.3, } |\mu_x| = \frac{6n \ln |\Gamma|}{\sigma}\right) \\
&\leq \rho n \, .
\end{aligned}$$

Consequently, by Lemma 4.3, $\mu_x(\mathcal{C}) \leq 2\sigma$.

By the definition of error rate, $\mu_x(\mathcal{O}) \geq 1 - \sigma$. Since $\{\mathcal{O}_j : 0 \leq j \leq q\}$ is a partition of $\mathcal{O}$ (because $\{\mathcal{D}_j\}$ is a partition),

$$\mu_x \left(\bigcup_{0<j\leq q} \mathcal{O}_j \right) = \mu_x(\mathcal{O}) - \mu_x(\mathcal{O}_0) - \mu_x(\mathcal{C}) \geq 1 - 5\sigma \, .$$

As μ_x is uniform, each element of the multi-collection $\mathcal{O}$ has weight exactly $1/|\mu_x|$. Therefore,

$$\sum_{j=1}^{q} |\mathcal{O}_j| = |\mu_x| \cdot \mu_x(\cup_{j\in[q]} \mathcal{O}_j) \geq |\mu_x|(1 - 5\sigma) \geq |\mu_x|/2 \, ,$$

where the last inequality follows from the assumption that $5\sigma \leq 1/2$ (which follows, e.g., from $q \geq 3$, which Lemma 4.4 ensures). Let j be such that

$$|\mathcal{O}_j| \geq \frac{|\mu_x|}{2q} \, ; \tag{4.11}$$

by averaging, such a j indeed exists. Our goal now is to show that with probability at least $9/10$, there are at least τ_j sets $S \in \mathcal{O}_j$ whose petal is in Q, i.e., such that $S \setminus K_j \subseteq Q$.

Instead of proving this directly on $\mathcal{O}_j$, we do so on collections that form a partition of $\mathcal{O}_j$ and have a useful structure. The sets in $\mathcal{O}_j$ are also in $\mathcal{D}_j$, so that $\mathcal{O}_j$ is also a daisy with kernel K_j. By Claim 4.5, for every set $S \in \mathcal{O}_j$, there exist at most $2n^{\max\{1,j-1\}/q} - 1$ sets $S' \in \mathcal{O}_j \setminus \{S\}$ whose petals have a non-empty intersection with the petal of S, i.e, such that $(S \cap S') \setminus K_j \neq \varnothing$. This enables us to apply Lemma 4.2

to $\mathcal{O}_j$, partitioning it into $\{\mathcal{S}_i : i \in [t]\}$ simple daisies of equal sizes, where

$$t \leq 2n^{\max\{1,j-1\}/q} . \tag{4.12}$$

Thus, for every $i \in [t]$,

$$|\mathcal{S}_i| = \frac{|\mathcal{O}_j|}{t} . \tag{4.13}$$

Let $\mathcal{O}'_j$ be the multi-collection of all sets $S \in \mathcal{O}_j$ such that $S \setminus K_j \subseteq Q$. In the same manner, for every $i \in [t]$, let $\mathcal{S}'_i$ be the multi-collection of all sets $S \in \mathcal{S}_i$ such that $S \setminus K_j \subseteq Q$.

By construction, the collections $\{\mathcal{S}'_i\}$ are pairwise disjoint. Also, by the definition of v_j, we have $v_j(x) = \left|\mathcal{O}'_j\right| = \sum_{i=1}^{t} |\mathcal{S}'_i|$. Therefore, the event that $v_j(x) \leq \tau_j$ can only occur if there exists $i \in [t]$ such that $|\mathcal{S}'_i| \leq \tau_j/t$.

Consequently, we obtain

$$\begin{aligned}
\Pr\left[v_j(x) \leq \tau_j\right] &\leq \Pr\left[\left|\mathcal{S}'_i\right| \leq \frac{\tau_j}{t} \text{ for some } i \in [t]\right] \\
&\leq \sum_{i=1}^{t} \Pr\left[\left|\mathcal{S}'_i\right| \leq \frac{\tau_j}{t}\right] && \text{(union bound)} \\
&\leq t \Pr\left[\left|\mathcal{S}'_1\right| \leq \frac{\tau_j}{t}\right] . && \text{(all } \mathcal{S}_i \text{ have equal size)}
\end{aligned}$$

We show afterwards that the probability of the event $|\mathcal{S}'_1| \leq \frac{\tau_j}{t}$ is strictly less than $1/10t$, which by the inequality above implies the claim.

We later use the Chernoff bound with $\mathcal{S}_1$, and hence we start by bounding $\mathbb{E}[|\mathcal{S}'_1|]$ from below. Recall that the petal of every set $S \in \mathcal{S}_1 \subseteq \mathcal{D}_j$ has size j (i.e., $|S \setminus K_j| = j$), and therefore is in $\mathcal{S}'_1$ with probability exactly p^j. So

$$\begin{aligned}
\mathbb{E}[|\mathcal{S}'_1|] = |\mathcal{S}_1| p^j &= \frac{|\mathcal{O}_j| p^j}{t} && \text{(by Eq. 4.13, } |\mathcal{S}_1| = |\mathcal{O}_j|/t) \\
&\geq \frac{|\mu_x| p^j}{2tq} && \left(\text{by Eq. 4.11, } |\mathcal{O}_j| \geq \frac{|\mu_x|}{2q}\right) \\
&= \frac{\tau_j}{t} . && \left(\text{by Eq. 4.6, } \tau_j = \frac{|\mu_x|}{2q} \cdot p^j\right)
\end{aligned} \tag{4.14}$$

Thus,

$$\Pr\left[\left|\mathcal{S}'_1\right| \leq \frac{1}{2}\mathbb{E}[|\mathcal{S}'_1|]\right] \geq \Pr\left[\left|\mathcal{S}'_1\right| \leq \frac{\tau_j}{t}\right] .$$

Next we show that the probability of the event $|\mathcal{S}'_1| \leq \frac{1}{2}\mathbb{E}[|\mathcal{S}'_1|$ is at most $1/10t$, which concludes the proof. Since $\mathcal{S}_1$ is a simple daisy, the petals of sets in $\mathcal{S}_1$ are pairwise disjoint and hence the events $S \setminus K_j \subset Q$, for every $S \in \mathcal{S}_1$, are all independent. This enables us to use the Chernoff bound to get that

$$\begin{aligned}
&\Pr\left[|\mathcal{S}_1'| \le \frac{1}{2}\mathbb{E}[|\mathcal{S}_1'|]\right] \\
&\quad\le \exp\left(-\frac{\mathbb{E}[|\mathcal{S}_1'|]}{8}\right) && \text{(Chernoff bound)} \\
&\quad\le \exp\left(-\frac{|\mu_x|p^j}{16tq}\right) && \text{(by Eq. 4.14)} \\
&\quad\le \exp\left(-\frac{6n\ln|\Gamma|}{\sigma}\cdot\frac{p^j n}{16tq}\right) && \text{(by Eq. 4.3)} \\
&\quad\le \exp\left(-4q\cdot\frac{3\ln|\Gamma|p^j n}{8tq}\right) && \text{(by Eq. 4.2)} \\
&\quad\le \exp\left(-\frac{3\ln|\Gamma|p^j}{4}\cdot n^{1-\frac{\max\{1,j-1\}}{q}}\right) && \text{(by Eq. 4.12)} \\
&\quad\le \frac{1}{10t}\exp\left(-\gamma^j\cdot\frac{3\ln|\Gamma|}{4}\cdot n^{1-\frac{\max\{1,j-1\}}{q}-\frac{j}{2q^2}}+\ln(20t)\right) && \left(p=\gamma\cdot n^{-1/2q^2}\right) \\
&\quad\le \frac{1}{10t}\exp\left(-\gamma\cdot\frac{3\ln|\Gamma|}{4}\cdot n^{\frac{1}{q}-\frac{1}{2q}}+\ln(10t)\right) && (1\le j\le q) \\
&\quad\le \frac{1}{10t},
\end{aligned}$$

where the last inequality follows for n sufficiently large because $\ln t \le \frac{\max\{1,j-1\}}{q}\ln n + 1 = o(n^{1/2q})$ and $\gamma\cdot\ln|\Gamma| = \Omega(1)$. □

Note that, although a success probability of 9/10 suffices to ensure correctness of a single run of N, Claim 4.10 yields a much stronger result: the failure probability is *exponentially small*. This is because Claim 4.10 does not enumerate over kernel assignments. Moreover, the analysis for the case $j = 1$ can be improved significantly (as will be necessary in Claim 4.11), but this does not yield in an overall improvement in our results.

4.4.2.2 Correctness on robust inputs

In the following claim we note that $|K_1|/n$-robustness suffices for the analysis, since it ensures all kernel assignments κ lead x_κ to also output $f(x) = 0$.

Claim 4.11. *Suppose the input $x \in \Gamma^n$ is $|K_1|/n$-robust for M' and $f(x) = 0$. Then, for every $j \in [q]$ and every assignment κ to the kernel K_j, the vote count satisfies $v_j(x_\kappa) < \tau_j$ with probability at least $1 - |\Gamma|^{|K_j|}/(10q)$.*

Proof. For ease of notation, fix $j \in [q]$, an assignment κ to K_j and x as in the statement, and let $\mathcal{O} := \mathcal{O}^{x_\kappa}$. If $j > 1$, define the subcollection of $\mathcal{O}$ in $\mathcal{D}_j$ by

$\mathcal{O}_j \coloneqq \mathcal{O} \cap \mathcal{D}_j$; if $j = 1$, define $\mathcal{O}_1 \coloneqq (\mathcal{O} \cap \mathcal{D}_1) \setminus \mathcal{C}$, where $\mathcal{C} \subseteq \mathcal{O} \cap \mathcal{D}_1$ contains every $S \in \mathcal{O} \cap \mathcal{D}_1$ for which there exist at least $\alpha - 1$ other sets $S' \in \mathcal{O} \cap \mathcal{D}_1$ that have the same petal as S, i.e., such that $S \setminus K_1 = S' \setminus K_1$. We also take n to be sufficiently large when necessary for an inequality to hold.

Note that x_κ may not be in the domain of f, but the robustness of x allows us to bound the size of $\mathcal{O} = \mathcal{O}^{x_\kappa}$ regardless. Moreover, since $f(x) = 0$, we know that $\mu_x(\mathcal{O}) \leq \sigma$. As μ_x is uniform, each element of the multi-collection has $\mathcal{O}$ weight exactly $1/|\mu_x|$. Therefore, for every $i \in [q]$,

$$|\mathcal{O}_i| \leq \sigma |\mu_x| \,. \tag{4.15}$$

Our goal now is, for every $j \in [q]$, to upper bound the probability that there are at least τ_j sets $S \in \mathcal{O}_j$ whose petal is in Q, i.e., such that $S \setminus K_j \subseteq Q$.

For every $j \in [q]$, let β_j be such that for every set $S \in \mathcal{O}_j$ there exist at most $\beta_j - 1$ other distinct sets $S' \in \mathcal{O}_j$ whose petal intersects the petal of S, i.e., $(S \setminus K_1) \cap (S' \setminus K_1) \neq \varnothing$.

For the time being let us fix $j \in [q]$. By applying Lemma 4.2, we partition $\mathcal{O}_j$ into $\{\mathcal{S}_i : i \in [\beta_j]\}$, such that for every $i \in [q]$,

$$|\mathcal{S}_i| = \frac{|\mathcal{O}_i|}{\beta_j} \leq \frac{\sigma|\mu_x|}{\beta_j} \,, \tag{4.16}$$

where the inequality follows from Eq. 4.15, and each $\mathcal{S}_i$ is a simple daisy of size $|\mathcal{O}_1|/\beta_j$.

Let $\mathcal{O}'_j$ be the multi-collection of all sets $S \in \mathcal{O}_j$ such that $S \setminus K_j \subseteq Q$. In the same manner, for every $i \in [\beta_j]$, let $\mathcal{S}'_i$ be the multi-collection of all sets $S \in \mathcal{S}_i$ such that $S \setminus K_j \subseteq Q$. By the definition of v_j and the fact that $\{\mathcal{S}_i\}$ is a partition

$$v_j(x_\kappa) = \left|\mathcal{O}'_j\right| = \sum_{i=1}^{\beta_j} \left|\mathcal{S}'_i\right|.$$

Since the event $v_1(x_\kappa) \geq \tau_j$ can only occur if $|\mathcal{S}'_i| \geq \frac{\tau_j}{\beta_j}$ for some $i \in [\beta_j]$, we obtain

$$\begin{aligned}
\Pr\left[v_j(x) \geq \tau_j\right] &\leq \Pr\left[\left|\mathcal{S}'_i\right| \geq \frac{\tau_j}{\beta_j} \text{ for some } i \in [\beta_j]\right] \\
&\leq \sum_{i=1}^{\beta_j} \Pr\left[\left|\mathcal{S}'_i\right| \geq \frac{\tau_j}{\beta_j}\right] && \text{(union bound)} \\
&\leq \beta_j \cdot \Pr\left[\left|\mathcal{S}'_1\right| \geq \frac{\tau_j}{\beta_j}\right] \,. && \text{(all } \mathcal{S}_i \text{ have equal size)}
\end{aligned}$$

Now our goal is to show that the event that $|\mathcal{S}_1'| \geq \frac{\tau_j}{\beta_j}$ happens with probability at most $\frac{|\Gamma|^{-|K_1|}}{10q\beta_j}$. Note that this is sufficient for proving the claim because plugging this into the previous equation gives $\Pr\left[v_j(x) \geq \tau_j\right] \leq |\Gamma|^{-|K_j|}/(10q)$.

Since the sets in $\mathcal{S}_1$ are pairwise disjoint, we can and do use the Chernoff bound. In order to do so we first bound the value of $\mathbb{E}[|\mathcal{S}_1'|]$ from above. Recall that the petal of every set $S \in \mathcal{S}_j \subseteq \mathcal{D}_j$ has size j (i.e., $|S \setminus K_j| = j$), and therefore S is in $\mathcal{S}_j'$ with probability exactly p^j. So,

$$\begin{aligned}
\mathbb{E}[|\mathcal{S}_1'|] &= |\mathcal{S}_1| \cdot p^j \\
&\leq \frac{\sigma \cdot |\mu_x| \cdot p^j}{\beta_j} && \text{(by Eq. 4.16)} \\
&= \frac{\tau_j}{2\beta_j}\,. && \left(\text{by Eq. 4.2 and Eq. 4.6, } \sigma = \frac{1}{4q} \text{ and } \tau_j = \frac{|\mu_x|}{2q} \cdot p^j\right)
\end{aligned}$$

We now use the Chernoff bound, stopping at a partial result and providing separate analyses for the cases $j = 1$ and $j > 1$.

$$\begin{aligned}
\Pr\left[|\mathcal{S}_1'| \geq \frac{\tau_j}{\beta_j}\right] &= \Pr\left[|\mathcal{S}_1'| \geq \frac{\tau_j}{\beta_j \mathbb{E}[|\mathcal{S}_1'|]}\mathbb{E}[|\mathcal{S}_1'|]\right] \\
&\leq \exp\left(-\left(\frac{\tau_j}{\beta_j \mathbb{E}[|\mathcal{S}_1'|]} - 1\right)^2 \cdot \frac{\mathbb{E}[|\mathcal{S}_1'|]}{3}\right) && \text{(Chernoff bound)} \\
&\leq \exp\left(-\frac{\tau_j}{3\beta_j}\right) && \text{(explained aferwards)} \\
&= \exp\left(-\frac{|\mu_x| \cdot p^j}{6q\beta_j}\right) && \text{(by Eq. 4.6)} \\
&= \exp\left(-\frac{\ln|\Gamma| np^j}{q\beta_j \cdot \sigma}\right)\,, && \text{(by Eq. 4.3)}
\end{aligned}$$

where the second inequality follows from $\left(\frac{\tau_j}{\beta_j \mathbb{E}[|\mathcal{S}_1'|]} - 1\right)^2 \cdot \frac{\mathbb{E}[|\mathcal{S}_1'|]}{3}$ being minimal when $\mathbb{E}[|\mathcal{S}_1'|]$ is at its upper bound of $\frac{\tau_j}{2\beta_j}$. We next proceed to the first of the two cases.

Take $j = 1$. In this case, by the construction of the daisy partition (Lemma 4.1), every set $S \in \mathcal{O}_1$ has a petal $S \setminus K_1$ of cardinality exactly 1. By the definition of $\mathcal{O}_1$, each set $S \in \mathcal{O}_1$ has at most $\alpha - 1$ other sets $S' \in \mathcal{O}_1$ whose petal intersects the petal of S, i.e., $(S \setminus K_1) \cap (S' \setminus K_1) \neq \varnothing$ (and thus $S \setminus K_1 = S' \setminus K_1$, since both petals have size 1). Therefore, at most $\beta_1 - 1 = \alpha - 1$ distinct sets of $\mathcal{O}_1$ intersect each

$S \in \mathcal{O}_1$, which follows from Eq. 4.10. Now,

$$
\begin{aligned}
&\exp\left(-\frac{\ln|\Gamma|np}{q\alpha\cdot\sigma}\right) \\
&\quad = \exp\left(-\frac{n\cdot p\cdot\rho}{12q}\right) && \text{(by Eq. 4.10)} \\
&\quad = \exp\left(-\gamma\cdot\frac{\rho}{12q}\cdot n^{1-1/2q^2}\right) && \left(p=\gamma\cdot n^{-1/2q^2}\right) \\
&\quad = \frac{1}{10q}\exp\left(-\gamma\cdot n^{1-1/q}\cdot\frac{\rho\cdot n^{\frac{1}{q}-\frac{1}{2q^2}}}{12q}+\ln(10q)\right) \\
&\quad \le \frac{1}{10q}\exp\left(-\ln|\Gamma|\cdot\gamma\cdot q^2\cdot n^{1-1/q}\right) && \text{(large enough } n\text{)} \\
&\quad \le \frac{|\Gamma|^{-|K_1|}}{10q},
\end{aligned}
$$

where the last inequality follows because $|K_1| \le \gamma\cdot q^2\cdot n^{1-1/q}$ by Claim 4.8 (and $\ln|\Gamma| \le \log n$).

Now, take $j > 1$. By Claim 4.5, $\beta_j = 2h(j-1) = 2n^{(j-1)/q}$, which implies the first equality in the following.

$$
\begin{aligned}
\exp\left(-\frac{\ln|\Gamma|\cdot np^j}{q\beta_j\cdot\sigma}\right) &= \exp\left(-\frac{\ln|\Gamma|\cdot np^j}{2q\cdot\sigma}\cdot n^{-\frac{j-1}{q}}\right) \\
&= \exp\left(-2\ln|\Gamma|\cdot p^j\cdot n^{1-\frac{j-1}{q}}\right) && \text{(by Eq. 4.2)} \\
&= \exp\left(-2\ln|\Gamma|\cdot\gamma^j\cdot n^{1-\frac{j-1}{q}-\frac{j}{2q^2}}\right) && \left(p=\gamma\cdot n^{-1/2q^2}\right) \\
&= \exp\left(-2\ln|\Gamma|\cdot\gamma^j\cdot n^{1-\frac{j}{q}+\frac{2q-j}{2q^2}}\right) \\
&\le \frac{1}{10q}\exp\left(-\ln|\Gamma|\cdot\gamma\cdot n^{1-\frac{j}{q}}\cdot 2n^{\frac{1}{2q}}+\ln(10q)\right) && (1<j\le q) \\
&\le \frac{1}{10q}\exp\left(-\ln|\Gamma|\cdot\gamma\cdot q^2\cdot n^{1-j/q}\right) && \text{(large enough } n\text{)} \\
&\le \frac{|\Gamma|^{-|K_j|}}{10q},
\end{aligned}
$$

where the last inequality follows because $|K_j| \le \gamma\cdot q^2\cdot n^{1-j/q}$ by Claim 4.8. □

4.4.3 Concluding the proof

We conclude the proof Theorem 4.7 by applying the two previous claims. Recall that we transformed a ρ-robust local algorithm M for a function f, with query complexity ℓ, into a ρ-robust local algorithm M' with query complexity $q = O(\ell \log \ell)$ and suitable error rate. Then we transformed M' into a sample-based algorithm N with sample complexity $n^{1-1/O(q^2)} = n^{1-1/O(\ell^2 \log^2 \ell)}$, an upper bound guaranteed by the sampling step (Step 1) in the construction of N. It remains to show correctness of the algorithm on every input in the domain of f.

We first consider errors that may arise in the sampling step. By the Chernoff bound, it chooses more than $2pn = 2n^{1-j/(2q^2)}$ points to query and thus outputs arbitrarily with probability at most $1/10$. Otherwise, it proceeds to the next steps.

In the next part of the proof we analyse $v_j(x)$ instead of analyzing v (of Step 2a) in algorithm N; this is sufficient, since by Claim 4.9, they are distributed identically over Q.

Suppose the input $x \in \Gamma^n$ is such that $f(x) = 0$. Since x is ρ-robust, it is in particular $|K_1|/n$-robust (because $|K_1| = o(n)$). Then Claim 4.11 ensures that, for every $j \in [q]$ and kernel assignment κ to K_j, the vote counter satisfies $v_j(x_\kappa) \geq \tau_j$ with probability at most $|\Gamma|^{-|K_j|}/(10q)$. A union bound over all $j \in [q]$ and $|\Gamma|^{|K_j|}$ assignments to the kernel K_j ensures the probability this happens, causing N to output 1 in the threshold check step (Step 2b), is at most $1/10$; otherwise, N will enumerate over every assignment and then (correctly) output 0 in Step 3.

Now suppose $x \in \Gamma^n$ is such that $f(x) = 1$. Then Claim 4.10 ensures that, for some $j \in [q]$, the kernel assignment $\kappa = x_{|K_j}$ will make the vote count satisfy $v_j(x) \geq \tau_j$ with probability at least $9/10$, in which case N (correctly) outputs 1 in the threshold check step (Step 2b).

Therefore, N proceeds beyond the sampling step with probability $9/10$ and outputs correctly (due to Claim 4.11 and Claim 4.10) with probability at least $9/10 - 1/10 \geq 2/3$. This concludes the proof of Theorem 4.7.

Remark 4.8. Notice that the claims actually prove a stronger statement: the failure probability is not merely $1/3$, but *exponentially small*. For each $j \in [q]$, the error probability is

$$\exp\left(-\Omega\left(n^{1-\frac{j}{q}+\frac{2q-j}{2q^2}}\right)\right),$$

but it must withstand a union bound over $\exp\left(O\left(n^{1-j/q}\right)\right)$ events (corresponding to the assignments to the kernel K_j). The smallest slackness is in the case $j = q$, where the success probability is still $\exp\left(-\Omega\left(n^{1/2q}\right)\right)$; this implies that correctness holds for $\exp\left(c \cdot n^{1/2q}\right)$ many executions, if the constant c is sufficiently small. Therefore, *the same samples can be reused for exponentially many runs of possibly different algorithms.*

4.5 Applications

In this section, we derive applications from Theorem 4.7 which range over three fields of study: property testing, coding theory, and probabilistic proof systems. We first give a brief overview of the applications in the following paragraphs, then proceed to the proofs in Sections 4.5.1 to 4.5.3.

Query-to-sample tradeoffs for adaptive testers. The application to property testing is an immediate corollary of Theorem 4.7: since an $\varepsilon/2$-tester is a $(\varepsilon/2, 0)$-robust algorithm for the problem of testing with proximity parameter $\varepsilon/2$, Corollary 4.1 shows that any $\varepsilon/2$-tester making q adaptive queries can be transformed into a sample-based ε-tester with sample complexity $n^{1-1/O(q^2 \log^2 q)}$. In addition, we also show an application to multi-testing (Corollary 4.2).

Relaxed LDC lower bound. By a straightforward extension of our definition of robust local algorithms to allow for outputting a special failure symbol $\perp$, our framework captures *relaxed* LDCs (see Section 4.5.2). We remark that, although standard LDCs have *two-sided* robustness, the treatment of relaxed LDCs is analogous to one-sided robust algorithms.

By applying Theorem 4.7 to a relaxed local decoder *once for each bit to be decoded*, we obtain a *global* decoder that decodes uncorrupted codewords with $n^{1-1/O(q^2 \log^2 q)}$ queries; by a simple information-theoretic argument, we obtain a rate lower bound of $n = k^{1+1/O(q^2 \log^2 q)}$ for relaxed LDCs (see Corollaries 4.3 and 4.4).

Tightness of the separation between MAPs and testers. Theorem 4.7 applies to the setting of *Merlin-Arthur proofs of proximity* (MAPs) via a description of MAPs as coverings by partial testers (Claim 5.1). In Section 4.5.3, we show that the existence of an adaptive MAP for a property Π with query complexity q and proof length m implies the existence of a sample-based tester for Π with sample complexity $m \cdot n^{1-1/O(q^2 \log^2 q)}$ (Theorem 4.11).

This implies that there exists no property admitting a MAP with query complexity $q = O(1)$ and logarithmic proof length (in fact, much longer proof length) that requires at least $n^{1-1/\omega(q^2 \log^2 q)}$ queries for testers, showing the (near) tightness of the separation from [GR18].

Optimality of Theorem 4.7. We conclude Section 4.5.3 with a direct corollary of the tightness of the aforementioned separation between MAPs and testers of [GR18], we obtain that the general transformation in Theorem 4.7 is optimal, up to a quadratic gap in the dependency on the sample complexity. This follows simply because a transformation with smaller sample complexity could have been used to improve Theorem 4.11, yielding a tester with query complexity that contradicts the lower bound (see Theorem 4.12).

4.5.1 Query-to-sample tradeoffs for adaptive testers

Recall that a property tester T for property $\Pi \subseteq \Gamma^n$ is an algorithm that receives explicit access to a proximity parameter $\varepsilon > 0$, query access to $x \in \Gamma^n$ and *approximately decides* membership in Π: it accepts if $x \in \Pi$ and rejects if x is ε-far from Π, with high probability.

By Claim 4.1, an ε-tester with $\varepsilon \in (0,1)$ is an ε-robust local algorithm that computes the function $f\colon \Pi \cup \overline{B_{2\varepsilon}(\Pi)} \to \{0,1\}$ defined as follows.

$$f(x) = \begin{cases} 1, & \text{if } x \in \Pi \\ 0, & \text{if } x \in \overline{B_{2\varepsilon}(\Pi)}. \end{cases}$$

Note, moreover, that a local algorithm that solves f is by definition a 2ε-tester, accepting elements of Π and rejecting points that are 2ε-far from it with high probability. A direct application of Theorem 4.7 thus yields the following corollary, which improves upon the main result of [FLV15], by extending it to the two-sided adaptive setting.

Corollary 4.1. *For every fixed $\varepsilon > 0, q \in \mathbb{N}$, any ε-testable property of strings in Γ^n with q queries admits a sample-based 2ε-tester with sample complexity $n^{1-1/O(q^2 \log^2 q)}$.*

This also immediately extends an application to multitesters in [FLV15]. By standard error reduction, for any $k \in \mathbb{N}$, an increase of the sample complexity by a factor of $O(\log k)$ ensures each member of a collection of k sample-based testers errs with probability $1/(3k)$. A union bound allows us to *reuse the same samples for all testers*, so that all will output correctly with probability $2/3$. Taking $k = \exp\left(n^{1/\omega(q^2 \log^2 q)}\right)$, the sample complexity becomes $n^{1-1/O(q^2 \log^2 q)} \cdot n^{1/\omega(q^2 \log^2 q)} = o(n)$, which yields the following corollary.

Corollary 4.2. *If a property $\Pi \subseteq \Gamma^n$ is the union of $k = \exp\left(n^{1/\omega(q^2 \log^2 q)}\right)$ properties $\Pi_1, \dots, \Pi_k$, each ε-testable with q queries, then Π is 2ε-testable via a sample-based tester with sublinear sample complexity.*

A tester for the union simply runs all (sub-)testers, accepting if and only if at least one of them accepts. A proof for a generalisation of this corollary, which holds for *partial testers*, is given in the Section 4.5.3.

4.5.2 Stronger relaxed LDC lower bounds

Relaxed LDCs are codes that relax the notion of LDCs by allowing the local decoder to abort on a small fraction of the indices, yet crucially still avoid errors. This seemingly modest relaxation turns out to allow for dramatically better parameters (an exponential improvement on the rate of the best known $O(1)$-query LDCs). However, since these algorithms are much stronger, obtaining lower bounds on relaxed LDCs is significantly harder than on standard LDCs. Indeed, the first lower bound on relaxed LDCs [GL21] was only shown more than a decade after the notion was introduced; this bound shows that to obtain query complexity q, a relaxed LDC $C\colon \{0,1\}^k \to \{0,1\}^n$ must have blocklength

$$n \geq k^{1+\frac{1}{O(2^{2q}\cdot\log^2 q)}} ,$$

In this section, we use Theorem 4.7 to obtain an improved lower bound with an exponentially better dependency on the query complexity. We begin by recalling the definition of relaxed LDCs.

Definition 4.12 (Definition 5.4, restated)**.** *A code $C\colon \{0,1\}^k \to \{0,1\}^n$ whose distance is δ_C is a q-local relaxed LDC with success rate ρ, decoding radius $\delta \in (0, \delta_C/2)$ and error rate $\sigma \in (0, 1/3]$ if there exists a randomised algorithm D, known as a* relaxed decoder*, that, on input $i \in [k]$, makes at most q queries to an oracle w and satisfies the following conditions.*

1. *Completeness: For any $i \in [k]$ and $w = C(x)$, where $x \in \{0,1\}^k$,*
$$\Pr[D^w(i) = x_i] \geq 1 - \sigma .$$
2. *Relaxed Decoding: For any $i \in [k]$ and $w \in \{0,1\}^n$ that is δ-close to a (unique) codeword $C(x)$,*
$$\Pr[D^w(i) \in \{x_i, \perp\}] \geq 1 - \sigma .$$
3. *Success Rate: There exists a constant $\rho > 0$ such that, for any $w \in \{0,1\}^n$ that is δ-close to a codeword $C(x)$, there exists a set $I_w \subseteq [k]$ of size at least ρk such that for every $i \in I_w$,*
$$\Pr[D^w(i) = x_i] \geq 2/3 .$$

Remark 4.9. The first two conditions imply the latter, as shown by [BGH+06]. Therefore, it is not necessary to show the success rate condition when verifying that an algorithm D is a relaxed local decoder.

Note that, whenever D^w outputs $\perp$, it detected that the input is *not valid*, since it is inconsistent with *any* codeword $C(x)$. We slightly generalise local algorithms (Definition 4.2) to capture this behaviour, by allowing them to output $\perp$ as well as the correct function evaluation $f(z, x)$ (except for a prescribed set of valid inputs). Formally,

Definition 4.13 (Relaxed local algorithm)**.** *Let Γ be a finite alphabet, Z a finite set and $\{\mathcal{P}_z : z \in Z\}$ a family of sets $\mathcal{P}_z \subseteq \Gamma^n$ indexed by Z. With $\mathcal{P} \coloneqq \{(z, x) : z \in Z, x \in \mathcal{P}_z\}$, let $f \colon \mathcal{P} \to \{0,1\}$ be a partial function.*

A relaxed q-local algorithm *M* for computing *f with* valid input set *$V \subseteq \Gamma^n$ and* error rate *σ receives explicit access to $z \in Z$, query access to $x \in \mathcal{P}_z$, makes at most q queries to x and satisfies*

$$\Pr\left[M^x(z) \in \{f(z,x), \perp\}\right] \geq 1 - \sigma.$$

Moreover, if $x \in V$, then M satisfies

$$\Pr[M^x(z) \in f(z,x)] \geq 1 - \sigma.$$

We shall also need to generalise the notion of robustness (Definition 4.3) accordingly.

Definition 4.14 (Robustness)**.** *Let $\rho > 0$. A local algorithm M for computing $f \colon \mathcal{P} \to \{0,1\}$ is* ρ-robust *at the point $(z, x) \in \mathcal{P}$ if* $\Pr[M^w(z) \in \{f(z,x), \perp\}] \geq 1 - \sigma$ *for all $w \in B_\rho(x)$. We say that M is* (ρ_0, ρ_1)-robust *if, for all $z \in Z$ and $b \in \{0,1\}$, M is ρ_b-robust at every x such that $f(z,x) = b$.*

We remark that robustness for algorithms that allow aborting allows the correct value to change to $\perp$ (but, crucially, not to the wrong value) even if only one bit is changed. This makes the argument more involved than an argument for LDCs, and indeed, our theorem for *relaxed* LDCs relies on the full machinery of Theorem 4.7.

Note that an algorithm that ignores its input and always outputs $\perp$ fits both definitions above, but has no valid inputs and clearly does not display any interesting behaviour. We also remark that the set of valid inputs captures completeness (but *not* the success rate) in the case of relaxed LDCs.

With these extensions, a relaxed local decoder D with decoding radius δ fits the definition of a (relaxed) local algorithm that receives $i \in [k]$ as explicit input, where the code C comprises the valid inputs and every $x \in C$ is δ-robust for D.

While a relaxed local algorithm is very similar in flavour to a standard local algorithm, it may not be entirely clear whether a transformation analogous to Theorem 4.7 holds in this case as well. We next show that one indeed does: with small modifications to the algorithm constructed in Section 4.4.1, we leverage the same analysis of Section 4.4.2 to prove the following variant of Theorem 4.7.

Theorem 4.10. *Suppose there exists a (ρ_0, ρ_1)-robust relaxed local algorithm M for computing the function $f\colon \mathcal{P} \to \{0,1\}$ (where $Z \times \Gamma^n$) with query complexity $\ell = O(1)$ and $\rho_0, \rho_1 = \Omega(1)$. Let $V \subseteq \Gamma^n$ be the valid inputs of M. Then, there exists a* sample-based *relaxed local algorithm N for f with sample complexity $n^{1-1/O(\ell^2 \log^2 \ell)}$ with the same set V of valid inputs.*

Proof. Throughout the proof, we assume the explicit input to be fixed and omit it from the notation. First, note that error reduction (Claim 4.6) and randomness reduction (Claim 4.7) apply in the relaxed setting: the analysis is identical on valid inputs, and holds likewise for the remainder of the domain of f (with correctness of M relaxed to be $M^x \in \{f(x), \perp\}$). Thus Lemma 4.4 enables the transformation of M into another robust algorithm M' with small error rate that uniformly samples a decision tree from a multi-collection of small size.

Recall that the construction of the sample-based algorithm in Section 4.4.1 uses a collection of triplets obtained from the behaviour of M' when it outputs 1. A corresponding collection can be obtained for the case where M' outputs 0. Denote by $\mathcal{T}_b$ the collection that corresponds to output $b \in \{0,1\}$, and let N_b be the sample-based algorithm that

- uses the triplets $\mathcal{T}_b$ to construct its daisy partition in the preprocessing step;
- outputs b if the counter crosses the threshold in Step 2b; and
- outputs $\perp$ in Step 3 if the threshold is never reached;

but is otherwise the same as the construction of Section 4.4.1.

The analysis of Section 4.4.2 applies to N_b^x: if $x \in V$, the analysis of Claim 4.10 is identical; while if x is robust and $f(x) = \neg b$, Claim 4.11 requires a lower bound on the probability that M' outputs b when its input is x (and enables an application of the volume lemma), which holds by the definition of error rate of a relaxed local algorithm. Therefore, the probability of each the following events is bounded by 1/10:

(i) N_b^x outputs arbitrarily in the sampling step;
(ii) N_b^x outputs $\perp$ when $f(x) = b$; and
(iii) N_b^x outputs b when x is robust and $f(x) = \neg b$.

Finally, the relaxed sample-based algorithm N simply executes the sampling step of N_0 then the enumeration steps of N_0 and N_1 on these samples, outputs b if one of N_b outputs b, and outputs $\perp$ otherwise. Then, $N^x = f(x)$ if $x \in V$ and $N^x = \perp$ if $x \notin V$, with probability $7/10 \geq 2/3$. $\square$

By casting a relaxed decoder as a robust relaxed local algorithm and applying Theorem 4.10, we obtain the following corollary.

Corollary 4.3. *Any binary code $C\colon \{0,1\}^k \to \{0,1\}^n$ that admits a relaxed local decoder D with decoding radius δ and query complexity $q = O(1)$ also admits a sample-based relaxed local decoder D' with decoding radius $\delta/2$ and sample complexity $n^{1-1/O(q^2 \log^2 q)}$.*

We are now ready to state the following corollary, which improves on the previous best rate lower bound for relaxed LDCs [GL21] by an application of the theorem above to the setting of relaxed local decoding. This follows from the construction of a global decoder (which is able to decode the entire message) that is only guaranteed to succeed with high probability when its input is *a perfectly valid codeword.*

Corollary 4.4. *Any code $C\colon \{0,1\}^k \to \{0,1\}^n$ that is relaxed locally decodable with $q = O(1)$ queries satisfies*

$$n = k^{1+\frac{1}{O(q^2 \log^2 q)}}.$$

Proof. Let D' be the sample-based relaxed LDC with sample complexity q' obtained by Corollary 4.3 from a relaxed LDC with query complexity q for the code C. Reduce the error rate of D' to $1/3k$ by repeating the algorithm $O(\log k)$ times and taking the majority output, thus increasing the sample complexity to $O(q' \cdot \log k) = n^{1-1/t}$ with $t = O(q^2 \log^2 q)$.

Now, consider the *global decoder G* defined as follows: on input w, execute the sampling stage once and the enumeration stages of $D^w(1), \ldots, D^w(k)$ on the same samples. A union bound ensures that, with probability at least $2/3$, the outputs satisfy $D^w(i) = x_i$ for all i if $w = C(x)$.

The global decoder G obtains k bits of information from $n^{1-1/t}$ bits with probability above $1/2$. Information-theoretically, we must have

$$k \leq \frac{n^{1-1/t}}{2} = \frac{n^{\frac{t-1}{t}}}{2},$$

so that $n \geq 2k^{1+\frac{1}{t-1}}$. Since $t = O(q^2 \log^2 q)$, we have $n = k^{1+1/O(q^2 \log^2 q)}$. $\square$

4.5.3 A maximal separation between testers and proofs of proximity

Recall that a Merlin-Arthur proof of proximity (MAP, for short) for property Π is a local algorithm that receives explicit access to a proximity parameter $\varepsilon > 0$ and a purported proof string π, as well as query access to a string $x \in \Gamma^n$. It uses the information encoded in π to decide which coordinates of x to query, accepting if $x \in \Pi$ and π is a valid proof for x, and rejecting if x is ε-far from Π. (In particular, a MAP with proof length 0 is simply a tester.) In Claim 5.1, we showed that MAPs and coverings by partial testers are equivalent, a fact that we shall use shortly.

As discussed in the Section 1.1, one of the most fundamental questions regarding proofs of proximity is their relative strength in comparison to testers; that is, whether verifying a proof for an approximate decision problem can be done significantly more efficiently than solving it. This can be cast as an analogue of the P versus NP question for property testing.

Fortunately, in the setting of property testing, the problem of verification versus decision is very much tractable: one of the main results in [GR18] shows the existence of a property Π which: (1) admits a MAP with proof length $O(\log n)$ and query complexity $q = O(1)$; and (2) requires at least $n^{1-1/\Omega(q)}$ queries to be tested without access to a proof. (The lower bound of [GR18] is stated in a slightly weaker form. However, it is straightforward to see that the stronger form holds; see discussion at the end of this section.)

While this implies a nearly exponential separation between the power of testers and MAPs, it remained open whether the aforementioned sublinear lower bound on testing is an artefact of the techniques, or whether it is possible to obtain a stronger separation, where the property is harder for testers.

Claim 5.1 and Theorem 4.7 allow us to prove the following corollary, which shows that the foregoing separation is nearly tight.

Theorem 4.11. *If a property $\Pi \subseteq \Gamma^n$ admits a MAP with query complexity q, proof length m and proximity parameter $\varepsilon = \Omega(1)$, then it admits a sample-based 2ε-tester with sample complexity $m \cdot n^{1-1/O(q^2 \log^2 q)}$.*

Applying Theorem 4.11 to the special case of MAPs with *logarithmic* proof length, we obtain a sample-based tester with sample complexity $n^{1-1/O(q^2 \log^2 q)}$, showing that the separation in [GR18] is nearly optimal, and in particular that there cannot be a fully exponential separation between MAPs and testers.

Proof of Theorem 4.11. Let Π be a property and T be a MAP with proof length m as in the statement. By Claim 5.1, there exists a collection of partial testers

$\{T_i : i \leq |\Gamma|^m\}$ with query complexity q that satisfy the following. Each T_i accepts inputs in a property Π_i and rejects inputs that are ε-far from Π, with $\Pi \subseteq \cup_i \Pi_i$. By applying Corollary 4.1 to each of these testers, we obtain a collection of sample-based testers $\{S_i\}$ with sample complexity $q' = n^{1-1/O(q^2 \log^2 q)}$ for the same partial properties, but which only reject inputs that are 2ε-far from Π.

The execution of each of the S_i proceeds in two steps, as defined in Section 4.4.1: *sampling* (Step 1) and *enumeration* (Step 2). Note that the sampling step is exactly the same for every S_i.

Let $k = O(m \log |\Gamma|)$ such that taking the majority output from k repetitions of S_i yields an error rate of $1/(3|\Gamma|^m)$. We define a new sample-based algorithm S that repeats the following steps k times:

1. Execute both steps of S_1 (sampling and enumeration), recording the output.
2. For all $1 < i \leq |\Gamma|^m$, *only execute the enumeration step* of S_i on the samples obtained in Item 1, and record the output.

After all k iterations have finished, check if at least $k/2$ outputs of S_i were 1 for some i. If so, output 1, and output 0 otherwise.

First suppose S receives an input $x \in \Pi$, and let $i \leq |\Gamma|^m$ such that $x \in \Pi_i$. Then the majority output of the enumerations steps of S_i is 1 with probability $1 - 1/(3|\Gamma|^m) \geq 2/3$. Now suppose S receives an input x that is 2ε-far from Π. Then, for each i, the majority output of the enumeration step of S_i is 1 with probability at most $1/(3|\Gamma|^m)$. A union bound over all $i \leq |\Gamma|^m$ ensures this happens with probability at least $1/3$, in which case S correctly outputs 0.

S is therefore a 2ε-tester for the property Π with sample complexity $k \cdot q' = m \cdot n^{1-1/O(q^2 \log^2 q)}$ (recall that $\log |\Gamma| \leq \log n$), and the theorem follows. $\square$

Interestingly, as a direct corollary of Theorem 4.11, we obtain that the general transformation in Theorem 4.7 is optimal, up to a quadratic gap in the dependency on the sample complexity, as a transformation with a smaller sample complexity could have been used to transform the MAP construction in the MAPs-vs-testers separation of [GR18], yielding a tester with query complexity that contradicts the lower bound in that result.

Theorem 4.12. *There does* not *exist a transformation that takes a robust local algorithm with query complexity q and transforms it into a* sample-based *local algorithm with sample complexity at most $n^{1-1/o(q)}$.*

Proof. Let Π be the *encoded intersecting messages* property considered in [GR18, Section 3.1], for which it was shown that Π has a MAP with query complexity q and logarithmic proof complexity, but every tester for Π requires at least $n^{1-1/\Omega(q)}$ queries. Suppose towards contradiction that a transformation as in the hypothesis exists. Then, applying the transformation to the aforementioned MAP (as in Theorem 4.11) yields a tester for Π with query complexity $n^{1-1/o(q)}$, in contradiction to the lower bound. $\square$

On the lower bound in [GR18]. The separation between MAPs and testers in [GR18] is proved with respect to a property of strings that are encoded by relaxed LDCs; namely, the *encoded intersecting messages property*, defined as

$$\mathrm{EIM}_C = \left\{ \big(C(x), C(y)\big) \;:\; \begin{array}{c} x, y \in \{0,1\}^k,\ k \in \mathbb{N} \text{ and} \\ \exists i \in [k] \text{ s.t. } x_i \neq 0 \text{ and } y_i \neq 0 \end{array} \right\},$$

where $C\colon \{0,1\}^k \to \{0,1\}^n$ is a code with linear distance, which is both a relaxed LDC and an LTC. In [GR18] it is shown that there exists a MAP with proof length $O(\log n)$ and query complexity $q = O(1)$, and crucially for us, that any tester requires $\Omega(k)$ queries to be tested without access to a proof. The best constructions of codes that satisfy the aforementioned conditions [BGH$^+$06, CGS22, AS21] achieve blocklength $n = O(k^{1+1/q}) = k^{1+1/\Omega(q)}$, and hence the stated lower bound follows.

Chapter 5

Quantum proofs of proximity

Overview

We initiate the systematic study of QMA algorithms in the setting of property testing, to which we refer as *QMA proofs of proximity* (QMAPs). These are quantum query algorithms that receive explicit access to a sublinear-size untrusted proof and are required to accept inputs having a property Π and reject inputs that are ε-far from Π, while only probing a minuscule portion of their input.

We investigate the complexity landscape of this model, showing that QMAPs can be *exponentially* stronger than both classical proofs of proximity and quantum testers. To this end, we extend the methodology of Blais, Brody, and Matulef (Computational Complexity, 2012) to prove quantum property testing lower bounds via reductions from communication complexity. This also resolves a question raised in 2013 by Montanaro and de Wolf (cf. Theory of Computing, 2016).

Our algorithmic results include an algorithmic framework that enables quantum speedups for testing an expressive class of properties, namely, those that are succinctly *decomposable*. A consequence of this framework is a QMA algorithm to verify the parity of an n-bit string with $O(n^{2/3})$ queries and proof length. We also show a QMA algorithm for testing graph bipartiteness, a property that lies outside of this family and yet admits a quantum speedup.

Organisation

The rest of this chapter is organised as follows. QMA proofs of proximity are formally defined in Section 5.1, and we prove our complexity class separations in Section 5.2 (which Sections 5.7 and 5.8 complement with separations implied by known results). Section 5.3 proves lower bounds for QMAPs and concludes the complexity-theoretic part of the chapter. Proceeding to the algorithmic part, we show in Section 5.4 that

QMAP protocols enable speedups for proximity-oblivious MAPs. In Section 5.5, we define decomposability and prove the bulk of our algorithmic results, including exact decision problems as a special case. Finally, in Section 5.6 we show a QMAP protocol for testing graph bipartiteness.

5.1 Definition

This section provides a formal definition of QMAPs.

A *quantum Merlin-Arthur proof of proximity* (QMAP) for a property $\Pi = \bigcup \Pi_n$ is a proof system consisting of a quantum algorithm V, called a verifier, that is given as explicit input an integer $n \in \mathbb{N}$ and a proximity parameter $\epsilon > 0$. It has oracle access to a unitary $U \in \mathcal{V} \subseteq \mathcal{U}(2^n)$ acting on n qubits, which belongs to a *universe* $\mathcal{V}$ with an associated distance measure.[1] Furthermore, the verifier receives a p-qubit quantum state ρ explicitly as a purported proof that $U \in \mathcal{V}$.

The verifier V receives n and ε as inputs, and outputs a sequence of unitary operators $V_0 \dots V_q$ that satisfies the two following conditions.

1. *Completeness.* For every $n \in \mathbb{N}$ and $U \in \Pi_n$, there exists a p-qubit quantum state $|\psi\rangle$ such that,[2] for every proximity parameter $\epsilon > 0$,

$$\mathbb{P}\left[V^U(n, \varepsilon, |\psi\rangle) = 1\right] \geq 2/3 \ .$$

   Equivalently, *some* p-qubit quantum state $|\psi\rangle$ satisfies

$$\|(|1\rangle\langle 1| \otimes I) W(|\psi\rangle \otimes |\mathbf{0}\rangle)\|^2 \geq 2/3 \ ,$$

   where $|\mathbf{0}\rangle$ is the initial state of the verifier and W the unitary obtained by interspersing q calls to the oracle U between the V_i; that is, $W = V_q(U \otimes I)V_{q-1} \dots (U \otimes I)V_0$.

2. *Soundness.* For every $n \in \mathbb{N}$, $\varepsilon > 0$ and p-qubit quantum state $|\psi\rangle$, if $U \in \mathcal{U}(2^n)$ is ε-far from Π_n, then

$$\mathbb{P}\left[V^U(n, \varepsilon, |\psi\rangle) = 1\right] \leq 1/3 \ .$$

[1] Note that this definition generalises classical properties, where $\mathcal{V} = \{U_x : x \in \{0,1\}^n\} \subset \mathcal{U}(2^{n+1})$, the unitary U_x acts as $U_x |i\rangle |b\rangle = |i\rangle |b \oplus x_i\rangle$, and the distance between U_x and U_y is the Hamming distance between x and y.

[2] While the proof can also be a mixed state, assuming it to be pure is without loss of generality; see Remark 5.1.

Equivalently, *every* p-qubit quantum state $|\psi\rangle$ satisfies

$$\|(|1\rangle\langle 1| \otimes I)W(|\psi\rangle \otimes |\mathbf{0}\rangle)\|^2 \le 1/3 ,$$

where $W = V_q(U \otimes I)V_{q-1}\dots(U \otimes I)V_0$.

The *query complexity* of a QMAP is number of times the verifier calls the oracle U. More precisely, the query complexity is $q = q(n,\varepsilon)$ if, for every $n \in \mathbb{N}$, $\epsilon > 0$ and $U \in \mathcal{U}(2^n)$, the verifier makes at most q queries to the input. Its *proof complexity* is $p = p(n,\varepsilon)$ if, for every $n \in \mathbb{N}$ and $U \in \Pi_n$, there exists a 2^p-dimensional quantum state $|\psi\rangle$ satisfying both of the above conditions.

Definition 5.1 (QMAP complexity class)**.** *Fix a universe set of unitary operators $\mathcal{V}$ and distance measure $d : \mathcal{V} \times \mathcal{V} \to [0,1]$.* $\mathsf{QMAP}(\varepsilon,p,q)$ *is the class of properties $\Pi \subseteq \mathcal{V}$ that admit a verifier for proximity parameter ε with query complexity q and proof complexity p..*

The complexity class $\mathsf{QCMAP}(\varepsilon,p,q)$ *is defined as above, with the additional restriction that the proof be* classical*, i.e., that the p-qubit quantum state given as proof is a computational basis state.*

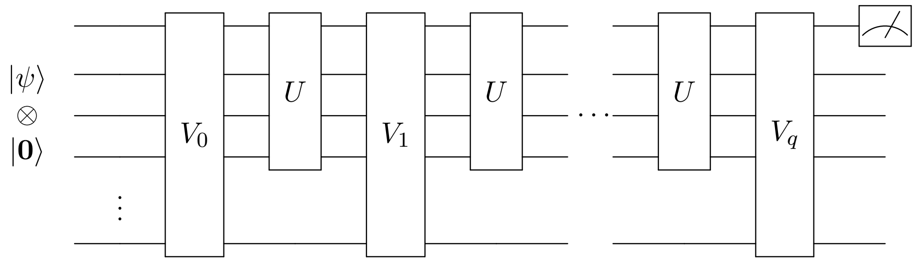


Figure 5.1: Schematic of a QMAP protocol that receives a proof state $|\psi\rangle$ and makes q queries to a unitary U.

We also denote the class of properties that are ε-testable by quantum testers with q queries as $\mathsf{QPT}(\varepsilon,q)$, that is, $\mathsf{QPT}(\varepsilon,q) := \mathsf{QMAP}(\varepsilon,0,q)$.

Remark 5.1 (Proofs are pure states)**.** Without loss of generality, the quantum state given as the proof is a pure state on p qubits, i.e., a rank one positive semi-definite matrix. To see why, note that, if some mixed state $\rho = \sum p_i |\psi_i\rangle\langle\psi_i|$ causes the verifier to accept with probability 2/3, then, by convexity, there exists a state $|\psi_k\rangle$ in that mixture that would also cause the verifier to output 1 with probability at least 2/3. Hence, the proof can be the pure state $|\psi_k\rangle$. Likewise, if no pure state can make the verifier accept with probability larger than 1/3, the same holds for mixed states.

5.2 Complexity separations

Armed with a formal definition of QMAPs, we begin to chart the landscape of complexity classes to which quantum proofs of proximity belong. In Section 5.2.1, we provide definitions that will be necessary in the remainder of the section, mainly pertaining to coding theory.

Our main goal is to prove Theorem 5, namely, that QMAPs can exploit quantum resources and the availability of a proof to gain expressivity that neither can provide separately. This theorem follows from the *incomparabilty* between the classes MAP and QPT: in Section 5.2.2, we exhibit a property Π_B that is easy to test classically with a short proof, but requires many queries (without a proof) even for a quantum tester (Theorem 5.5); moreover, in Section 5.2.3, we show the existence of a property Π_F that is easily testable quantumly but difficult to test classically, even with the aid of a proof (Theorem 5.6).

The aforementioned results immediately imply the existence of a property, namely $\Pi_B \times \Pi_F$, which does not admit efficient MAPs nor quantum testers, requiring large proof or query complexity, whereas a QMAP with logarithmic proof and query complexities does exist (indeed, one with a *classical* proof).

Theorem 5.2 (Theorem 5, restated)**.** *There exists a property $\Pi \subseteq \{0,1\}^n$ such that, for any small enough constant $\varepsilon > 0$,*

$$\Pi \in \mathsf{QCMAP}(\varepsilon, \log n, O(1))$$

and

$$\Pi \notin \mathsf{QPT}(\varepsilon, o(n^{0.49})) \cup \mathsf{MAP}(\varepsilon, p, q)$$

when $p \cdot q = o(n^{1/4})$.

5.2.1 Preliminaries

We first define the necessary notions of local codes that will be used in this section. We denote throughout a finite field of constant size by $\mathbb{F}$.

Definition 5.2 (Locally Testable Codes (LTCs))**.** *A code $C \colon \mathbb{F}^k \to \mathbb{F}^n$ is locally testable, with respect to proximity parameter ε and error rate σ, if there exists a probabilistic algorithm T that makes q queries to a purported codeword w such that:*

1. *If $w = C(x)$ for some $x \in \mathbb{F}^k$, then $\mathbb{P}[T^w = 1] \geq 1 - \sigma$.*
2. *For every w that is ε-far from C, we have $\mathbb{P}[T^w = 0] \geq 1 - \sigma$.*

Note that the algorithm T that an LTC admits is simply an ε-tester for the property of being a valid codeword of C.

Definition 5.3 (Locally Decodable Codes (LDCs))**.** *A code $C\colon \mathbb{F}^k \to \mathbb{F}^n$ is locally decodable with* decoding radius δ *and error rate σ if there exists a probabilistic algorithm D that given index $i \in [k]$ makes q queries to a string w promised to be δ-close to a codeword $C(x)$, and satisfies*

$$\mathbb{P}[D^w(i) = x_i] \geq 1 - \sigma.$$

Since the best known constructions of LDCs have superpolynomial blocklength, we will make use of a relaxation of this type of code that allows for much more efficient constructions and suffices for our purposes.

Definition 5.4 (Relaxed LDCs)**.** *A code $C\colon \mathbb{F}^k \to \mathbb{F}^n$ with relative distance δ_C is a q-local relaxed LDC with success rate ρ and decoding radius $\delta \in (0, \delta_C/2)$ if there exists a randomised algorithm D, known as a* relaxed decoder *that, on input $i \in [k]$, makes at most q queries to an oracle w and satisfies the following conditions.*

1. *Completeness: For any $i \in [k]$ and $w = C(x)$, where $x \in \mathbb{F}^k$,*

$$\mathbb{P}[D^w(i) = x_i] \geq 2/3.$$

2. *Relaxed Decoding: For any $i \in [k]$ and any $w \in \mathbb{F}^n$ that is δ-close to a (unique) codeword $C(x)$,*

$$\mathbb{P}[D^w(i) \in \{x_i, \perp\}] \geq 2/3.$$

3. *Success Rate: There exists a constant $\rho > 0$ such that, for any $w \in \mathbb{F}^n$ that is δ-close to a codeword $C(x)$, there exists a set $I_w \subseteq [k]$ of size at least ρk such that for every $i \in I_w$,*

$$\mathbb{P}[D^w(i) = x_i] \geq 2/3\ .$$

As shown by [BGH+06, GGK19, CGS22, AS21], there exist linear codes of only slightly superlinear blocklength that are both locally testable *and* relaxed locally decodable:[3]

Theorem 5.3. *For any constant $\gamma > 0$, there exist linear codes over $\mathbb{F}$ with blocklength $n = k^{1+\gamma}$ that are relaxed locally decodable with $O(1)$ queries with respect to decoding*

[3]We note that while LTCs and RLDCs are usually defined with respect to *binary* alphabets, they can be constructed over larger fields as well as fields of odd characteristic. The constructions essentially rely on two components: a base (linear) code and a PCP of proximity, both of which readily extend to larger alphabets.

radius $\delta = \Omega(1)$. Moreover, given any constant proximity parameter $\varepsilon \in (0, \delta]$, the code is also locally testable with $O(1)$ queries.

Moreover, the tester and local decoder for these codes are one-sided (i.e., always accept when given a valid codeword as input), and the blocklength cannot be improved to linear (see Chapter 4).

Communication complexity. In the model of quantum communication complexity, two parties with unbounded computational power aim to compute a joint predicate by communicating the smallest number of qubits with each other. Alice knows $x \in \{0,1\}^k$, Bob knows $y \in \{0,1\}^k$ and both hold a function $f : \{0,1\}^{2k} \to \{0,1\}$, and, by communicating qubits with each other, they must compute $f(x, y)$ with bounded probability of error. The *communication complexity* of f is the worst-case number qubits that need to be communicated in order to compute $f(x, y)$ over all x, y, minimised over all communication protocols.

We will make use of the well-known communication complexity problem of *disjointness.*

Definition 5.5. *Let $x, y \in \{0,1\}^k$ and $S, T \subseteq [k]$ be sets whose indicator vectors are x and y, respectively; i.e., $S = \{i \in [k] : x_i = 1\}$ and $T = \{i \in [k] : y_i = 1\}$. Then* $\mathrm{disj}_k(x, y) = 1$ *if and only if S and T are disjoint, that is,*

$$\mathrm{disj}_k(x,y) = \begin{cases} 1 & \textit{if } S \cap T = \varnothing \\ 0 & \textit{otherwise} \end{cases} = \begin{cases} 1 & \textit{if } x_i = 0 \textit{ or } y_i = 0 \textit{ for all } i \in [k], \\ 0 & \textit{otherwise.} \end{cases}$$

This problem is known to be hard for quantum communication protocols, requiring $\Omega(\sqrt{k})$ qubits of communication, as shown in [Raz03].

5.2.2 MAPs versus quantum testers

We now set out to prove Theorem 5.5, which shows a property Π_B that is efficiently testable with a short classical proof (Lemma 5.1) but for which a quantum tester must make a large number of queries (Lemma 5.2).

The property in question is defined as follows. First, consider a linear code $C : \mathbb{F}^k \to \mathbb{F}^n$ with $n = k^{1.001}$, over a field $\mathbb{F}$ of odd characteristic and size $O(1)$, that is both *relaxed locally decodable* with $O(1)$ queries and decoding radius $\delta = \Omega(1)$; as well as *locally testable* with $O(1)$ queries for any constant proximity parameter $\varepsilon \in (0, \delta]$ (recall that Theorem 5.3 shows that codes with these parameters exist).

The property Π_B is comprised of encodings of *non-boolean* messages, that is:

$$\Pi_B = \left\{ C(z) : z \in \mathbb{F}^k \setminus \{0,1\}^k \right\} .$$

We first show that the property Π_B is efficiently testable via a MAP protocol with a short proof.

Lemma 5.1. *For any constant $\varepsilon \in (0, \delta]$, $\Pi_B \in \mathsf{MAP}(\varepsilon, \log n, O(1))$.*

Proof. The verifier will follow the following strategy to test Π_B: first, test whether the input is ε-close to the code C (which can be accomplished with $O(1)$ queries due to the local testability of C). If the tester rejects, then not only is the input far from Π_B, but from all of C, in which case it rejects.

Except with small probability, if the tester accepts the input is δ-close to C, so we may locally decode any coordinate of the message (also with $O(1)$ queries); using the proof string to determine this location, the verifier then checks if the symbol at that coordinate is boolean-valued.

This strategy is laid out in Algorithm 5.1.

Algorithm 5.1: MAP verifier for Π_B

Input: explicit access to a proximity parameter $\varepsilon > 0$ and a proof string $\pi \in \{0,1\}^{\log n}$, as well as oracle access to $x \in \mathbb{F}^n$.

```
Step 1: Test if the input $w$ is a valid codeword with proximity parameter $\varepsilon$.
        Reject if the test rejects.

Step 2: Interpret the proof as an index $i \in [k]$, locally decode $z_i$ and accept if
        $z_i \in \mathbb{F} \setminus \{0,1\}$. Otherwise, reject.
```

Note that the proof complexity is $\log n$ by definition, and, since both local testing and local decoding have query complexity $O(1)$, the verifier makes $O(1)$ queries in total (note that querying an element of $\mathbb{F}$ requires $O(1)$ *bit* queries, so the complexities of the tester and decoder are still constant in terms of bit queries). Moreover, if $w \in \Pi_B$, then $w = C(z)$ for some $z \in \mathbb{F}^k \setminus \{0,1\}^k$, and local testing succeeds with probability 1; and when the prover specifies a coordinate i such that $z_i \notin \{0,1\}$, local decoding also succeeds with probability 1, so completeness follows.

Now, if w is ε-far from Π_B, then either (1) the input w is ε-far from *any* codeword of the code C; or (2) w is ε-close to some $C(z)$ such that $z \in \{0,1\}^k$.

In the first case, local testing (and thus the verifier) will reject with probability

2/3. In the second case, the testing step may not trigger a rejection, but the local decoder then outputs, regardless of the proof $i \in [k]$, either $\perp$ or $z_i \in \{0,1\}$ with probability 2/3, any of which cause the verifier to reject. $\square$

The next lemma shows that, unlike MAPs, quantum testers cannot test Π_B efficiently.

Lemma 5.2. *Any quantum tester for the property Π_B with constant proximity parameter $\varepsilon \in [0,\delta)$ must have query complexity $\Omega(n^{0.49})$.*

Proof. Recall that in the disjointness problem, Alice is given as input $x \in \{0,1\}^k$, Bob is given $y \in \{0,1\}^k$ and they must compute $\mathrm{disj}_k(x,y)$; and that Π_B is the encoding of non-boolean strings of length $k = n^{\frac{1}{1.001}}$ by the code C.

To show that any quantum tester needs $\Omega(n^{0.49})$ queries to ε-test Π_B, we give a reduction showing that a tester with query complexity q can be used to compute disj_k by communicating $O(q \log n)$ qubits. Since the quantum communication complexity of disj_k is $\Omega(\sqrt{k}) = \Omega(n^{\frac{1}{2}\cdot\frac{1}{1.001}}) = \Omega(n^{0.499})$, the query complexity of the quantum tester follows.

First, Alice and Bob use C to encode x and y, respectively. Now Alice holds $C(x) \in \mathbb{F}^n$ and Bob holds $C(y) \in \mathbb{F}^n$. Note that, defining $z \coloneqq x + y \in \mathbb{F}^k$, we have $\mathrm{disj}(x,y) = 0 \iff z \notin \{0,1\}^k \iff C(z) \in \Pi_B$ (recall that the characteristic of $\mathbb{F}$ is larger than 2, so that if $x_i = y_i = 1$, we have $z_i = 2 \notin \{0,1\}$). Now, Alice and Bob respectively set up the unitaries U_A and U_B shown below, where the first register holds $\log n$ qubits and the second holds $O(1)$ qubits (enough to specify a single element of $\mathbb{F}$).

$$\begin{aligned} \forall i \in [n], \alpha \in \mathbb{F}, \quad & U_A \left|i\right\rangle \left|\alpha\right\rangle = \left|i\right\rangle \left|\alpha + C(x)_i\right\rangle \\ & U_B \left|i\right\rangle \left|\alpha\right\rangle = \left|i\right\rangle \left|\alpha + C(y)_i\right\rangle \ . \end{aligned}$$

Alice then simulates the quantum tester and only communicates with Bob in order to make a query to the oracle; if the tester accepts, Alice outputs 0, and she outputs 1 otherwise.

More precisely, whenever the tester calls the oracle U, which acts as $U\left|i\right\rangle\left|\alpha\right\rangle = \left|i\right\rangle\left|\alpha + C(z)_i\right\rangle$ on a $(\log n + O(1))$-bit quantum state ρ, Alice first applies U_A to ρ, then sends all qubits to Bob; Bob then applies U_B on the qubits it receives and returns them to Alice. This communicates a total of $O(\log n)$ qubits and implements the same transformation as querying U, since $U_B \cdot U_A \left|i\right\rangle\left|j\right\rangle = \left|i\right\rangle\left|j + C(x)_i + C(y)_i\right\rangle = \left|i\right\rangle\left|j + C(x+y)_i\right\rangle$ by the linearity of the code C and the fact that $z = x + y$.

Each query made by the tester entails $O(\log n)$ qubits of communication, so that after q queries, Alice and Bob exchange $O(q \cdot \log n)$ qubits in total. The tester accepts with probability at least $2/3$ when $C(z) \in \Pi_B \iff \operatorname{disj}(x, y) = 0$, in which case Alice outputs 0. If $\operatorname{disj}(x, y) = 1$, we have that $C(z)$ is δ-far from Π_B (since the relative distance of C is δ). Since $\varepsilon \leq \delta$, the ε-tester rejects with probability $2/3$ and Alice outputs 1 in this case.

Thus, Alice is able to compute disj_k with $O(q \cdot \log n)$ qubits of communication. Since the quantum communication complexity of disj_k is $\Omega(\sqrt{k})$ [Raz03], we conclude that the tester must make $\Omega(\sqrt{k}/\log n) = \Omega(n^{0.499}/\log n) = \Omega(n^{0.49})$ queries. $\square$

Remark 5.4. Although we reduce disj to testing *non*-booleanity, a symmetric argument shows the same lower bound for the (arguably more natural) property of *booleanity* $\left\{C(z) : z \in \{0,1\}^k\right\}$. Often, one key step in PCP constructions is to check that an encoding corresponds to a logical assignment, i.e., that it is the encoding of a boolean message. Therefore, bounds on booleanity may have consequences for PCPs.

We conclude this section with the separation implied by Lemma 5.1 and Lemma 5.2.

Theorem 5.5. *Π_B belongs to $\mathsf{MAP}(\varepsilon, \log n, O(1))$ but not to $\mathsf{QPT}(\varepsilon, o(n^{0.49}))$, for every constant $\varepsilon \in (0, \delta]$. Therefore,*

$$\mathsf{MAP}(\varepsilon, \log n, O(1)) \not\subseteq \mathsf{QPT}(\varepsilon, o(n^{0.49})) .$$

5.2.3 Quantum testers versus MAPs

In this section, we will show the existence of a property that is easily testable with a quantum tester, but for which a classical tester – even with additional access to a proof – must make a number of queries that depends strongly on the length of the input. More formally, we will show in Theorem 5.6 the existence of a property of n-bit strings in $\mathsf{QPT}(\varepsilon, O(1/\varepsilon))$ that is not in $\mathsf{MAP}(\varepsilon, p, q)$ when $p \cdot q = o(n^{1/4})$ and ε is a small enough constant.

The property in question is derived from Forrelation, a problem that strongly separates classical and quantum algorithms in the query model; in fact, the work that proved such a separation already shows that it carries over to the property testing setting [AA18], which we will extend to the setting of MAPs. Formally, we have

Lemma 5.3 ([AA18]). *Define the property $\Pi_F := \{(f, g) : \Phi_{f,g} \leq 1/100\}$, where*

(f,g) are $n/2$-bit strings corresponding to pairs of $\log(n/2)$-bit boolean functions and

$$\Phi_{f,g} = (n/2)^{-3/2} \sum_{x,y\in\{0,1\}^{\log(n/2)}} f(x)(-1)^{x\cdot y} g(y).$$

Then, for any $\varepsilon > 0$ sufficiently small,

$$\Pi_F \in \mathsf{QPT}(\varepsilon, O(1/\varepsilon)) \text{ and } \Pi_F \notin \mathsf{PT}(\varepsilon, o(\sqrt{n}/\log n)).$$

Therefore, the property Π_F is easy for quantum testers and hard for their classical counterparts. This section is thus devoted to showing that testing Π_F is hard not only for property testers, but for MAPs as well: we will prove that a MAP for Π_F requires proof length p and query complexity q satisfying $pq = \Omega(n^{1/4})$. We first introduce relevant definitions and theorems, then describe the steps of the proof.

Recall that MA is the class of languages that are decidable *in polynomial time* with a proof string (of polynomial size), the analogue of which is MAP in the property-testing setting. By [HHT97], MA is contained in the class $\mathsf{BPP}_{\text{path}}$ of languages decidable (with high probability) by a randomised Turing machine whose computational paths are all equally likely.[4] Lower bounds on the query complexity of Forrelation (i.e., deciding whether $|\Phi_{f,g}| \leq 1/100$ or $\Phi_{f,g} > 3/5$ for a pair (f,g) of boolean functions) against the latter are known:

Proposition 5.1 ([Aar10, Che16])**.** *Any BPP$_{path}$ algorithm for Forrelation must make $\Omega(n^{1/4})$ queries to its input.*

We are now ready to describe the three steps taken in proving hardness of Π_F for MAP: we (1) show that transforming an MA algorithm with proof complexity p and query complexity q into a BPP$_{\text{path}}$ one [HHT97] yields an algorithm with query complexity $O(pq)$; (2) show how a MAP for Π_F implies an MA algorithm with the same parameters *for Forrelation*; and (3) conclude that $pq = o(n^{1/4})$ implies a $\mathsf{BPP}_{\text{path}}$ upper bound of $O(pq) = o(n^{1/4})$ for the query complexity of Forrelation, which contradicts Proposition 5.1.

The original proof of $\mathsf{MA} \subseteq \mathsf{BPP}_{\text{path}}$ ([HHT97], Theorem 3.7) takes an MA algorithm with proof complexity p and constant success probability, repeats the execution $O(p)$ times, amplifying the success probability to $1 - O(2^{-p})$, and then defines a BPP$_{\text{path}}$ machine as follows. The machine (non-deterministically) guesses a proof string and simulates the MA algorithm with it, spawning "dummy" execution

[4]In BPP, the probability of following a computational path is a function of its length (which coincides with the number of random coins flipped by the algorithm). $\mathsf{BPP}_{\text{path}}$ differs from BPP by lifting this restriction.

paths if the MA algorithm accepts. Inspecting this transformation in the query model, we obtain a quadratic overhead: if the MA algorithm has query complexity q and proof complexity p, the $\mathrm{BPP}_{\mathrm{path}}$ machine thus obtained has query complexity $O(pq)$ (the $O(p)$ repetitions of the MA algorithm increase its query complexity multiplicatively by this amount, while the dummy paths make no queries).

The second step is formalised by the following lemma.

Lemma 5.4. *A MAP protocol for Π_F with sufficiently small proximity parameter $\varepsilon > 0$ implies an* MA *algorithm for Forrelation with the same query and proof complexities as the MAP protocol.*

Proof. We define an MA protocol *for Forrelation* (as a gap problem) the natural way: the proofs and queries correspond to the proofs and queries of the MAP, and the MA verifier accepts if and only if the MAP rejects.

To show correctness of this protocol, we follow the reduction of [AA18]. Specifically, [AA18, Lemma 40] shows that any (f', g') such that f' is ε-close to f and g' is ε-close to g satisfies $|\langle f', Hg'\rangle - \langle f, Hg\rangle| = O(\sqrt{\varepsilon}\log(1/\varepsilon))$. By choosing a suitably small ε, the right-hand side is at most (say) $1/100$. Thus any (f, g) such that $\langle f, Hg\rangle > 3/5$ is ε-far from Π, and it follows that the MAP protocol will accept (with high probability) pairs (f, g) such that $|\langle f, Hg\rangle| \leq 1/100$ and will reject if $\langle f, Hg\rangle > 3/5$. Therefore, the MA protocol is able to distinguish between the two cases. □

A simple argument now proves the separation.

Theorem 5.6. *Let $\varepsilon > 0$ be a small enough constant. There exists a property Π_F such that $\Pi_F \in \mathsf{QPT}(\varepsilon, O(1/\varepsilon))$ and $\Pi_F \notin \mathsf{MAP}(\varepsilon, p, q)$ for any p, q such that $p \cdot q = o(n^{1/4})$.*

Proof. Suppose, towards contradiction, that there existed a MAP for Π_F with any proximity parameter ε, as well as proof complexity p and query complexity q satisfying $p \cdot q = o(n^{1/4})$.

By Lemma 5.4, there exists an MA protocol for Forrelation with the same query and proof complexities, which can then be transformed into a BPP_{path} algorithm with query complexity $O(p \cdot q) = o(n^{1/4})$ for the same problem. But this contradicts the $\Omega(n^{1/4})$ lower bound of Proposition 5.1. □

We are finally ready to prove the main separation, by exhibiting a property Π in QCMAP which is in neither MAP nor QPT.

Proof of Theorem 5.2. Recall that our goal is to show that the property $\Pi = \Pi_B \times \Pi_F$ that is efficiently ε-testable by a QCMAP, but not by a quantum tester (without a proof) nor classically with a proof, for some small enough $\varepsilon = \Omega(1)$. To this end, we invoke Theorem 5.5 and Theorem 5.6 and give the following verifier strategy explicitly. Note that, while the strings in Π_B are over an alphabet $\mathbb{F}$ larger than $\{0,1\}$, since $|\mathbb{F}| = O(1)$ each symbol can be represented by $O(1)$ bits (and the proof indicates the first bit in such a block).

```
Algorithm 5.2: QCMAP verifier for $\Pi_B \times \Pi_F$

Input: explicit access to a proximity parameter $\varepsilon' = 2\varepsilon > 0$ and a proof
$i \in [n/2]$, as well as oracle access to a string $z \in \{0,1\}^n$.
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Step 1: Interpret the input as a concatenation of $n/2$-bit strings $x$ and $y$. Use
        Algorithm 5.1 to verify, with $O(1)$ queries and the proof $i$, whether
        $x \in \Pi_B$ with proximity $\varepsilon := \varepsilon'/2$.

Step 2: Use the quantum tester of Lemma 5.3 to test, with $O(1)$ queries, if
        $y \in \Pi_F$ with proximity $\varepsilon$.

Step 3: If both of the previous tests accepted, then accept; otherwise, reject.
```

Completeness follows immediately from Lemma 5.1 and Lemma 5.3, since the verifier for Π_B accepts with certainty and the and tester for Π_F accepts with probability $2/3$ when $x \in \Pi_B$ and $y \in \Pi_F$. If, on the other hand, (x, y) is ε-far from $\Pi_B \times \Pi_F$, then either x is ε-far from Π_B or y is ε-far from Π_F, and either the verifier for Π_B or the tester for Π_F will reject (with probability $2/3$). Thus, the QCMAP verifier for Π (executed with respect to proximity parameter $\varepsilon' = 2\varepsilon$) implies $\Pi \in \mathsf{QCMAP}(\varepsilon, \log n, O(1))$.

All that remains is to show $\Pi = \Pi_B \times \Pi_F$ does not admit an efficient quantum tester nor a MAP. Assume, towards contradiction, that either $\Pi \in \mathsf{QPT}(\varepsilon, o(n^{0.49}))$ or $\Pi \in \mathsf{MAP}(\varepsilon, p, q)$ when $p \cdot q = o(n^{1/4})$. In the first case, applying the tester for Π to $\Pi_B \times \{y\}$ for some fixed $y \in \Pi_F$ shows that $\Pi_B \in \mathsf{QPT}(\varepsilon, o(n^{0.49}))$, a contradiction with Lemma 5.2. In the second case, applying the MAP protocol for Π to $\{x\} \times \Pi_F$, for some fixed $x \in \Pi_B$, shows that $\Pi_F \in \mathsf{MAP}(\varepsilon, p, q)$, a contradiction with Theorem 5.6. $\square$

5.3 A hard class of problems for QMAPs

When introducing a new complexity class in the landscape of known classes, it is important not only to exhibit problems it can solve, but also problems it cannot.

We set out to show a natural limitation on QMAPs in this section, by answering (negatively) the following question: if a property "looks random" on any subset of q coordinates, can a quantum proof be of any help to a verifier with query complexity q? Intuitively, the answer should be no: if querying q coordinates provides no information as to whether or not an input satisfies a property, then any proof (quantum or otherwise) should not be able to offer more information in conjunction with the queries than it does on its own.

We formalise this intuition in Theorem 5.8, which states the following: if a property $\Pi \subset \{0,1\}^n$ is k-wise independent and sparse (i.e., its size $|\Pi|$ is sufficiently small compared to the set of all 2^n bit strings), then k is a lower bound on the number of queries made by any randomised query algorithm that accepts all inputs in Π with probability strictly greater than $1/2$, and rejects with probability strictly greater than $1/2$ when run on any input that is far from Π. In other words, the UPP query complexity of testing Π is at least k (recall that UPP is the query model version of PP, which captures randomized computation with small bias). Note that *some* assumption on the sparsity of Π is necessary for any non-trivial lower bound to hold, if only to rule out, e.g., the trivially testable property $\Pi = \{0,1\}^n$.

Combining Theorem 5.8 with the well-known inclusion $\mathsf{QMA} \subseteq \mathsf{PP}$ [MW05] allows us to conclude the following: for any k-wise independent and sufficiently sparse property Π, the product of proof and query complexities of a QMAP for verifying membership in Π with constant proximity parameter ε is $\Omega(k)$ (see Corollary 5.1).

The proof of Theorem 5.8 works as follows. Our analysis shows that the sparsity of Π ensures there exists a subset $\Pi' \subset \{0,1\}^n$ that is far from Π such that Π' is *also* k-wise independent (see Lemma 5.5). This means that any query algorithm making fewer than k queries cannot distinguish a random input in Π from a random input in Π', as both sets "look random" when inspecting only k bits of a randomly chosen input from the set. Yet since Π' is far from Π, any testing procedure for Π must distinguish Π from Π'. Hence, any tester for Π must make k queries (even if it only outputs the correct answer on inputs in Π and Π' with probability strictly greater than $1/2$).

We begin by recalling the definition of k-wise independence.

Definition 5.6. *A set of strings $S \subseteq \{0,1\}^n$ is called k-wise independent if, for any fixed set of indices $I \subset [n]$ of size k, the string $x_{|I}$ is uniformly random when x is sampled uniformly from S. Equivalently, for every $y \in \{0,1\}^k$,*

$$\left|\left\{x \in S : x_{|I} = y\right\}\right| = \frac{|S|}{2^k}.$$

We next show that, given any small enough set S of strings, there exists a $\Omega(n)$-wise independent set that is ε-far from S. In the following lemma, H denotes the binary entropy function $H(\alpha) = -\alpha \log \alpha - (1-\alpha)\log(1-\alpha)$ (whose restriction to $[0,1/2]$ is bijective).

Lemma 5.5. *Let $\varepsilon \in (0, H^{-1}(1/5))$ and $S \subseteq \{0,1\}^n$ be such that $|S| < 2^{(1/4-H(\varepsilon))n}$. Then there exists a linear code C that is ε-far from S with dual distance $\Omega(n)$; equivalently, C is $\Omega(n)$-wise independent.*

Proof. Let $C\colon \{0,1\}^{3n/4} \to \{0,1\}^n$ be a random linear code (where each entry of its generator matrix is a Bernoulli(1/2) random variable). Then, for every $x \in \{0,1\}^{3n/4}$, the codeword $C(x)$ is uniformly random in $\{0,1\}^n$ (but *not* independent of other codewords). Denoting by $N_\varepsilon(S)$ the ε-neighbourhood of S (i.e., the set of bit strings at distance at most ε from S), we have:

$$\begin{aligned}\mathbb{P}[C \cap N_\varepsilon(S) \neq \varnothing] &\leq \sum_{x\in\{0,1\}^{3n/4}} \mathbb{P}[C(x) \in N_\varepsilon(S)] = 2^{3n/4} \cdot \frac{|N_\varepsilon(S)|}{2^n} \\ &\leq \frac{|S| \cdot 2^{H(\varepsilon)n}}{2^{n/4}} = o(1),\end{aligned}$$

and, by the probabilistic method, there exists a code $C \subset \{0,1\}^n$ of size $2^{3n/4}$ that is ε-far from S. Moreover, the dual code $C^\perp\colon \{0,1\}^{n/4} \to \{0,1\}^n$ is a linear code whose distance meets the Gilbert-Varshamov bound with high probability; that is, the distance of this dual code is $\Omega(n)$ with probability $1-o(1)$, proving the claim. $\square$

The previous lemma, when applied to a "random-looking" set S (i.e., a k-wise independent S, for $k = o(n)$), will ensure that S and the code C are hard to distinguish. To make this precise, we first recall the definition of the *threshold degree* of a (partial) function.

Definition 5.7. *Let $\mathcal{X} \subseteq \{1,-1\}^n$ and let $f : \mathcal{X} \to \{1,-1\}$ be any function defined on domain $\mathcal{X} \subseteq \{1,-1\}^n$.*[5] *The threshold degree of f, denoted* $\mathrm{thrdeg}(f)$*, is the minimal degree of an n-variate polynomial p that sign-represents f, i.e., such that $f(x) = \mathrm{sgn}(p(x))$ for all $x \in \mathcal{X}$.*[6] *Note that no constraints are placed on the behaviour of $p(x)$ at inputs in $\{1,-1\}^n \setminus \mathcal{X}$.*

The threshold degree is a measure of complexity of boolean functions (in particular), so that we expect functions with high threshold degree to also have high query complexity. This intuition is validated by the following folklore result: the

[5]For notational convenience, we consider boolean functions with codomain $\{1,-1\}$, noting that this is equivalent to the usual codomain $\{0,1\}$ by mapping $0 \to 1$, $1 \to -1$, and $\oplus$ to multiplication.

[6]Here, $\mathrm{sgn}(t)$ is defined to equal 1 if $t > 0$, -1 if $t < 0$, and 0 if $t = 0$.

minimal query complexity of a UPP algorithm that computes f is exactly equal to its threshold degree. We provide a proof of this fact for completeness, as, to the best of our knowledge, it is not explicitly proven in the literature.

We write $f \in \mathsf{UPP}(q)$ when there exists a UPP algorithm with query complexity q that computes f, and denote by $q(f)$ the integer such that $f \in \mathsf{UPP}(q(f))$ but $f \notin \mathsf{UPP}(q(f)-1)$.

Lemma 5.6. *For any $\mathcal{X} \subseteq \{1,-1\}^n$ and $f\colon \mathcal{X} \to \{1,-1\}$, it holds that* $\operatorname{thrdeg}(f) = q(f)$.

Proof. We prove both inequalities, starting with $q(f) \leq \operatorname{thrdeg}(f) \coloneqq d$.

Let $P(T_1,\dots,T_n) = \sum_{S\subset[n],|S|\leq d} \alpha_S \prod_{i\in S} T_i$ be a polynomial of degree d that sign-represents f, i.e., such that $f(x) = \operatorname{sgn}(P(x))$ for all $x \in \mathcal{X}$. Consider the algorithm A that queries the set of coordinates S with probability $|\alpha_S|/\sum_{|S'|\leq d}|\alpha_{S'}|$ and outputs $\operatorname{sgn}(\alpha_S)\cdot\prod_{i\in S} x_i$ (note that its query complexity is d). Fix $x \in \mathcal{X}$ and suppose, without loss of generality, that $f(x) = 1$. We thus have

$$\mathbb{E}[A^x] = \frac{1}{\sum_{|S|\leq d}|\alpha_S|}\sum_{|S|\leq d}|\alpha_S|\cdot\operatorname{sgn}(\alpha_S)\cdot\prod_{i\in S}x_i = \frac{P(x)}{\sum_{|S|\leq d}|\alpha_S|} > 0,$$

and, since A^x only outputs 1 or -1, we have $\mathbb{P}[A^x=-1]+\mathbb{P}[A^x=1]=1$ and thus $\mathbb{P}[A^x=1] > 1/2$. It follows that A is a UPP algorithm for f with query complexity d and thus $q(f) \leq \operatorname{thrdeg}(f)$.

To prove the reverse inequality, consider a UPP algorithm A that computes f with query complexity $q \coloneqq q(f)$, given by a distribution over decision trees of depth at most q. To see that the function computed by each decision tree D can be sign-represented by a polynomial of degree at most q (a standard fact), we follow the exposition on *leaf indicators* in [GM21]. Denote by L the set of leaves of D, and identify each $\ell \in L$ with its indicator function $\ell\colon \{1,-1\}^n \to \{0,1\}$ such that $\ell(x) = 1$ if and only if ℓ is the unique leaf reached on input x in D.

Then, if $c_\ell \in \{1,-1\}$ is the output of the decision tree when an execution ends at the leaf ℓ, the output of D on input x is $\sum_{\ell\in L} c_\ell\cdot\ell(x)$. Thus, showing $\ell(\cdot)$ can be represented by a polynomial of degree at most q implies the same degree bound for the computation of D. Fix $\ell \in L$, let $(i_1,\dots,i_d) \in [n]^q$ be the coordinates queried by the root-to-leaf path that ends at ℓ, and let the sequence of bits $(b_1^\ell,\dots,b_d^\ell) \in \{1,-1\}^q$ correspond to the queried values that cause this path to be followed. Then,

$$\ell(x_1,\dots,x_n) = \prod_{j=1}^{q}\frac{x_{i_j}+b_j}{2b_j},$$

so $\ell(\cdot)$ coincides with the degree-q polynomial $P(T_1, \ldots, T_n) = 2^{-q} \prod_{j=1}^{q} (T_{i_j} + b_j)/b_j$. Thus, the output of A^x *when it selects this tree* is $\sum_{\ell \in L} c_\ell \cdot \ell(x)$, and $\mathbb{E}[A^x]$ is a convex combination of such sums (which also has degree q). Since $f(x) = \operatorname{sgn}(\mathbb{E}[A^x])$ for all $x \in \mathcal{X}$, we conclude that $\operatorname{thrdeg}(f) \leq q(f)$ and the claim follows. □

The final ingredient to show the lower bound is the next theorem, a special case of the "Theorem of the Alternative" [OS10, ABFR94].

Theorem 5.7. *Let $\mathcal{X} \subseteq \{1, -1\}^n$ and let $f \colon \mathcal{X} \to \{1, -1\}$ be any partial boolean function defined over domain $\mathcal{X}$. If there exists a distribution $\mathcal{D}$ on $\mathcal{X}$ such that $\mathbb{E}_{x \leftarrow \mathcal{D}}[f(x) \cdot m(x)] = 0$ for every monomial m of degree less than k, then the threshold degree of f is at least k.*

We are now ready to prove the main result of this section.

Theorem 5.8. *Let $\Pi \subseteq \{1, -1\}^n$ be a k-wise independent property, with $k = o(n)$, such that $|\Pi| < 2^{(1/4 - H(\varepsilon))n}$. Then $f \notin \mathsf{UPP}(k-1)$, where f is the partial function such that $f(x) = -1$ when $x \in \Pi$ and $f(x) = 1$ when x is ε-far from Π (and f is undefined otherwise).*

Proof. First, apply Lemma 5.5 to obtain a k-wise independent code $C \subseteq \{0,1\}^n$ that is ε-far from Π. Let $\mathcal{D}$ be the distribution obtained by drawing a uniformly random element of Π with probability $1/2$ and drawing a uniformly random element of C with probability $1/2$. Then for every monomial m of degree less than k,

$$\begin{aligned}
\mathbb{E}_{x \leftarrow \mathcal{D}}[f(x)m(x)] &= \frac{\mathbb{E}_{x \leftarrow \Pi}[f(x)m(x)] + \mathbb{E}_{x \leftarrow C}[f(x)m(x)]}{2} \\
&= \frac{\mathbb{E}_{x \leftarrow \Pi}[m(x)] - \mathbb{E}_{x \leftarrow C}[m(x)]}{2} = 0.
\end{aligned}$$

The final equality above holds by virtue of the k-wise independence of both Π and C. Let $\mathcal{X}$ be the union of inputs in Π and inputs that are ε-far from Π. Define the partial function f over domain $\mathcal{X}$ via:

$$f(x) = \begin{cases} -1 & \text{, if } x \in \Pi \\ 1 & \text{, if } x \in \mathcal{X} \setminus \Pi. \end{cases}$$

By Theorem 5.7, the distribution $\mathcal{D}$ constructed above witnesses the fact that $\operatorname{thrdeg}(f) \geq k$. Since the UPP query complexity of f is $\operatorname{thrdeg}(f)$ by Lemma 5.6, the claim follows. □

We conclude the section with a corollary that follows from the inclusion $\mathsf{QMA} \subseteq \mathsf{PP}$. The proof of this inclusion (in the polynomial-time setting) proceeds in two steps: (1) reducing the error rate of a QMA algorithm to roughly 2^{-p}, where p is the length of the proof given to the verifier, by repeating the algorithm $O(p)$ times; and (2) running the verifier with the proof fixed to be the maximally mixed state. This exhibits a gap of roughly 2^{-p} between the acceptance probabilities of yes- and no-inputs, which suffices to place the problem in PP [MW05, Wat09]. The same transformation, carried out *in the query model*, implies that any sufficiently small $\Pi \in \mathsf{QMAP}(\varepsilon, p, q)$ can be "ε-tested" by a UPP algorithm with query complexity $O(pq)$; that is, any function f as in the statement of Theorem 5.8 is such that $f \in \mathsf{UPP}(O(pq))$. Therefore,

Corollary 5.1. *For any sufficiently constant small $\varepsilon > 0$ and k-wise independent property $\Pi \subseteq \{0,1\}^n$ such that $|\Pi| < 2^{n/5}$, we have $\Pi \notin \mathsf{QMAP}(\varepsilon, p, q)$ unless $pq = \Omega(k)$.*

5.4 Quantum speedups for proximity-oblivious MAPs

We now shift gears and move to our *algorithmic* results. We recall in this section the technique of quantum *amplitude amplification*, and prove its consequences for the classes of algorithms we consider in this section. Roughly speaking, given an algorithm that finds, with probability γ, a preimage of 1 of a boolean function, amplitude amplification allows us to repeat it $O(1/\sqrt{\gamma})$ times in order to find such a preimage with high probability (as opposed to $O(1/\gamma)$ repetitions classically). Formally, we have:

Theorem 5.9 ([BHMT02]). *Let $v \colon S \to \{0,1\}$ be a boolean function (from an arbitrary set S) and let A be a quantum algorithm that makes no intermediate measurements (i.e., is a unitary transformation), such that measuring the state $A|\mathbf{0}\rangle$ yields as outcome $s \in v^{-1}(1)$ with probability $\gamma > 0$. Then there exists a quantum algorithm B that uses $O(1/\sqrt{\gamma})$ applications of the unitaries A and A^{-1}, such that measuring $B|\mathbf{0}\rangle$ yields as outcome $s \in v^{-1}(1)$ with probability $2/3$.*

We note that the theorem applies to classical randomised algorithms as a special case. An immediate corollary for *promise problems* in the query model (which is the setting for property testers, MAPs and variations thereof) is the following.[7]

Corollary 5.2 (Amplitude amplification for promise problems in the query model)**.** *Let $Y, N \subseteq \{0,1\}^n$ with $Y \cap N = \varnothing$ define a promise problem on n-bit strings whose yes- and no-inputs are Y and N, respectively. Let A be a randomised algorithm with*

[7] While we could state amplitude amplification for testers directly, a subtle issue would arise: MAPs are equivalent to a collection of *partial* testers, which are not "vanilla" testers but are still promise problems.

oracle access to a string $x \in \{0,1\}^n$ that makes q queries, always accepts when $x \in Y$ and rejects with probability at least γ when $x \in N$. Then, there exists a quantum algorithm B that makes $O(q/\sqrt{\gamma})$ queries to the unitary $U_x |i\rangle |b\rangle = |i\rangle |b \oplus x_i\rangle$, always accepts when $x \in Y$ and rejects with probability $2/3$ when $x \in N$.

This follows from the observation that each $x \in Y \cup N$ induces a function $f_x \colon \{0,1\}^r \to \{0,1\}$ where r is the number of random bits used by A. If A^x accepts when the outcome of its random coin flips is s, we define $f_x(s) = 0$, and if A^x rejects when its random string is s, then $f_x(s) = 1$. We then apply Theorem 5.9 to the algorithm A^x, for each fixed $x \in Y \cup N$ (or, more precisely, to the modified algorithm that computes f_x written as a reversible circuit and thus implements a query to x as $(i,b) \mapsto (i, b \oplus x_i)$), obtaining B^{U_x} (recall that U_x is the unitary mapping $|i\rangle |b\rangle \mapsto |i\rangle |b \oplus x_i\rangle$). Measuring $B^{U_x} |\mathbf{0}\rangle$, using the outcome as the random string for an execution of A^x and outputting accordingly yields the claimed algorithm.

Note that Corollary 5.2 directly applies to *one-sided proximity-oblivious* testers, which are testers that always accept n-bit strings in the property and reject strings that are ε-far from it with *detection probability* $\rho(\varepsilon, n)$. We prove the that the same speedup can be obtained for MAPs; more precisely, properties that admit one-sided proximity-oblivious MAPs allow for more efficient verification by a quantum algorithm using the same proof string. Before, however, we formalise an observation made in [FGL14], which shows an equivalence between MAPs and coverings by *partial testers*.

A partial tester T is a relaxation of the standard definition of a tester, that accepts inputs inside a property Π_1 and rejects inputs that are far from a *larger* property Π_2 that contains Π_1 (standard testing is the case where $\Pi_2 = \Pi_1$).

Claim 5.1. *A MAP verifier V for property $\Pi \subseteq \Gamma^n$ with error rate σ and query complexity $q = q(n, \varepsilon)$, that receives a proof of length p, is equivalent to a collection of* partial testers *$\{T_\pi : \pi \in \{0,1\}^p\}$. Each $T_\pi(\varepsilon)$ accepts inputs in the property Π_π and rejects inputs that are ε-far from Π, with the same query complexity q and error rate σ as V. The properties Π_π satisfy $\Pi_\pi \subseteq B_\varepsilon(\Pi)$ and $\Pi \subseteq \cup_\pi \Pi_\pi$.*

Proof. Consider a MAP verifier V with parameters as in the statement, and define $T_\pi(\varepsilon) \coloneqq V(\varepsilon, \pi)$ for each purported proof $\pi \in \{0,1\}^p$. Clearly the query complexity and error rate of T_π match those of V, and these testers reject points that are ε-far from Π. The property Π_π is, by definition, the set of inputs that T_π accepts (with probability at least $1 - \sigma$), which is contained in $B_\varepsilon(\Pi)$ (since elements of $\overline{B_\varepsilon(\Pi)}$ are rejected), and may possibly be empty. But since the definition of a MAP ensures that, for each $x \in \Pi$, the tester T_π^x accepts for some proof π (with probability $1 - \sigma$), we have $\Pi \subseteq \cup_\pi \Pi_\pi$.

Consider, now, a collection of testers $\{T_\pi : \pi \in \{0,1\}^p\}$ as in the statement, and define a MAP verifier V that simply selects the tester indexed by the received proof string; i.e., $V(\varepsilon, \pi) \coloneqq T_\pi(\varepsilon)$. Then, with probability at least $1-\sigma$, the verifier V rejects inputs that are ε-far from Π and accepts $x \in \Pi$ when its proof string is $\pi \in \{0,1\}^p$ such that $x \in \Pi_\pi$. □

We are now ready to prove the formal version of Theorem 8.

Theorem 5.10. *Let Π be a property admitting a one-sided proximity-oblivious MAP protocol, which receives a proof of length $p = p(n)$, makes $q = q(n)$ queries and rejects strings ε-far from Π with probability at least $\rho = \rho(\varepsilon, n)$. Then, for any $\varepsilon \in (0,1)$,*

$$\Pi \in \mathsf{QCMAP}\left(\varepsilon, p, \frac{q}{\sqrt{\rho}}\right) .$$

Proof. By Claim 5.1, a MAP verifier V can be equivalently described as a collection of probabilistic algorithms $\{T_\pi : \pi \in \{0,1\}^p\}$ indexed by all proof strings π. By definition, for every $x \in \Pi$ there exists $\pi \in \{0,1\}^p$ such that T_π^x always accepts; and, for every x that is ε-far from Π, every proof string π is such that T_π^x rejects with probability at least ρ. Therefore, T_π solves the promise problem whose yes-inputs comprise the subset of Π for which π is a valid proof, and whose no-inputs are the strings ε-far from Π.

Let W_π be the algorithm obtained from T_π by Corollary 5.2. Then W_π^x accepts (with probability 1) when $x \in \Pi$ and π is a valid proof for x, and W_π^x rejects (with probability 2/3) when x is ε-far from Π and π is any proof string; in other words, the algorithm W that executes W_π when it receives π as a proof string is a QCMAP verifier for Π. Moreover, since the proof string is reused and W makes $O(q/\sqrt{\rho})$ queries, the proof and query complexities are as stated. □

We conclude with two applications of Theorem 5.10: to *read-once branching programs* (ROBPs) and *context-free languages* (CFLs), which are shown to admit proximity-oblivious MAPs in [GGR18] (see Remark 5.13 for details on these results).

Theorem 5.11 ([GGR18], Lemma 3.1)**.** *For every read-once branching program on n variables of size $s = s(n)$, let $A_B \coloneqq \{x \in \{0,1\}^n : B(x) = 1\}$ be the set of strings accepted by B. Then, for every $k \le n$, the property Π_B admits a one-sided proximity-oblivious MAP with communication complexity $O(k \log s)$, query complexity n/k and detection probability $\rho(\varepsilon, n) = \varepsilon$.*

Theorem 5.12 ([GGR18], Lemma 4.5)**.** *For every $k \le n$, every context-free language L admits a one-sided proximity-oblivious MAP with communication complexity $O(k \log n)$, query complexity n/k and detection probability $\rho(\varepsilon, n) = \varepsilon$.*

Therefore, applying Theorem 5.10 to Theorems 5.11 and 5.12, we obtain:

Corollary 5.3. *For every read-once branching program B on n variables of size $s = s(n)$, denote by $A_B := \{x \in \{0,1\}^n : B(x) = 1\}$ the set of strings accepted by B. Then*

$$A_B \in \mathsf{QCMAP}\left(\varepsilon, O(k \log s), O\left(\frac{n}{k\sqrt{\varepsilon}}\right)\right) \quad \text{for every } k \leq n \text{ and } \varepsilon \in (0,1).$$

Corollary 5.4. *For every context-free language L,*

$$L \in \mathsf{QCMAP}\left(\varepsilon, O(k \log n), O\left(\frac{n}{k\sqrt{\varepsilon}}\right)\right) \quad \text{for every } k \leq n \text{ and } \varepsilon \in (0,1).$$

Interestingly, these corollaries make explicit a phenomenon in quantum proofs of proximity that does not hold for their classical counterparts: it is possible to test with proximity $\varepsilon = 1/n$, i.e., solve the *exact decision* problem of acceptance by an ROBP and membership in a context-free language, with sublinear proof and query complexity. In particular, taking $k = n^{3/4}$, both complexities are $O(n^{3/4})$. Nonetheless, for the case of branching programs, we will show in Section 5.5.3 how to lift the read-once restriction and improve on the parameters by directly exploiting *decomposability*.

Remark 5.13. We note that a context-free language L is defined in terms of an alphabet of *terminals* (which is generally larger than $\{0,1\}$) as well as an alphabet of variables. However, if both alphabets have constant size, we may represent symbols as bit strings with a constant overhead per query; thus Corollary 5.4 holds for languages over large (constant-size) alphabets.

Moreover, the results of [GGR18] corresponding to Corollaries 5.3 and 5.4 are in fact stronger: both apply more generally to IPPs, and thus to MAPs as a special case (see [GGR18, Section 3.2] for details). In addition, the detection probability of the MAP for ROBPs is $\varepsilon n/n'$ if the branching program has an accepting path of length $n' \leq n$; and the MAP for context-free languages works for *partial derivation languages*, a generalisation of CFLs whose strings may include variable symbols as well as terminals.

5.5 Decomposable properties

In this section, we show how quantum speedups can be applied to proof of proximity protocols for properties that can be broken up into sub-problems in a distance-preserving manner. Roughly speaking, a property Π of n-bit strings is (k, s)-

decomposable if, using an s-bit string y (which we call a *specification*), Π can be mapped to k properties $\{\Lambda^{(i)}\}$ and the input string x can be mapped to a set of k strings $\{x^{(i)}\}$ satisfying the following conditions: (1) when $x \in \Pi$, there exists a specification such that $x^{(i)} \in \Lambda^{(i)}$ for all $i \in [k]$; and (2) when x is ε-far from Π, then, for some specification, $x^{(i)}$ is roughly ε-far from $\Lambda^{(i)}$ for an average $i \in [k]$.

Definition 5.8 (Decomposable property). *Let $\Pi = \bigcup \Pi_n$ be a property of bit strings. For $k = k(n)$, $s = s(n)$, $m_1 = m_1(n), \ldots, m_k = m_k(n)$, we say Π is* (k, s)-decomposable *if there exists a mapping from $S \subseteq \{0,1\}^s$ to (possibly distinct) subproperties $\Lambda^{(1)} \subset \{0,1\}^{m_1}, \ldots, \Lambda^{(k)} \subset \{0,1\}^{m_k}$ such that every $x \in \{0,1\}^n$ uniquely determines $x^{(i)} \in \{0,1\}^{m_i}$ satisfying:*[8]

1. *If $x \in \Pi$, then there exists $y \in S$ such that $x^{(i)} \in \Lambda^{(i)}$ for all $i \in [k]$; and*
2. *If x is ε-far from Π, then, for all $y \in S$ and $i \in [k]$, the string $x^{(i)}$ is ε_i-far from $\Lambda^{(i)}$ and $\mathbb{E}_{i \leftarrow \mathcal{D}}[\varepsilon_i] = \Omega(\varepsilon)$, where $\mathcal{D}$ is the distribution over $[k]$ with probability mass $m_i / (\sum_{j \in [k]} m_j)$ on i.*

If $s = O(k \log n)$ we say Π is succinctlty *k-decomposable. If the strings $x^{(i)}$ form a partition of x, we say Π is* (k, s)-partitionable.

Note that k-decompositions specified by $O(k)$ coordinates of the input string are succinct. All of our applications are to succinctly decomposable properties, and often the $\{x^{(i)}\}$ form an equipartition of x (so each bit of $x^{(i)}$ depends on a single bit of x) and $\mathcal{D}$ is thus the uniform distribution. However, note that if a decomposition is significantly asymmetric, then $\mathcal{D}$ preserves (average) distance while uniform sampling may deteriorate it to $o(\varepsilon)$ (e.g., if $m_i = o(m_1)$ when $i > 1$ and $x^{(1)}$ concentrates all of the corruption).

While the second condition of Definition 5.8 requires the expectation lower bound to hold for arbitrary ε, the definition is still meaningful when it holds only for restricted values of ε. Indeed, we will make use of it for *exact decision* problems in Section 5.5.3, where the only proximity parameter we consider is $\varepsilon = 1/n$; we call such properties decomposable (or partitionable) with respect to exact decision.

As we will see in the next sections, decomposable properties enable the construction of efficient proof of proximity protocols and generalise the notion of "parameterised concatenation properties" introduced by [GR18].

[8]We remark that the mappings $\Pi \mapsto (\Lambda^{(1)}, \Lambda^{(2)}, \ldots, \Lambda^{(k)})$ and $x \mapsto (x^{(1)}, x^{(2)}, \ldots, x^{(k)})$ are functions of the specification $y \in S$ of the decomposition. Although the notation $x^{(i),y}$ and $\Lambda^{(i),y}$ is formally more accurate, the dependency on y will be clear from context and we omit it for ease of notation.

5.5.1 Boosting decompositions via amplitude amplification

As the next theorem shows, decomposable properties allow for quantum speedups regardless of whether they admit proximity-oblivious MAPs.

Theorem 5.14. *Let Π be a property that is (k,s)-decomposable into properties of m_i-bit strings, and set $m = \max_{i \in [k]} \{m_i\}$. Suppose each bit of $x^{(i)}$ can be determined by reading b bits of the input string, and each $\Lambda^{(i)}$ admits a one-sided MAP with proximity parameter ε, query complexity $q = q(m,\varepsilon) = m^\alpha/\varepsilon^\beta$ and proof complexity $p = p(m,\varepsilon)$. Then*

$$\Pi \in \mathsf{QCMAP}(\varepsilon, s + kp, q') ,$$

with

$$q' = \begin{cases} \tilde{O}\left(b \cdot m^\alpha \cdot \varepsilon^{-\max\left(\frac{1}{2},\beta\right)}\right) & \text{if } \alpha > 0 \text{ and } \beta \geq 0 \\ \tilde{O}\left(b \cdot \min\left\{m^{1-\frac{1}{2\beta}}/\sqrt{\varepsilon},\ m^{1-\frac{1}{\beta}}/\varepsilon\right\}\right) & \text{if } \alpha = 0 \text{ and } \beta \geq 1, \end{cases}$$

where $\tilde{O}$ hides polylogarithmic factors in $1/\varepsilon$ (but not m). Moreover, for exact decision *(i.e., testing with proximity $\varepsilon = 1/n$), a proof of length s and $O(bm\sqrt{k})$ queries suffice.*

Before proceeding to the proof, we note that if the MAP protocols for the subproperties are proximity-oblivious, the query complexity can be improved (see Remark 5.15). Let us also summarise the proof strategy of [GR18], which we build upon and generalise.

Consider the special case where a property Π is *k-partitionable* and the strings $x^{(i)}$ are simply the substrings of x of length $m = n/k$ which, concatenated, form x. Suppose, moreover, that the subproperties $\Lambda^{(i)}$ admit *testers* with query complexity $q = m^\alpha/\varepsilon^\beta$ and that Π is the union of $\Lambda^{(1)} \times \Lambda^{(2)} \times \cdots \Lambda^{(k)}$ (over all strings in S). Note that while, in general, a specification must show how to obtain $\Lambda^{(i)}$ from Π *and* how to obtain $x^{(i)}$ from x, for an equipartition the latter is implicit.

A natural candidate for a (classical) MAP protocol for Π is to guess an index $i \in [k]$ and run the tester for $\Lambda^{(i)}$ on $x^{(i)}$. If $x \in \Pi$, then $x^{(i)} \in \Lambda^{(i)}$ for $i \in [k]$ and the tester always accepts; while if x is ε-far from Π, then $x^{(i)}$ is ε_i-far from $\Lambda^{(i)}$ for some ε_i satisfying $\frac{1}{k}\sum_i \varepsilon_i \geq \varepsilon$, *regardless of the specification* (recall that x is ε-far from $\bigcup_{y \in S} \Lambda^{(1)} \times \Lambda^{(2)} \times \cdots \Lambda^{(k)}$).

We now proceed to the proof of the general case, where the decomposition need not be a partition, and it suffices for the subproperties to admit a MAP (rather than a tester). Moreover, we show that quantum algorithms enable a speedup via amplitude amplification (but this requires the MAPs to be *one-sided*, unlike in the classical case).

Proof. Recall that we have a property Π that is (k, s)-decomposable by a collection of strings $S \subseteq \{0,1\}^s$, where each $y \in S$ determines k properties $\Lambda^{(i)} \subseteq \{0,1\}^{m_i}$ and a decomposition of x into k strings $x^{(i)} \in \{0,1\}^{m_i}$. Moreover, each $\Lambda^{(i)}$ admits a MAP with proof complexity p and query complexity $m^\alpha/\varepsilon^\beta$. The verifier for Π executes Algorithm 5.3.

We note that a more naive strategy would succeed, albeit with a worse dependence on ε: choosing $i \in [k]$ with probability proportional to m_i yields a string $x^{(i)}$ which is $\varepsilon/2$-far from $\Lambda^{(i)}$ with probability at least $\varepsilon/2$, so that one could execute the MAP verifier for $\Lambda^{(i)}$ with proximity parameter $\varepsilon/2$ (and use amplitude amplification to achieve constant soundness by repeating this $O(1/\sqrt{\varepsilon})$ times). However, the technique of *precision sampling* [Lev87] overcomes the issue of not knowing the distances ε_i between $x^{(i)}$ and $\Lambda^{(i)}$ more economically: trying every proximity parameter 2^j with $j \in [O(\log 1/\varepsilon)]$, in the spirit of binary search, incurs a merely logarithmic overhead.

Algorithm 5.3: QCMAP verifier for a (k, s)-decomposable property Π

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

Step 1: Interpret the proof as a concatenation of a string $y \in \{0,1\}^s$ with k strings $\pi_1, \dots, \pi_k \in \{0,1\}^p$. If $y \notin S$, i.e., y does not specify a decomposition, then reject.

Step 2: For every $j \in [\lceil \log 1/\varepsilon \rceil + 1]$, let M_j be the algorithm obtained from Corollary 5.2 by performing $O\left(\sqrt{\frac{\log 1/\varepsilon}{2^j \varepsilon}}\right)$ rounds of amplitude amplification to the following subroutine:

Sample $i \in [k]$ with probability $\frac{m_i}{\sum_{j \in [k]} m_j}$ and run the MAP verifier for $\Lambda^{(i)}$ on $x^{(i)}$, with proximity parameter 2^{-j}, using π_i as the proof string. Reject if the MAP for $\Lambda^{(i)}$ rejects.

Step 3: Execute M_j for every $j \in [O(\log 1/\varepsilon)]$. If any of them rejects, then reject; otherwise, accept.

Completeness follows immediately: if $x \in \Pi$, then there exists a string $y \in S$ such that x determines $x^{(i)} \in \Lambda^{(i)}$ for all $i \in [k]$. Since the properties $\Lambda^{(i)}$ admit one-sided MAP protocols with proof complexity p, the subroutine in Step 2 always accepts when given the proof string $\pi = (y, \pi_1, \dots, \pi_k)$, where π_i is a valid proof for

$x^{(i)}$. Therefore, the verifier always accepts as well.

Now, suppose x is ε-far from Π and the proof string $\pi = (y, \pi_1, \ldots, \pi_n)$ is such that $y \in S$ (since otherwise the verifier rejects immediately). Then, since Π is decomposable, $x^{(i)}$ is ε_i-far from $\Lambda^{(i)}$ and $\mathbb{E}_{i \leftarrow \mathcal{D}}[\varepsilon_i] = \Omega(\varepsilon)$, where $\Lambda^{(i)}$ are the subproperties defined by y and $\mathcal{D}$ is the distribution over $[k]$ that samples i with probability proportional to m_i.

To show soundness, we will make use of the following (precision sampling) lemma.

Lemma 5.7 ([Gol14, Fact A.1]). *There exists* $j \in \left[\lceil \log \frac{1}{\varepsilon} \rceil + 1\right]$ *such that*

$$\mathbb{P}_{i \leftarrow \mathcal{D}}[\varepsilon_i \geq 2^{-j}] = \Omega\left(\frac{2^j \varepsilon}{\log 1/\varepsilon}\right) .$$

If the procedure in Step 2 samples $i \in [k]$ such that $\varepsilon_i \geq 2^{-j}$, then it rejects with probability 2/3 (since the MAP has soundness 2/3). With j as ensured by Lemma 5.7, the probability it samples such an $i \in [k]$ is $\Omega\left(\frac{2^j \varepsilon}{\log 1/\varepsilon}\right)$, so that the probability it rejects is $\frac{2}{3} \cdot \Omega\left(\frac{2^j \varepsilon}{\log 1/\varepsilon}\right) = \Omega\left(\frac{2^j \varepsilon}{\log 1/\varepsilon}\right)$; therefore, the algorithm M_j obtained from Corollary 5.2 by $O\left(\sqrt{\frac{\log 1/\varepsilon}{2^j \varepsilon}}\right)$ rounds of amplitude amplification rejects, causing the verifier to also reject, with probability 2/3.

We now prove the stated upper bounds on the query complexity. For every j, each execution of the MAP verifier for $\Lambda^{(i)}$ makes $q(m, 2^{-j})$ queries to $x^{(i)}$, which translate into $b \cdot q(m, 2^{-j})$ queries to x (since each query to $x^{(i)}$ can be emulated with b queries to x). The total query complexity is therefore

$$\sum_{j \in [\lceil \log 1/\varepsilon \rceil + 1]} \sqrt{\frac{\log 1/\varepsilon}{2^j \varepsilon}} \cdot b \cdot q\left(m, 2^{-j}\right) = \tilde{O}\left(\frac{b}{\sqrt{\varepsilon}} \sum_{j \in [\lceil \log 1/\varepsilon \rceil + 1]} \frac{q\left(m, 2^{-j}\right)}{2^{j/2}}\right) .$$

If $q(m, \varepsilon) = m^\alpha / \varepsilon^\beta$ with $\alpha > 0$ and $\beta \geq 0$, then

$$\begin{aligned} \tilde{O}\left(\frac{b}{\sqrt{\varepsilon}} \sum_{j \in [\lceil \log 1/\varepsilon \rceil + 1]} \frac{q\left(m, 2^{-j}\right)}{2^{j/2}}\right) &= \tilde{O}\left(\frac{b m^\alpha}{\sqrt{\varepsilon}} \sum_{j \in [\lceil \log(1/\varepsilon) \rceil + 1]} 2^{j\left(\beta - \frac{1}{2}\right)}\right) \\ &= \tilde{O}\left(b m^\alpha \varepsilon^{-\max(1/2, \beta)}\right) . \end{aligned}$$

If $\alpha = 0$ and $\beta > 0$, we use the bound $q(m, 2^{-j}) \leq m$ for all ε (from the trivial tester that queries the entire input) to obtain two upper bounds for the query complexity:

the first is

$$\begin{aligned}\tilde{O}\left(\frac{b}{\sqrt{\varepsilon}}\sum_{j\in[\lceil\log(1/\varepsilon)\rceil+1]}\frac{q\left(m,2^{-j}\right)}{2^{j/2}}\right) &= \tilde{O}\left(\frac{b}{\sqrt{\varepsilon}}\sum_{j\in[\lceil\log(1/\varepsilon)\rceil+1]}\min\left\{\frac{m}{2^{j/2}},2^{j\left(\beta-\frac{1}{2}\right)}\right\}\right)\\ &= \tilde{O}\left(\frac{b}{\sqrt{\varepsilon}}\sum_{j\in[\lceil\log(1/\varepsilon)\rceil+1]}m^{1-\frac{1}{2\beta}}\right)\\ &= \tilde{O}\left(\frac{bm^{1-\frac{1}{2\beta}}}{\sqrt{\varepsilon}}\right).\end{aligned}$$

The second upper bound matches that obtained by the classical MAP, and is tighter when $m^{\frac{1}{2\beta}} \geq 1/\sqrt{\varepsilon}$. Since $2^{j/2} \leq 4/\sqrt{\varepsilon}$ for all j in the sum above, we have $\sqrt{\varepsilon} = O(2^{-j/2})$; therefore,

$$\begin{aligned}\tilde{O}\left(\frac{b}{\sqrt{\varepsilon}}\sum_{j\in[\lceil\log(1/\varepsilon)\rceil+1]}\frac{q\left(m,2^{-j}\right)}{2^{j/2}}\right) &= \tilde{O}\left(\frac{b}{\varepsilon}\sum_{j\in[\lceil\log(1/\varepsilon)\rceil+1]}\frac{\sqrt{\varepsilon}}{2^{j/2}}\cdot q\left(m,2^{-j}\right)\right)\\ &= \tilde{O}\left(\frac{b}{\varepsilon}\sum_{j\in[\lceil\log(1/\varepsilon)\rceil+1]}\min\left\{\frac{m}{2^{j}},2^{j(\beta-1)}\right\}\right)\\ &= \tilde{O}\left(\frac{b}{\varepsilon}\sum_{j\in[\lceil\log(1/\varepsilon)\rceil+1]}m^{1-\frac{1}{\beta}}\right)\\ &= \tilde{O}\left(\frac{bm^{1-\frac{1}{\beta}}}{\varepsilon}\right).\end{aligned}$$

Finally, observe that, for testing with $\varepsilon = 1/n$ (i.e., deciding exactly), one may take the MAPs for $\Lambda^{(i)}$ to be the trivial testers (with query complexity m and no proof). Moreover, it is unnecessary to iterate over j and apply Lemma 5.7; sampling $i \in [k]$ uniformly and running the trivial tester requires bm queries to x and leads to a rejection with probability at least $1/k$, since $\frac{1}{k}\sum_{i\in[k]}\varepsilon_i > 0$ implies $x^{(i)} \notin \Lambda^{(i)}$ for at least one $i \in [k]$. Therefore, applying $O(\sqrt{k})$ rounds of amplitude amplification to this procedure ensures rejection of an x that is ε-far from Π with constant probability and yields query complexity $O(bm\sqrt{k})$. □

Remark 5.15 (Additional speedup for proximity-oblivious MAPs)**.** If the MAP verifiers for the subproperties $\Lambda^{(i)}$ are proximity-oblivious, it is possible to improve on the query complexity of Theorem 5.14 significantly. More precisely, suppose each $\Lambda^{(i)}$ admits a proximity-oblivious MAP with query complexity $O(1)$ and detection probability $\rho(\varepsilon, m) = \varepsilon^\beta / m^\alpha$. Then, if an input is ε-far from Π, for some $j \in [O(\log 1/\varepsilon)]$ (as ensured by Lemma 5.7), the procedure of Step 2 samples $i \in [k]$ such that $x^{(i)}$ is ε_i-far from $\Lambda^{(i)}$ with $\varepsilon_i = \tilde{\Omega}(2^j \varepsilon)$.

The procedure rejects with probability $\tilde{\Omega}(2^j\varepsilon) \cdot \rho(2^{-j}, m) = \tilde{\Omega}(2^{j(1-\beta)}\varepsilon/m^\alpha)$ in this case. By applying $\tilde{O}(m^{\alpha/2}/\sqrt{2^{-j(\beta-1)}\varepsilon})$ rounds of amplitude amplification for each j (instead of $\tilde{O}(1/\sqrt{2^j\varepsilon})$ performed for MAPs that are not proximity-oblivious), and, since $2^{-j} = \Omega(\varepsilon)$, the total query complexity becomes $\tilde{O}(b\sqrt{m^\alpha/\varepsilon^\beta})$.

We finish this section with a corollary of Theorem 5.14 in the *graph orientation model*. A directed graph is called *Eulerian* if the in-degree of each of its vertices is equal to its out-degree. An orientation is a mapping from the edges to $\{0,1\}$, representing whether each edge is oriented from i to j or from j to i, and the distance between two orientations is the fraction of edges whose orientation must be changed to transform one into the other. Let Π_E be the property consisting of all Eulerian orientations of the complete bipartite graph $K_{2,n-2}$, i.e., the graph with vertex set $[n]$ and edge set $\{\{i,j\} : i \leq 2, j \geq 3\}$.

A MAP protocol for Π_E parameterised by an integer k is constructed in [GR18]; its proof and query complexities are $O(k \cdot \log n)$ and $\tilde{O}(n/(\varepsilon k))$, respectively. That protocol is obtained by applying their (classical) version of Theorem 5.14 using the trivial tester for each of the subproperties. Using the same proof (and thus the same decomposition), and recalling that the trivial tester makes $q(m,\varepsilon) = m$ queries for a sub-property of length m, we obtain

Corollary 5.5. *The property Π_E has a one-sided QCMAP, with respect to proximity parameter ε, that uses a proof of length $O(k \cdot \log n)$ and has query complexity $\tilde{O}\left(\frac{n}{k\sqrt{\varepsilon}}\right)$.*

Note that the query complexity of the classical MAP becomes linear with $\varepsilon = \Omega(1/k)$, whereas the QCMAP is able to decide *exactly* (i.e. test with $\varepsilon = 1/n$) with query complexity $O(n^{3/2}/k)$ (which is still sublinear whenever $k = \omega(\sqrt{n})$). In particular, with $k = n^{3/4}$, both query and communication complexities are $\tilde{O}(n^{3/4})$; see the Section 5.5.3 for further discussion and applications of Theorem 5.14 to exact decision problems.

5.5.2 k-monotonicity

In this section, we show that a generalisation of monotonicity of boolean functions over the line $[n]$ is efficiently testable by QCMAP protocols. A function $f : [n] \to \{0,1\}$ is *k-monotone* if any sequence of integers $1 \leq x_1 < x_2 \ldots < x_\ell \leq n$ such that $f(x_1) = 1$ and $f(x_i) \neq f(x_{i+1})$ for all $i < \ell$ has length $\ell \leq k$. This problem was studied in [CGG+19], where one-sided ε-testers for k-monotonicity on the line are shown to require $\Omega(k/\varepsilon)$ queries, while two-sided testers can achieve query complexity $\tilde{O}(1/\varepsilon^7)$ (which, although a far worse dependence on ε than the lower bound, is *independent* of k).

In order to apply Theorem 5.14, we must only show that k-monotone functions are decomposable. Define $\Pi_{k,[n]}$ as the set of k-monotone boolean functions on the line $[n]$.[9] Then,

Theorem 5.16. *For any $k \in [n]$, the property $\Pi_{k,[n]}$ is succinctly k-decomposable.*

Proof. Since a k-monotone function f has at most $k-1$ critical points, where it changes from nondecreasing to nonincreasing or vice-versa, specifying these points yields a decomposition of f into (1-)monotone subfunctions.

More precisely, a string of length $s \leq (k-1)\log n$ determines a decomposition of the input f by specifying $\ell \leq k-1$ integers $1 < n_1 < n_2 < \cdots < n_\ell < n$. Define $n_0 = 1$, $n_{\ell+1} = n$, $m_i := n_i - n_{i-1} + 1$ and the function $f_i : [m_i] \to \{0,1\}$ by $f_i(x) = f(x + n_{i-1} - 1)$ for all $i \in [\ell+1]$. Then, $f \in \Pi_{k,[n]}$ if and only if:

1. for $i \in [\ell+1]$ odd, $f_i \in \Lambda^{(i)} := \{g \in \Pi_{1,m_i} : g(1) = f_i(1)\}$; and
2. for $i \in [\ell+1]$ even, $f_i \in \Lambda^{(i)} := \{1 - g : g \in \Pi_{1,m_i} \text{ and } g(1) = 1 - f_i(1)\}$.

It is clear that, when $f \in \Pi_{k,[n]}$, there exists a set of $\ell \leq k-1$ distinct integers in $[2, n-1]$ that satisfies both conditions. When f is ε-far from $\Pi_{k,[n]}$, the sum of absolute distances $\varepsilon_i m_i$ from each f_i to $\Lambda^{(i)}$ is $\sum_i \varepsilon_i m_i \geq \varepsilon n$. Therefore,

$$\mathbb{E}_{i \leftarrow \mathcal{D}}[\varepsilon_i] \geq \varepsilon \cdot \frac{n}{\sum_{i \in [\ell+1]} m_i} = \varepsilon \cdot \frac{n}{n+\ell} = \Omega(\varepsilon) ,$$

where $\mathcal{D}$ is the distribution over $[\ell+1]$ that has probability mass $m_i / (\sum_{j \in [\ell+1]} m_j)$ at point i.

Finally, while this ensures $(\ell+1)$-decomposability for some $\ell \leq k-1$, one may deterministically transform it into a k-decomposition (and, in fact, a K-decomposition for any $K \geq \ell$) by, for example, iteratively finding the largest interval and dividing it at its midpoint $k - \ell - 1$ times (with the requirement that nonincreasing functions in the large interval are monotone nonincreasing in both subintervals, and likewise for the nondecreasing case). □

Since monotonicity on the line $[m]$ is ε-testable with $q(m, \varepsilon) = O(1/\varepsilon)$ queries [Gol17, Proposition 1.5], applying (the second case of) Theorem 5.14 yields a QCMAP protocol for k-monotonicity with proof complexity $O(k \log n)$ and query complexity $\tilde{O}(1/\varepsilon)$. Formally,

[9] We consider the standard representation of a boolean function as the bit string obtained by concatenating all function evaluations, i.e., $f\colon [n] \to \{0,1\}$ is represented by $x \in \{0,1\}^n$ with $x_i = f(i)$.

Corollary 5.6. *Let $\Pi_{k,[n]}$ denote the set of* k-monotone *functions $f\colon [n] \to \{0,1\}$, i.e., those which change from nondecreasing to nonincreasing and vice-versa at most $k-1$ times. For all $\varepsilon \in (0,1)$,*

$$\Pi_{k,[n]} \in \mathsf{QMAP}\left(\varepsilon, k\log n, \tilde{O}\left(\frac{1}{\varepsilon}\right)\right) .$$

(The theorem also gives an upper bound of $\tilde{O}(\sqrt{n/\varepsilon})$, which is no better: $\tilde{O}(1/\varepsilon)$ is smaller up to $\varepsilon \approx 1/n$, where the two bounds match.)

It is worth noting that the standard monotonicity tester on the line is *not* proximity-oblivious, unlike, e.g., on the boolean hypercube, where both the "edge tester" [GGL$^+$00] and the state-of-the-art [KMS18] are proximity-oblivious; thus, one could not directly apply amplitude amplification, and must exploit decomposability via Theorem 5.14.

Note that the (one-sided) QCMAP for k-monotonicity is more efficient than any one-sided tester, e.g., when $k = \Theta(\log n)$ and $\varepsilon = \Theta(1/\log^2 n)$: then the QCMAP's proof and query complexities are $\tilde{O}(\log^2 n)$, whereas one-sided testers must make $\Omega(\log^3 n)$ queries. Moreover, our QCMAP outperforms the best known *two-sided* tester of [CGG$^+$19], which makes $\tilde{O}(1/\varepsilon^7)$ queries, even with mild dependencies of the proximity parameter ε on n. Indeed, when $\varepsilon = o(1/\log^{1/7} n)$ and $k = O(1)$, the tester's query complexity is superlogarithmic while the proof and query complexities of the QCMAP are logarithmic.

We can also compare our one-sided QCMAP against one-sided MAPs obtained by the transformation from two-sided testers to one-sided MAPs of [GR18]. The aforementioned two-sided tester yields a MAP with proof complexity $\operatorname{polylog} n$ and query complexity $O(\varepsilon^{-7} \operatorname{polylog}(n/\varepsilon))$, so the QCMAP is more efficient except when k and ε are large (e.g., $k = \omega(\operatorname{polylog} n)$ and $\varepsilon = \Omega(1)$).

5.5.3 Exact problems

To conclude the discussion of decomposability and its consequences, we shift focus to a special case: that of testing n-bit strings with proximity parameter $\varepsilon = 1/n$. Since, for any $\Pi \subseteq \{0,1\}^n$ and $x \in \{0,1\}^n \setminus \Pi$, the string x is at least $1/n$-far from Π, this is the task of *exactly* deciding membership in Π.

Observe that for classical MAPs, nontrivial properties require $\Omega(n)$ queries in this case: even if a verifier receives as proof a claim x' that is allegedly equal to its input string, it requires $O(1/\varepsilon) = \Omega(n)$ queries to check the validity of the claim. Remarkably, *quantum* algorithms are able to solve exact decision problems with sublinear queries (as illustrated by Grover's algorithm, which makes $O(\sqrt{n})$ queries).

Thus, as we show next, insights arising from decomposability are applicable to this setting. We begin by showing a nontrivial QCMA protocol for the parity of a bit string in Section 5.5.3.1, and then extend it to *branching programs* in Section 5.5.3.2.

5.5.3.1 Parity

Consider the problem of deciding if an n-bit string has even parity. (Recall the informal discussion in Section 3.2.2.) This is clearly maximally hard, requiring $\Omega(n)$ queries even for *interactive proofs with arbitrary communication*, which we show next for completeness.

Lemma 5.8. *Any IP verifier that accepts strings of even parity and rejects strings of odd parity with probability* $2/3$ *must make at least* $n/3$ *queries to its input.*

Proof. Let V and P be a verifier and an honest prover for an IP for parity, and assume, towards contradiction, that the query complexity of V is less than $n/3$.

Fix an arbitrary input $x \in \{0,1\}^n$ of even parity. Define S_x as the random variable comprising all the coordinates queried by V in an execution $\langle V^x, P(x)\rangle$ of the protocol, and let $I \in [n]$ be a uniform random variable independent from S_x. Then, since $|S_x| < n/3$,

$$\frac{1}{n}\sum_{i=1}^{n} \mathbb{P}[i \in S_x] = \sum_{i=1}^{n} \mathbb{P}[I = i] \cdot \mathbb{P}[I \in S_x \mid I = i] = \mathbb{P}[I \in S_x] < \frac{1}{3} ,$$

so there exists $i \in [n]$ such that

$$\mathbb{P}[V^x \text{ queries } i \text{ in the execution } \langle V^x, P(x)\rangle] = \mathbb{P}[i \in S_x] < \frac{1}{3} .$$

Now, consider the execution of a protocol on input $y \in \{0,1\}^n$ obtained by flipping the i^{th} bit of x (i.e., such that $y_j = x_j$ if $j \neq i$ and $y_i = 1 - x_i$ otherwise). Let $\tilde{P}$ be a (malicious) prover that executes on y exactly as P does on x; that is, set $\tilde{P}(y) = P(x)$. We thus have

$$\begin{aligned}\mathbb{P}[\langle V^y, \tilde{P}(y)\rangle \text{ rejects}] = \mathbb{P}[i \in S_y] \cdot \mathbb{P}[\langle V^y, \tilde{P}(y)\rangle \text{ rejects} \mid i \in S_y] \\ + \mathbb{P}[i \notin S_y] \cdot \mathbb{P}[\langle V^y, \tilde{P}(y)\rangle \text{ rejects} \mid i \notin S_y] ,\end{aligned}$$

and, moreover, the following equalities between events hold:

$$\begin{gathered}[i \notin S_y] = [i \notin S_x] \text{ , and thus} \\ [\langle V^y, \tilde{P}(y)\rangle \text{ rejects} \mid i \notin S_y] = [\langle V^x, P(x)\rangle \text{ rejects} \mid i \notin S_x] .\end{gathered}$$

Therefore,

$$\begin{aligned}\mathbb{P}[\langle V^y, \tilde{P}(y)\rangle \text{ rejects}] &< \frac{1}{3} + \mathbb{P}[i \notin S_y] \cdot \mathbb{P}[\langle V^y, \tilde{P}(y)\rangle \text{ rejects} \mid i \notin S_y] \\ &= \frac{1}{3} + \mathbb{P}[\langle V^x, P(x)\rangle \text{ rejects and } i \notin S_x] \\ &\leq \frac{1}{3} + \mathbb{P}[\langle V^x, P(x)\rangle \text{ rejects}] \leq \frac{2}{3}\,,\end{aligned}$$

contradicting the correctness of the protocol. □

Rather surprisingly, however, there exists a quantum *non-interactive* protocol that exploits amplitude amplification and achieves sublinear query and communication complexities. This is a direct consequence of the following.

Proposition 5.2. *For any $k \leq n$, the property $\Pi := \left\{x \in \{0,1\}^n : \bigoplus_{j\in[n]} x_j = 0\right\}$ is succinctly k-partitionable (with respect to exact decision).*

Proof. The set of strings that specify decompositions is

$$S = \left\{ y \in \{0,1\}^k : \bigoplus_{i\in[k]} y_i = 0 \right\},$$

i.e., the set of k-bit strings of even parity. A string $y \in S$ specifies a decomposition where, for each $i \in [k]$, the i^{th} subproperty is

$$\Lambda^{(i)} = \left\{ x^{(i)} \in \{0,1\}^{n/k} : \bigoplus_{j\in[n/k]} x^{(i)}_j = y_i \right\}$$

and $x \in \{0,1\}^n$ induces $x^{(i)}$ as the substring of x consisting of the i^{th} block of n/k bits, i.e., $x^{(i)} = \left(x_{\frac{(i-1)n}{k}+1}, x_{\frac{(i-1)n}{k}+2}, \ldots, x_{\frac{in}{k}}\right)$.

Note that the condition $x^{(i)} \in \Lambda^{(i)}$ for all $i \in [k]$ uniquely defines $y \in \{0,1\}^k$ by $y_i = \bigoplus_{\frac{(i-1)n}{k} < j \leq \frac{in}{k}} x_j$. Therefore, if x has even parity, there exists $y \in S$ satisfying the condition, while if $x \notin \Pi$ the only string that satisfies it is not in S. Thus x is $1/n$-far from Π and, *for all $y \in S$*, there exists $i \in [k]$ such that $x^{(i)} \notin \Lambda^{(i)}$, so that the distances ε_j from $x^{(j)}$ to $\Lambda^{(j)}$ satisfy $\frac{1}{k}\sum_{j\in[k]} \varepsilon_j \geq \frac{\varepsilon_i}{k} \geq \frac{1}{k\cdot\frac{n}{k}} = 1/n$. □

Applying Theorem 5.14 with $k = n^{2/3}$, we have:

Corollary 5.7. *There exists a QCMA protocol for parity with $O(n^{2/3})$ query and communication complexities.*

5.5.3.2 Branching programs

First, recall the definition of a branching program on n variables: a directed acyclic graph that has a unique source vertex v_0 with in-degree 0 and (possibly) multiple sink vertices with out-degree 0. Each sink vertex is labeled either with 0 (i.e., reject) or 1 (i.e., accept). Each non-sink vertex is labeled by an index $i \in [n]$ and has exactly 2 outgoing edges, which are labeled by 0 and 1. The output of the branching program B on input $x \in \{0,1\}^n$, denoted $B(x)$, is the label of the sink vertex reached by taking a walk, starting at the source vertex v_0, such that at every vertex labeled by $i \in [n]$, the step taken is on the edge labeled by x_i.

A branching program is said to be *read-once* (or ROBP for short) if, along every path from source to sink, every index $i \in [n]$ appears at most once. The size of a branching program B is the number of vertices in its graph.

A branching program is *layered* if its nodes can be partitioned into V_0, V_1, $\dots$, V_ℓ, where V_0 only contains the source node, V_ℓ are the sink nodes and every edge is between V_{i-1} and V_i for some $i \in [\ell]$. The *length* of a layered branching program is its number of (nontrivial) layers ℓ, and its *width* w is the maximum size of its layers, i.e., $\max_{i\in[\ell]} \{|V_i|\}$.

Recall that Corollary 5.3 shows membership in the set of strings accepted by *read-once* branching programs can be decided with $O(n^{3/4})$ query and proof complexities; thus, since parity is computed by an ROBP (of width 2), we obtain a QCMA protocol with the same parameters. Observe, however, that this is significantly worse than the $O(n^{2/3})$ upper bound of the previous section, which directly exploits decomposability. This suggests a similar improvement may be possible for branching programs, which we now show to be indeed the case for *layered* branching programs.

Proposition 5.3. *Let B be a layered branching program on n-bit strings of length $\ell = \ell(n)$ and width $w = w(n)$. For any $k \leq \ell$, the set $A_B \subseteq \{0,1\}^n$ of strings accepted by B is $(k, O(k \log w))$-partitionable (with respect to exact decision).*

Proof. Let V be the vertex set of the graph that defines the branching program B, whose layers are $V_0 = \{v_0\}$, V_1, $\dots$, V_ℓ and whose set of accepting nodes is $F \subseteq V_\ell$. Decompositions are specified by the set $S = \big\{(v_1,\dots,v_k) : v_i \in V_{i\ell/k} \text{ and } v_k \in F\big\}$ (whose elements are the $k \log w$-bit representations of the sequence of vertices). (Note that $|V_i| \leq w$ implies $\log w$ bits suffice to identify each vertex in a given layer.) A specification fixes k nodes that partition an alleged accepting path into paths of length ℓ/k. Thus, $\Lambda^{(i)}$ is the set of strings accepted by the branching program that is the subgraph of B with layers $\{v_{i-1}\}$ and V_j for $\frac{(i-1)\ell}{k} < j \leq \frac{i\ell}{k}$, source node v_{i-1} and accepting node v_i. The string $x^{(i)}$ is the i^{th} block of ℓ/k bits of the input x.

If $x \in A_B$, the path $(v_0, u_1, \dots, u_\ell)$ determined by the execution of B on x is accepting. Therefore, $y = (u_{\ell/k}, u_{2\ell/k}, \dots, u_{(k-1)\ell/k}, u_\ell) \in S$ is a specification satisfying $x^{(i)} \in \Lambda^{(i)}$ for all $i \in [k]$. If $x \notin A_B$, on the other hand, every specification is such that at least one sub-path does not match the corresponding sub-path determined by x (since otherwise x would be accepted by B), implying $x^{(i)} \notin \Lambda^{(i)}$ for some i. □

Applying Theorem 5.14 with $k = \ell^{2/3}$, we conclude that width-w, length-ℓ branching programs admit QCMAP protocols with proof complexity $O(\ell^{2/3} \log w)$ and query complexity $O(\ell/\sqrt{k}) = O(\ell^{2/3})$. Formally,

Theorem 5.17. *There exists a QCMA protocol for acceptance of n-bit strings by layered branching programs of width $w = w(n)$ and length $\ell = \ell(n)$ with query complexity $O(\ell^{2/3})$ and proof complexity $O(\ell^{2/3} \log w)$.*

Note that the $\Omega(n)$ classical complexity lower bound for parity implies the same bound for width-2 branching programs of length n, and that the complexities of the QCMA protocol are sublinear up to length $o(n^{3/2}/\log w)$; in particular, this holds for width-2^{n^α} branching programs of length $o(n^{\frac{3}{2}(1-\alpha)})$ for every $\alpha \geq 0$. Thus, besides lifting the read-once restriction, it improves on Corollary 5.3 for a wide range of parameters.

5.6 Bipartiteness in bounded-degree graphs

In the bounded-degree graph model, an algorithm is given query access to the *adjacency list* of a graph G whose vertices have their degree bounded by $d = O(1)$. More precisely, given a vertex v and an index $i \in [d]$, the oracle corresponds to the mapping $(v, i) \mapsto w$, where w is the i^{th} neighbour of v (if it exists) or $w = \perp$ (if v has fewer than i neighbours). The distance between two graphs is the fraction of pairs (v, i) whose outputs differ between the adjacency list mappings.

Our goal in this section will be to construct a QMAP protocol for testing whether a bounded-degree *rapidly-mixing* graph G (i.e., where the last vertex in a random walk of sufficiently large length $\ell = O(\log n)$ starting from any vertex is distributed roughly uniformly), given as an adjacency list oracle, is bipartite or ε-far from every (rapidly-mixing) bipartite graph. Note that this differs from the standard testing setting on graphs by the additional restriction that *both* yes- and no-inputs be rapidly mixing: whether G is bipartite or ε-far from bipartite, for any pair u, v of vertices, the length-ℓ random walk starting from u ends at v with probability between $1/(2n)$ and $2/n$.

Our QMAP protocol for bipartiteness builds on the MAP protocol of [GR18, Theorem 7.1], modifying it to take full advantage of quantum speedups. In the strategy laid out in that work, the classical verifier receives as proof a set S of k vertices that are allegedly on the same side of a bipartition. It repeatedly samples a uniformly random vertex, takes many lazy random walks of length ℓ and,[10] for each of them, records if it stops at a vertex in S as well as the parity of the number of non-lazy steps (where the walk moves to another vertex). If two walks starting from the same vertex end in S with different parities, the verifier has found a witness to the fact that the graph is not bipartite and rejects.

For any proximity parameter ε and $k \leq n/2$, the MAP protocol of [GR18] requires a proof of length $k \log n$ and makes $O(\varepsilon^{-2} \cdot n/k)$ queries. By making the verifier quantum (but using the same classical proof), we obtain improvements in the dependence on both n/k and ε:

Theorem 5.18. *For every $k \leq n/2$, there exists a one-sided QCMAP for deciding whether an n-vertex bounded-degree graph G is bipartite or ε-far from being bipartite, under the promise that G is rapidly-mixing, with proof complexity $k \cdot \log n$ and query complexity $\tilde{O}((n/k)^{2/3} \cdot \varepsilon^{-5/6})$.*

Our strategy to obtain a quantum speedup uses the quantum *collision-finding* algorithm [Amb07, ACL11] to a suitable modification of the verifier strategy outlined above.[11] We note that the function f to which we will apply the collision-finding algorithm is *not* the adjacency list oracle, but one whose unitary representation can be obtained by adjacency list queries and classical computation.

Theorem 5.19 (Quantum collision-finding [ACL11, Theorem 9])**.** *Let $f \colon X \to Y$ be a function given via a unitary oracle and $R \subseteq Y \times Y$ a symmetric binary relation. There exists a quantum algorithm that makes $O(|X|^{2/3} \operatorname{polylog} |Y|)$ queries to f, accepts (with probability 1) if $(x, x') \notin R$ for every distinct $x, x' \in X$, and rejects with probability $2/3$ if there exist distinct $x, x' \in X$ such that $(x, x') \in R$.*

Returning to the aforementioned classical MAP, observe that the information of each set of t random walks of length ℓ starting from a fixed vertex can be represented by a function $f \colon T \to \{0,1\}^2$, where $T \subset \{0,1\}^{\ell'}$ is composed of t strings of length $\ell' = O(\ell)$ (note that each step of the random walk can be performed with $O(1)$ bits, for a total of $\ell' = O(\ell)$ random coins per walk) and $f(r)$ encodes whether the walk reaches a vertex in S as well as its parity when its inner randomness corresponds

[10] A lazy random walk moves from a vertex v to a uniform random neighbour with probability $\frac{d_v}{2d}$ and otherwise stays at v, where d_v is the degree of v and $d = O(1)$ is the graph's degree bound.

[11] We remark that collision-finding here refers to a generalisation of the *element distinctness* algorithm to symmetric relations beyond equality. In particular, it is *not* a quantum algorithm for the "r-to-1 collision problem", where (many) collisions are promised to exist.

to the string r. Now, applying the collision-finding algorithm to this function for each sampled vertex already yields a speedup [ACL11]; but it can be improved by combining this strategy with amplitude amplification (which improves quadratically the number of samples taken as the starting vertices of the random walks), as we show next.

Proof of Theorem 5.18. We first consider the classical MAP outlined earlier when it samples a *single* starting vertex for a random walk, and apply collision-finding to the function f whose domain is T, a random (multi-)set of sequences of coin flips. The proof specifies a subset $S \subseteq L$ of size k, for some L (allegedly) defining a bipartition $V = L \cup R$ of G with $|L| \geq |R|$. The resulting algorithm (which is *not* yet the final QCMAP verifier), is shown in Algorithm 5.4.

Algorithm 5.4: Quantum algorithm for bipartiteness with low detection probability

Input: oracle access to the adjacency list of an n-vertex graph G, as well as explicit access to a proximity parameter $\varepsilon > 0$ and a proof string $\pi \in \{0,1\}^{k \log n}$ (representing a set $S \subset V(G)$ of size k).

```
Step 1: Sample a uniformly random vertex $v \in V(G)$ and select a multi-set
        $T \subset \{0,1\}^{\ell'}$, where $\ell' = O(\ell)$ and $\ell = O(\log n)$, by sampling $O\left(\frac{n}{k} \cdot \frac{\log n}{\varepsilon}\right)$
        bit strings uniformly and independently.
Step 2: Let $f : T \to \{0,1\}^2$ be the function computed by the following subroutine:
            Let $r \in T$ be the input. Take the (lazy) random walk of length $\ell$ starting
            at $v$ using $r$ as the random bits. Let $a \in \{0,1\}$ be the indicator of
            whether the walk stops at a vertex in $S$ and $b \in \{0,1\}$ be the parity of
            the walk (i.e., the number of non-lazy steps). Return $(a, b)$.
Step 3: Execute the algorithm of Theorem 5.19 with respect to $f$ to find $t$ and
        $t'$ such that $f(t) = (a, b)$ and $f(t') = (a', b')$ with $a = a' = 1$ and $b \neq b'$.
        Reject if such a pair is found, and accept otherwise.
```

First, note that the function f contains a collision with respect to the relation $R \subset \{0,1\}^4$ that comprises all pairs of pairs of the type $((1, b), (1, 1-b))$ *if and only if* there are two random walks of length ℓ that start from v end at a vertex in S with different parities.

If G is bipartite with vertex set $L \cup R$ and $S \subseteq L$, every path with both endpoints in S has even length. But the existence of two paths from the same vertex into S with different parities implies the existence of a path of odd length with both endpoints in S; therefore, since the function f does not contain a collision for any starting vertex v, the verifier always accepts when G is bipartite and the proof consists of k elements on the same side of a bipartition.

If G is ε-far from bipartite, then [GR18] (building on [GR99]) show the following: defining a as the indicator of whether a random walk of length $\ell = O(\log n)$ starting from vertex v stops at a vertex in S, and b as the parity of this random walk, then an $\Omega(\varepsilon/\log n)$ fraction of vertices in $V(G)$ are such that *both* $\mathbb{P}[a = 1, b = 0]$ and $\mathbb{P}[a = 1, b = 1]$ are $\Omega(\frac{k\varepsilon}{n \log n})$. Therefore, with probability $\Omega(\varepsilon/\log n)$, the sampled vertex $v \in V(G)$ satisfies this condition. If such a vertex was sampled, with probability $\Omega(1)$, in a sufficiently large number $|T| = O(\frac{n \log n}{k\varepsilon})$ of random walks starting from it there exists a pair of length-ℓ walks that end in S with different parities. Finally, since the algorithm of Theorem 5.19 rejects with constant probability if the function f contains a collision, Algorithm 5.4 rejects with probability $\Omega(\varepsilon/\log n)$.

The quantum collision-finding algorithm computes $O(|T|^{2/3})$ times the function f, each of which simulates a random walk of length $\ell = O(\log n)$. Each step of the random walk requires $O(1)$ queries to the graph G, so that $O(\log n)$ queries are made per walk. The query complexity of Algorithm 5.4 is therefore

$$O\left(|T|^{2/3} \cdot \ell\right) = \tilde{O}\left(\left(\frac{n}{k\varepsilon}\right)^{2/3}\right).$$

Finally, applying amplitude amplification (Theorem 5.9) to this algorithm (rather, more precisely, to a reversible quantum algorithm that is obtained from Algorithm 5.4 by deferring its measurements; see, e.g., [NC16]), we obtain a QCMAP verifier for bipartiteness of bounded-degree graphs with query complexity

$$\tilde{O}\left(\left(\frac{n}{k\varepsilon}\right)^{2/3}\right) \cdot O\left(\sqrt{\frac{\log n}{\varepsilon}}\right) = \tilde{O}\left(\left(\frac{n}{k}\right)^{2/3} \cdot \frac{1}{\varepsilon^{5/6}}\right). \qquad \square$$

We remark that classically testing bipartiteness in the bounded-degree model requires $\Omega(\sqrt{n})$ queries *even under the rapidly-mixing promise*, as shown by Goldreich and Ron [GR02]; therefore, for constant ε, a QCMAP with proof complexity $\tilde{O}(n^{1/4})$ is enough to overcome the testing lower bound, while the MAP of [GR18] requires a proof of length $\tilde{O}(\sqrt{n})$. Of course, a fairer comparison would be to *quantum* testers, for which $\tilde{O}(n^{1/3})$ queries suffice [Amb07] but no nontrivial lower bound is known. The quantum tester is sound without the rapidly-mixing promise, however, so even though a QCMAP with proof length $\tilde{O}(\sqrt{n})$ makes fewer queries than the best known quantum tester, it is not clear how the two algorithms compare.

5.7 A QCMAP lower bound for testing unitaries

The known QMA versus QCMA (oracle) separation of Aaronson and Kuperberg [AK07] is naturally cast as a testing problem, and thus yields a corresponding separation between QMAP and QCMAP. More precisely, we consider the following property of unitaries:

$$\Pi_{AK} = \left\{ U \in \mathbb{C}^{n\times n} : \begin{array}{l} UU^\dagger = U^\dagger U = I \text{ and } \exists \left|\psi\right\rangle \text{ such that } U\left|\psi\right\rangle = -\left|\psi\right\rangle \\ \text{and } U\left|\varphi\right\rangle = \left|\varphi\right\rangle \text{ when } \left\langle\varphi|\psi\right\rangle = 0 \end{array} \right\},$$

where the distance measure $d(U,V) = \sqrt{\frac{1}{n}\cdot \mathrm{Tr}\left((U-V)^\dagger(U-V)\right)}$ is the one induced by the normalised Hilbert-Schmidt norm. When measuring distance to the $n\times n$ matrix $\mathbf{I}$ (for which any orthonormal basis is an eigenvector basis), each eigenvalue λ of U is mapped to the eigenvalue $(\lambda-1)(\lambda^{-1}-1)$ of $(U-I)^\dagger(U-I)$. Therefore, for every $U \in \Pi_{AK}$,

$$d(U,I) = \frac{2}{\sqrt{n}},$$

Then $d(\Pi_{AK}, I) = 2/\sqrt{n}$, so that distinguishing between a unitary in Π_{AK} and $\mathbf{I}$ reduces to testing Π_{AK} with proximity parameter $\varepsilon \leq 2/\sqrt{n}$. Formally,

Theorem 5.20. *For any $\varepsilon \leq 2/\sqrt{n}$, we have $\Pi_{AK} \in \mathsf{QMAP}(\varepsilon, \log n, 1)$ and $\Pi_{AK} \notin \mathsf{QCMAP}(\varepsilon, p, q)$ when $\sqrt{p}\cdot q = o(\sqrt{n})$.*

Proof. Without loss of generality, we assume query access to a *controlled* unitary $U \in \Pi_{AK}$ or to $\mathbf{I}$. Note that, since the identity is ε-far from Π_{AK}, distinguishing between $U \in \Pi_{AK}$ and $\mathbf{I}$ is at least as hard as testing Π_{AK}.

A QMAP algorithm with logarithmic proof length and query complexity 1 is as follows: given a $\log n$-qubit quantum state $\left|\psi\right\rangle$ as proof, apply the unitary to $\left|\psi\right\rangle$, accepting if and only if a phase flip is detected (by measuring and inspecting the outcome of the control qubit). Clearly, the eigenstate with eigenvalue -1 is a proof that causes the algorithm to accept with certainty if $U \in \Pi_{AK}$, while no quantum state is accepted if $U = I$ (also with certainty).

The lower bound is immediate from the one proven in [AK07], which can be rephrased as follows: any QCMA algorithm that receives a proof of length p and makes $q = o(\sqrt{n/p})$ oracle queries to $U \in \Pi_{AK}$ or the identity operator either accepts $\mathbf{I}$ or rejects some element of Π_{AK} with probability at least $1/2$. $\square$

Remark 5.21. In the usual encoding of an oracle for an n-bit string x as a unitary $|i\rangle |b\rangle \mapsto |i\rangle |b \oplus x_i\rangle$, a Hamming distance of $\Theta(1)$ translates into $\Theta(1)$ distance in the Hilbert-Schmidt metric. Therefore, Theorem 5.20 falls short of proving $\mathsf{QMAP} \not\subseteq \mathsf{QCMAP}$ (where the omitted proof and query complexities are polylogarithmic, and the proximity parameter is constant).

5.8 Interaction versus quantum proofs

In this section we compare the power of classical interactive proofs of proximity (IPPs) and non-interactive quantum proofs of proximity (QMAPs), and show that the rather well studied problem of *permutation testing* admits an efficient IPP but no efficient QMAP. In fact, for permutation testing, even an Arthur-Merlin Proof of Proximity is sufficient, as shown in [GLR21]. Informally, an Arthur-Merlin Proof of Proximity (AMP) is a proof system with one round of communication where the verifier sends the first message.

Let Π_P be the property (of *functions* $f : [n] \to [n]$) defined as

$$\Pi_P = \{f : f \text{ is a bijection}\} \ ,$$

i.e., Π_P is the set of all permutations. We note that in an oracle query to f, an algorithm sends $x \in [n]$ and receives (the $\log n$-bit string) $f(x)$; accordingly, a quantum query maps ($2\log n$-qubit states via) $|x\rangle |y\rangle \mapsto |x\rangle |y + f(x)\rangle$. Moreover, distance is measured in terms of the fraction of inputs where functions disagree (rather than with respect to their representations as bit strings), i.e., f and g are ε-far when $|\{x \in [n] : f(x) \neq g(x)\}|/n \geq \varepsilon$.

The separation follows immediately from the two following theorems.

Theorem 5.22 ([GLR21, Lemma 4.2])**.** *For every with $\varepsilon > 0$, There exists an AMP for ε-testing Π_P with query complexity $O(1/\varepsilon)$ and communication complexity $O(\log n/\varepsilon)$, that communicates two messages: the first is sent from verifier to prover and the second from prover to verifier. Therefore,*

$$\Pi_P \in \mathsf{AMP}(\varepsilon, O(\log n/\varepsilon), O(1/\varepsilon), 2) \ .$$

Since $\mathsf{AMP}(\varepsilon, c, q, r) \subseteq \mathsf{IPP}(\varepsilon, c, q, r + 1)$ (by initiating the protocol with a "dummy" message by the prover), the following corollary is immediate.

Corollary 5.8. $\Pi_P \in \mathsf{IPP}(\varepsilon, O(\log n/\varepsilon), O(1/\varepsilon), 3)$.

Having established that Π_P admits an efficient IPP, we must now show it is *not* efficiently testable by a QMAP:

Lemma 5.9 ([ST23, Theorem 1.2])**.** *Any QMAP protocol for testing Π_P with respect to proximity parameter $\varepsilon = \Omega(1)$, using a proof of length p and making q queries, satisfies $p \cdot q^3 = \Omega(n)$; i.e.,*

$$\Pi_P \notin \mathsf{QMAP}(\varepsilon, p, q) \text{ when } p \cdot q^3 = o(n) \ .$$

Note that this implies that either p or q must be $\Omega(n^{1/4})$ in any QMAP protocol for Π_P. Finally, Theorem 5.22 and Lemma 5.9 together imply the main result of this section.

Theorem 5.23. *Let $\Pi_P \subset \{f : [n] \to [n]\}$ be the set of bijective functions from $[n]$ to $[n]$, with the distance between functions f and g defined as $|\{x \in [n] : f(x) \neq g(x)\}|/n$. Then, for any $\varepsilon = \Omega(1)$,*

$$\begin{aligned} &\Pi_P \in \mathsf{IPP}(\varepsilon, O(\log n), O(1), 3) \text{ and} \\ &\Pi_P \notin \mathsf{QMAP}(\varepsilon, p, q) \text{ when } p \cdot q^3 = o(n) \ . \end{aligned}$$

In particular,

$$\mathsf{IPP}(\varepsilon, O(\log n), O(1), 3) \not\subseteq \mathsf{QMAP}(\varepsilon, o(n^{1/4}), o(n^{1/4})) \ .$$

Chapter 6

Streaming zero-knowledge proofs

Overview

We initiate the study of zero-knowledge proofs for data streams. Streaming interactive proofs (SIPs) are well-studied protocols whereby a space-bounded algorithm with one-pass access to a massive stream of data communicates with a powerful but untrusted prover to verify a computation that requires large space.
We define the notion of *zero-knowledge* in the streaming setting and construct zero-knowledge SIPs for the two main building blocks in the streaming interactive proofs literature: the *sumcheck* and *polynomial evaluation* protocols. To the best of our knowledge *all* known streaming interactive proofs are based on either of these tools, and indeed, this allows us to obtain zero-knowledge SIPs for central streaming problems such as index, point and range queries, median, frequency moments, and inner product. Our protocols are efficient in terms of time and space, as well as communication: the space complexity is $\mathrm{polylog}(n)$ and, after a non-interactive setup that uses a random string of near-linear length, the remaining parameters are $n^{o(1)}$.
En route, we develop a toolkit for designing zero-knowledge data stream protocols, consisting of an *algebraic streaming commitment protocol* and a *temporal commitment protocol*. The analysis of our protocols relies on delicate algebraic and information-theoretic arguments and reductions from average-case communication complexity.

Organisation

This chapter is organised as follows. In Section 6.1, we formally define the notion of streaming zero-knowledge and discuss key conceptual points. In Section 6.2 we construct the two commitment protocols that comprise the main components for our polynomial evaluation and sumcheck protocols. We construct the protocols, prove their zero-knowledge property and show applications for them in Sections 6.3 and 6.4, respectively.

6.1 Zero-knowledge streaming interactive proofs

This section motivates and provides a definition of zero-knowledge proofs in the data stream model. We start by discussing the differences between the streaming and the traditional settings as well as establish necessary notation. We then we provide a formal definition in Section 6.1.1.

The notion of zero-knowledge proofs in a computational model should capture the intuition that, when engaged in an interactive protocol, a verifier algorithm V should learn nothing but the truth of some hard-to-compute statement about its input x (e.g., that x is in a language L). For consistency with the general notion we define zero-knowledge for *decision problems* in the streaming model, but remark that the definition extends to search problems in the standard way (i.e., the verifier V learns nothing but a valid solution to the search problem).

In the traditional setting, V can easily store the entirety of x and make polynomial-time computations without the assistance of a prover. This implies that the sensitive information a zero-knowledge proof in this setting must not leak is the result of a computation on x beyond the verifier's reach, i.e., one that requires superpolynomial time to obtain from the information available to V. In the streaming setting, however, the notion of "hard-to-compute" changes dramatically: the model puts *space* as the primary resource, so that computations within the reach of V are those possible with a small amount of space and sequential one-pass access to the input (but arbitrarily large time complexity). Knowledge then essentially corresponds to all information that V cannot compute in low space complexity using its streaming access. As a result, zero-knowledge streaming interactive proofs (zkSIPs) must satisfy a much more stringent requirement: that they not leak any information *about the input x itself* (which in the traditional setting is fully known to the verifier).

In order to capture such a stringent notion of sensitive information, we define zkSIPs as protocols such that no *streaming* algorithm can distinguish a real transcript of the protocol from one that is generated by a (streaming) simulator. To this end, we first recall the formalisation of *streaming interactive proofs* (SIPs) [CTY11] without any zero-knowledge requirement.

Definition 6.1. *A* streaming interactive proof *(SIP) for a language L is an interactive proof defined by a pair (P, V) of algorithms: a computationally unbounded prover P and streaming verifier V with space $s = o(n)$. The verifier engages in an iteractive* *protocol with P and streams, at a predetermined step, the bit string $x \in \{0,1\}^n$, which P also observes.*[1] *At the end of the protocol, V outputs a binary decision $\langle P, V\rangle(x)$ satisfying*

- *(completeness) if $x \in L$, then $\mathbb{P}[\langle P, V\rangle(x) = 1] \geq 2/3$; and*
- *(soundness) if $x \notin L$, then then $\mathbb{P}[\langle P, V\rangle(x) = 1] \leq 1/3$.*

We call s the *space complexity* (of the verifier). Note that, while the constant $1/3$ is arbitrary, soundness amplification does not hold for streaming algorithms due to the need to reread the input; nevertheless, many SIPs (including all those considered in this chapter) allow for improving soundness by a desired factor with a logarithmic increase to their space complexity (see Section 6.2.1). We stress that Definition 6.1 constrains the verifier *only* in terms of space, which allows arbitrarily large time complexities for both prover and verifier. (This is similar to other settings such as communication complexity and property testing, where the primary resources are communication and queries, respectively.)

Loosely speaking, we capture the notion of zero-knowledge in the data stream model by saying that an SIP is zero-knowledge if there exists a streaming *simulator algorithm* S, with roughly the same space as the verifier V, able to simulate a prover-verifier interaction that is indistinguishable from a real one; that is, S generates a *view* of the verifier (defined next) that no *distinguisher* algorithm with power comparable to V (i.e., a streaming algorithm with roughly the same space) can tell apart from a real interaction. We stress that while the distinguisher D is reminiscent of computational zero-knowledge, the security of our protocols is information-theoretic and *does not rely on computational assumptions.*

Definition 6.2. *Let (P, V) be an SIP with a space-s verifier, where P sends k_1 messages to V before the verifier streams its input, and an additional k_2 messages afterwards. Denote the prover's messages by $y_1 \in \{0,1\}^{p_1}, \ldots, y_{k_1+k_2} \in \{0,1\}^{p_{k_1+k_2}}$; the input by x; and the verifier's and prover's internal randomness by r and t, respectively.*

The view *of the verifier $\widetilde{V}$, denoted* $\mathrm{View}_{P,\widetilde{V}}(x, r)$, *is the random variable defined as*

$$\mathrm{View}_{P,\widetilde{V}}(x, r; t) = (r, y_1, \ldots, y_{k_1}, x, y_{k_1+1}, \ldots, y_{k_1+k_2}).[2]$$

[1]The definition could allow for alternating between streaming parts of x and communicating with the prover, as well as adaptively choosing the round(s) on which to read the input. Our protocols do not require this flexibility, however, so we assume the entirety of x is read at a fixed step along the communication protocol.

[2]We note that a more general definition allows the random bits r to be partially streamed throughout the protocol, rather than only in the beginning. This simpler definition suffices to capture the honest V in all of our protocols, but we assume the more general version when (a malicious) $\widetilde{V}$ consumes more randomness than it can store.

While Definition 6.2 is similar to its polynomial-time analogue, we highlight an important distinction: to faithfully correspond to what $\widetilde{V}$ sees, the order in which the view is streamed must be preserved. Indeed, a step-by-step execution of $\widetilde{V}$ in an interaction with P corresponds exactly to its streaming $\mathrm{View}_{P,\widetilde{V}}(x,r)$ one symbol at a time. Order preservation is also consistent with the input stream x being observed by all parties simultaneously (which are, in a simulation, $\widetilde{V}$, the simulator S and a distinguisher D).

6.1.1 Definition

We now ready to give a formal definition of zero-knowledge streaming interactive proofs.

Definition 6.3 (zkSIP). *Let L be a language and (P,V,S) be a triplet where (P,V) is an SIP with a space-s verifier V and S is a streaming* $\mathrm{poly}(s)$*-space simulator with* white-box access *to the verifier, streaming access to the input x and additional query access to a random bit string t.*

(P,V,S) *forms a* zero-knowledge streaming interactive proof *(zkSIP) for L that is secure against space-s' adversaries if, for any space-s algorithm $\widetilde{V}$ and $x \in L$, the random variables* $\mathrm{View}_{P,\widetilde{V}}(x,r)$ *and* $S(\widetilde{V},x,r)$ *are indistinguishable by any streaming space-s' algorithm. That is, for every space-s' streaming algorithm D,*

$$\left|\mathbb{P}\left[D\big(\mathrm{View}_{P,\widetilde{V}}(x,r)\big) \text{ accepts}\right] - \mathbb{P}\left[D\big(S(\widetilde{V},x,r)\big) \text{ accepts}\right]\right| = o(1).$$

We note that all our applications have $s = \mathrm{polylog}(n)$, and the protocols are secure against adversaries with any space $s' = \mathrm{poly}(s)$ (see Remark 6.9).

Remark 6.1. Recall that the analogue of Definition 6.3 in the polynomial-time setting requires a much stronger notion of indistinguishability: *negligible* (i.e., sub-inverse-polynomial), rather than $o(1)$, bias. This is necessary for the notion to be robust with respect to poly-time algorithms, as otherwise repeating polynomially many executions of D would boost its success probability arbitrarily close to 1.

This raises a number of interesting questions on the achievable notions of security for zkSIPs: can we obtain tighter bounds, such as $1/\mathrm{poly}(n)$ or negligible? (Perhaps even in the statistical case?) An answer to each such question ensures security against one type of adversary (i.e., distinguisher): we will study the natural threat model where all parties are streaming algorithms and argue why $o(1)$ is a sufficient bound in this case. Before doing so, however, we briefly discuss an important alternative.

As explained above, streaming verifiers secure against polynomial-time adversaries require negligible distinguishability. This has been previously studied, most notably for zero-knowledge interactive proofs that reduce to evaluating low-degree polynomials defined by the input and allow for it to be processed in a streaming fashion, such as [GKR15]. (We stress, however, that such protocols rely on computational assumptions.) An interesting question that we leave to future work is whether zkSIPs can *simultaneously* achieve security against different adversaries – e.g., with negligible bias for poly-time distinguishers (under cryptographic assumptions) in addition to subconstant bias for streaming distinguishers.

Recall that a key distinction between the poly-time and streaming settings is the *one-pass* restriction of the latter, which prevents even a single repetition of (a streaming) D – indeed, INDEX trivialises with 2 passes (as do many fundamental streaming problems). In other words, as the common technique of *amplification* is unavailable in the streaming model, $o(1)$ bias is a sufficiently robust requirement that guarantees the probability of information leakage tends to 0. (We note that the weaker requirement of arbitrarily small constant bias would also suffice, i.e., the existence of $(P_\varepsilon, V_\varepsilon, S_\varepsilon)$ achieving ε bias for every $\varepsilon > 0$. We adopt the simpler and stronger subconstant version, which our protocols satisfy.)

The streaming simulator. For technical reasons, the simulator is given white-box access to the verifier and explicit access to a random string. We stress that this auxiliary information is completely independent of the input. This can viewed as allowing the verifier to obtain some computation about auxiliary information (about its own strategy, or a uniformly chosen random string), but learn absolutely *zero information about the input stream x.*

While white-box access gives the simulator S knowledge of any function of the verifier's strategy, we do not require such generality; indeed, we will only be interested in questions about the most likely messages that $\widetilde{V}$ may send at a single point of the protocol. As such, the weaker definition that follows is sufficient.

Definition 6.4. *Let A be a space-s streaming algorithm that reads an n-bit string y and outputs an m-bit string z. We define* white-box access *to A as oracle access to a function $\mathcal{W}$ with two inputs, a snapshot $b \in \{0,1\}^s$ and a candidate output $z \in \{0,1\}^m$; the oracle returns the maximum probability* over all inputs y *with which A, starting with memory state b, outputs z; that is,*

$$\mathcal{W}(b,z) = \max_{y \in \{0,1\}^n} \left\{ \mathbb{P}[A(y) \text{ outputs } z \text{ when its initial snapshot is } b] \right\}.$$

Remark 6.2. While the honest verifier V does not use a large random string, malicious verifiers $\widetilde{V}$ with this additional resource can readily be simulated by S as above. We assume hereafter that $\widetilde{V}$ has the same resources as the honest verifier, but note that the simulations extend straightforwardly to verifiers with both white-box access (to their strategies) and query access to a random string.

6.2 Algebraic and temporal commitments

A commitment protocol is a two-party protocol (or, more accurately, a pair of protocols) that allows the transmission of a message from one party to another to be split into two parts: a *commitment*, where the message is transmitted in a form that cannot be interpreted by the recipient; followed, at some point in the future, by a *decommitment*, where the sender transmits additional information with which the recipient can read the message. (A useful analogy is that the commitment amounts to sending a locked box containing the message, and the decommitment to sending the key.)

In the standard setting [Blu83] we have two parties: a sender and a receiver, which we will refer to as prover and verifier, respectively. The prover wishes to communicate a symbol α, and does so by first choosing a random *key* k and sending another string $c = \mathsf{commit}(\alpha, k)$. Then, at some point in the future, prover and verifier engage in a protocol at the end of which the receiver obtains $\alpha = \mathsf{decommit}(c)$. (We will refer to the streaming analogue as a commitment *protocol*, rather than scheme, to avoid ambiguity with the polynomial-time analogue.)

Commitment protocols are extremely useful components for the construction of interactive protocols, and should satisfy two properties: *hiding*, i.e., the commitment alone should prevent the verifier from obtaining a non-negligible amount of information about the message α; and *binding*, i.e., the prover should not be able to decommit to a message that differs from the one it committed to. We will construct a commitment protocol whose hiding property follows from the average-case hardness of SEARCH-INDEX for streaming algorithms, while binding follows from the soundness of the pep protocol (which we introduce formally in Section 6.2.1).

We first formally define streaming commitment protocols. We note that while the definition that follows can be generalised,[3] it suffices to capture our constructions.

Definition 6.5. *A* streaming commitment protocol *for alphabet* Γ *(with security parameter* p*) and space bound* s *consists of a function* $\mathsf{commit} : \Gamma \times K \to C$*, where*

[3]A natural generalisation is to parameterise the bias in the hiding property as well as the completeness and soundness in binding by $\varepsilon_b, \varepsilon_c, \varepsilon_s \in (0, 1)$; our definition has $\varepsilon_b, \varepsilon_s = o(1)$ and $\varepsilon_c = 0$.

$K \subseteq \{0,1\}^p$ is the set of keys and C is the set of commitments, and a space-s SIP (P,V) which satisfy the following conditions.

- Hiding: *Fix any pair of distinct messages $\alpha, \beta \in \Gamma$ and sample $k \sim K$. Set $c = \mathsf{commit}(\alpha) = \mathsf{commit}(\alpha, k)$ and $c' = \mathsf{commit}(\beta) = \mathsf{commit}(\beta, k)$. Every (streaming) space-s distinguisher D tells the two commitments apart with at most subconstant bias (with respect to the parameter p); that is,*

$$\Big|\mathbb{P}[D(c) \text{ accepts}] - \mathbb{P}[D(c') \text{ accepts}]\Big| = o(1).$$

- Binding: *Fix $k \in K$ and $\alpha \in \Gamma$. Then*

$$\mathbb{P}\big[\langle P,V\rangle\big(\mathsf{commit}(\alpha,k),\alpha\big) = 1\big] = 1,$$

  and for any $\beta \neq \alpha$,

$$\mathbb{P}\big[\langle P,V\rangle\big(\mathsf{commit}(\alpha,k),\beta\big) = 1\big] = o(1).$$

Note that, with some abuse of notation, the binding condition corresponds to (P,V) being an SIP for the language $L = \{(\mathsf{commit}(\alpha,k),\alpha) : \alpha \in \Gamma, k \in K\}$.

The next sections introduce the commitment protocols we will use to build our protocols. Section 6.2.1 begins by defining the concepts and tools we build upon: low-degree extensions and the polynomial evaluation protocol (pep). In Section 6.2.2, we use them to construct a basic scheme that allows for the communication of a single symbol (which we use as a stepping stone), based on the hardness of INDEX (or, more accurately, SEARCH-INDEX); in it, the keys are simply long strings paired with a coordinate, i.e., $K = \Gamma^p \times [p]$, and commitments are keys appended with a single extra symbol (i.e., $C \subset \Gamma^{p+1} \times [p]$).

Section 6.2.3 then extends the construction of Section 6.2.2 into an *algebraic* commitment protocol, which allows for the commitment of low-degree polynomials. In both the basic and algebraic schemes, hiding is achieved by overwhelming V with "too much information", and can only be broken if a malicious verifier is lucky enough to retain a critical fragment of the information stream; indeed, as we will see, breaking it amounts to solving INDEX. Binding, on the other hand, relies on the pep protocol, which we introduce in the next section.

While commitment protocols are not a prerequisite for a zero-knowledge protocol, they also serve as inspiration for our second main component: Section 6.2.4 shows how the verifier can perform a *temporal commitment* to show its alleged internal randomness is uncorrelated with its input, and thus that it is not behaving maliciously.

6.2.1 Low-degree extensions and polynomial evaluation

Fingerprinting is a technique that enables streaming algorithms to approximately verify an arbitrary coordinate of a long string in small space. It exploits *low-degree extensions* (LDEs), extremely useful objects in the design of interactive proofs more broadly. (Recall Section 2.5.)

Given a data set x, viewed as a string of n elements in a finite field $\mathbb{F} = \mathbb{F}_q$, an LDE is a low-degree polynomial that interpolates every data point. More precisely, we may view x as a function $x : [n] \to \mathbb{F}$; given a *dimension* m and defining the *degree* d as the smallest (positive) integer such that $n \leq (d+1)^m$, we can also view $x : [d+1]^m \to \mathbb{F}$ by some canonical injection $[n] \hookrightarrow [d+1]^m$ (padding with zeroes if $n < (d+1)^m$). Then, as long as $q > d$, we can also view (via another canonical injection $[d+1] \hookrightarrow \mathbb{F}$) the data set as the restriction of a function from $\mathbb{F}^m$ to $\mathbb{F}$.

Standard properties of polynomials imply that if this function is an m-variate polynomial of individual degree d, then the extension is unique; we thus denote by $\hat{x} : \mathbb{F}^m \to \mathbb{F}$ the unique degree-d polynomial whose restriction to $[n]$ is equal to x. Explicitly, with $(i_1, \ldots, i_m)$ as the image of i by $[n] \hookrightarrow \mathbb{F}^m$,

$$\hat{x} = \sum_{i=1}^{n} x_i \chi_i = \sum_{i_1,\ldots,i_m \in [d+1]} x_{i_1,\ldots,i_m} \chi_{i_1,\ldots,i_m}$$

where the χ_i are the Lagrange basis polynomials, given by

$$\chi_i(\alpha_1, \ldots, \alpha_m) \coloneqq \prod_{j=1}^{m} \prod_{\substack{k=1 \\ k \neq i_j}}^{d+1} \frac{\alpha_j - k}{i_j - k}$$

(viewing $k \in [d+1]$ as an element of $\mathbb{F}$); equivalently, the Lagrange polynomials are the unique m-variate degree-d polynomials satisfying $\chi_i(j) = \mathbb{1}[i = j]$ when $i, j \in [d+1]$. We note that LDEs and Lagrange polynomials can equivalently be defined with an injection from $\{0\} \cup [d]$, rather than $[d+1]$, to $\mathbb{F}$; then they satisfy the previous condition for all $0 \leq i, j \leq d$. We will use the characterisation that is most convenient, which will be clear from context (e.g., an LDE that involves the evaluation of a polynomial at 0 is of the latter type).

We will also use $\boldsymbol{\chi}(\boldsymbol{\alpha})$ to denote the vector $\big(\chi_1(\boldsymbol{\alpha}), \ldots, \chi_n(\boldsymbol{\alpha})\big)$ of evaluations of Lagrange polynomials; note that this allows us to write $\hat{x}(\boldsymbol{\alpha})$ as the dot product $\boldsymbol{\chi}(\boldsymbol{\alpha}) \cdot x$ of n-dimensional vectors.

Now, given a string $x \in \mathbb{F}^n$, a *fingerprint* is simply an evaluation of the LDE of x at a random point, that is, $\hat{x}(\boldsymbol{\rho})$ with $\boldsymbol{\rho} \sim \mathbb{F}^m$. The key property of fingerprints is that they are extremely unlikely to match for two different strings when the underlying field is large enough, as a consequence of the Schwartz-Zippel lemma [Sch80, Rab81].

Lemma 6.1 (Schwartz-Zippel)**.** *If $x, y \in \mathbb{F}_q^n$ are distinct, then $\mathbb{P}_{\boldsymbol{\rho}\sim\mathbb{F}^m}\left[\hat{x}(\boldsymbol{\rho}) = \hat{y}(\boldsymbol{\rho})\right] \leq dm/q$.*

Importantly for streaming algorithms, fingerprints can be computed with $O(dm)$ time per entry of the input and $O(m)$ field elements (thus $O(m \log q)$ bits) of space [CTY11].

The polynomial evaluation protocol is an interactive proof that enables a streaming verifier with a single random evaluation $f(\boldsymbol{\rho})$ of a degree-d polynomial $f : \mathbb{F}^m \to \mathbb{F}$ to evaluate f at any other point, assisted by a prover with knowledge of f in its entirety. Note that the prover could help the verifier compute f at a point (non-interactively) by simply sending an interpolating set of the polynomial; but any such set has size $(d+1)^m$. The pep (polynomial evaluation) protocol, detailed in Protocol 6.1, allows us to reduce the communication from $O(d^m \log q)$ to $O(dm \log q)$ by adding interaction.

In order to better compare the original pep protocol with the zero-knowledge version that we will construct, we consider a general problem that the protocol is able to solve (as in [CCM+19]). We use f as shorthand for a mapping $x \mapsto f^x$ (or, equivalently, a set $f \subseteq \{f^x : x \in \mathbb{F}^n\}$) where one evaluation $f^x(\boldsymbol{\rho})$ can be computed by a space-bounded algorithm that streams x. The problem PEP(f, α) is to decide whether $f^x(\boldsymbol{\beta}) = \alpha$ when the input stream is x followed by an evaluation point $\boldsymbol{\beta} \in \mathbb{F}^m$.

Protocol 6.1: pep(f, α)

Input: Explicit access to $\alpha \in \mathbb{F}$ and a set $f \subseteq \{f^x : x \in \mathbb{F}^n\}$ of m-variate degree-d polynomials over $\mathbb{F}$. Streaming access to $(x, \boldsymbol{\beta}) \in \mathbb{F}^n \times \mathbb{F}^m$.

V: Sample $\boldsymbol{\rho} \sim \mathbb{F}^m$. Stream x and compute $f^x(\boldsymbol{\rho})$. Store $\boldsymbol{\beta}$.

Compute the line $L : \mathbb{F} \to \mathbb{F}^m$ such that $L(0) = \boldsymbol{\beta}$ and $L(\rho) = \boldsymbol{\rho}$ with $\rho \sim \mathbb{F}$, then send L to the prover.

P: Compute and send $f^x_{|L}$.[4]

V: Compute $g(\rho)$, where $g : \mathbb{F} \to \mathbb{F}$ is the degree-dm low-degree extension of the sequence of evaluations sent by P such that $g(0) = \alpha$.[5] Accept if $g(\rho) = f^x(\boldsymbol{\rho})$ and reject otherwise.

Assuming an evaluation of f^x can be computed by streaming x with $O(m \log q)$ space, Protocol 6.1 is a streaming interactive proof for $\textsc{pep}(f, \alpha)$ with communication complexity $O(dm \log q)$ and verifier space complexity $O(m \log q)$. We note that $\texttt{pep}(f, \alpha)$ can easily be modified into an algorithm for a search problem without a candidate value α for $f^x(\boldsymbol{\beta})$, by having V output $g(0)$ instead of accepting.

It is clear that V accepts in Protocol 6.1 when P is honest; the protocol's soundness relies on the fact that if the prover were to send an incorrect $g \neq f^x_{|L}$, it is highly unlikely that it will agree with the verifier's evaluation at the (unknown) location $\boldsymbol{\rho}$.

In conjuction with the streaming nature of LDEs, (the search version of) Protocol 6.1 yields a simple and efficient streaming interactive proof for $\textsc{search-index}$. This SIP, introduced by [CCM+15], has $O(\log n \log \log n)$ space and communication complexities for a stream $(x, j) \in \mathbb{F}^n \times [n]$ where $q = |\mathbb{F}| = \operatorname{polylog}(n)$ (and $\boldsymbol{\beta} \in \mathbb{F}^m$ is the identification of j); it is simply an instantiation of $\texttt{pep}$ where $d = 2$, $m = \log n$ and the function $f^x = \hat{x}$ is the m-variate (multilinear) LDE of x,[6] an evaluation $\hat{x}(\boldsymbol{\rho})$ of which can be computed incrementally as values of x are revealed in the stream. Then $\hat{x}(\boldsymbol{\rho}) = \hat{x}_{|L}(\rho)$ allows the verifier to check that the prover is being honest (i.e., that the polynomial it sent is $\hat{x}_{|L}$), as well as to learn $x_j = \hat{x}(j) = \hat{x}_{|L}(0)$.

Observe that $\texttt{pep}$ is *not* zero knowledge: the verifier learns all of $f^x_{|L}$, which it is not be able to construct by virtue of only learning $\boldsymbol{\beta}$ (and thus L) *after* streaming x. Note, however, that the *honest* verifier only inspects two evaluations of $f^x_{|L}$, namely, at 0 and ρ. In the following sections we construct a commitment protocol that lets the prover only reveal information about these two points, without sacrificing soundness.

6.2.2 A prover-to-verifier commitment protocol

Our commitment protocol, designed to allow an unbounded-space sender to commit to a streaming receiver, directly uses the (average-case) hardness of the $\textsc{index}$ problem. By sending a message hidden at a random coordinate, we exploit the fact that any

[4] Recall that the line L and $f^x_{|L}$ are sent in a canonical form: L as the evaluation $L(1)$ and $f^x_{|L}$ as the vector $\left(f^x \circ L(i) : i \in [dm]\right)$. (There is no need to send $L(0) = \boldsymbol{\beta}$ or $f^x_{|L}(0) = f^x(\boldsymbol{\beta}) = \alpha$, as they are known to V.)

[5] Note that the Lagrange polynomials in this case satisfy $\chi_i(j) = \mathbb{1}[i = j]$ for all $0 \leq i, j \leq dm$.

[6] The space complexity can be reduced to $O(\log n)$ with the choice of parameters for q, d and m in Corollary 6.1.

streaming algorithm requires a linear amount of space to be able to recall a random item from a string after it has been seen. We begin by formally defining (the search and decision versions of) INDEX *in the one-way communication complexity model.*

Definition 6.6. SEARCH-INDEX, over alphabet Γ and with message length s, is the one-way communication problem defined as follows: Alice receives a string $x \in \Gamma^n$ and sends Bob an s-bit message $a = A(x)$. Bob receives, besides $a \in \{0,1\}^s$, an index $j \in [n]$, and outputs a symbol $b = B(a, j) \in \Gamma$. The execution succeeds if $b = x_j$.

Definition 6.7. DECISION-INDEX(α) *(with alphabet Γ and message length s) is the one-way communication problem defined as follows: Alice receives a string $x \in \Gamma^n$ and sends Bob an s-bit message $a = A(x)$. Bob receives, besides Alice's message, an index $j \in [n]$, and outputs a bit $b = B(a, j) \in \{0,1\}$. The execution succeeds if $b = 1$ when $x_j = \alpha$, and $b = 0$ otherwise.*

It is well known that INDEX is extremely hard, even *on average* and in the one-way communication model *with shared randomness.*

Proposition 6.1. *Any one-way communication protocol (A, B) for* SEARCH-INDEX that sends a message of length s satisfies

$$\mathbb{P}_{\substack{x \sim \Gamma^p \\ j \sim [p]}}\left[B\big(A(x), j\big) = x_j\right] = \frac{1}{|\Gamma|} + O\left(\sqrt{\frac{s}{p}}\right).$$

In other words, the chance of correctly recalling a random symbol is at best slightly better than uniform guessing if the string p is much longer than the message length s of the protocol. We note that this bound was known for $\Gamma = \{0,1\}$ [RY20], but it extends to larger alphabets; we now provide a proof of this fact for completeness.

Proof of Proposition 6.1. Define, for ease of notation, $\gamma = |\Gamma|$. We follow the strategy used in [RY20] for the binary case. First, note that by the minimax theorem we may assume Alice's and Bob's strategies are deterministic; i.e., that Alice sends $A(x) \in \{0,1\}^s$ and Bob outputs $B(A(x), j) \in \Gamma$ for some functions A and B.

Let λ be the distribution of Alice's message $A = A(x)$ induced by the (uniform) distribution of x, partitioning Γ^p into $\{P_a\}$ where $P_a = A^{-1}(a) = \{x \in \Gamma^p : A(x) = a\}$. Note that the distribution of x conditioned on $A = a$ is uniform over P_a, and that

$\mathbb{P}_{A\sim\lambda}[A=a] = |P_a|/\gamma^p$. Then,

$$\begin{aligned}
\mathbb{P}_{\substack{x\sim\Gamma^p\\ j\sim[p]}}[\text{Bob outputs } x_j] &= \sum_{a\in\{0,1\}^s} \mathbb{P}_{x\sim\Gamma^p}[A(x)=a]\cdot \mathbb{P}_{\substack{x\sim\Gamma^p\\ j\sim[p]}}\big[b(a,j)=x_j \;\big|\; A(x)=a\big] \\
&= \sum_{a\in\{0,1\}^s} \mathbb{P}_{A\sim\lambda}[A=a]\cdot \mathbb{P}_{\substack{x\sim P_a\\ j\sim[p]}}\big[b(a,j)=x_j\big] \\
&= \mathbb{E}_{\substack{A\sim\lambda\\ j\sim[p]}}\left[\mathbb{P}_{x\sim P_A}\big[b(A,j)=x_j\big]\right] \\
&\le \mathbb{E}_{\substack{A\sim\lambda\\ j\sim[p]}}\left[\max_{\alpha\in\Gamma}\{\mathbb{P}_{x\sim P_A}[x_j=\alpha]\}\right], \qquad (6.1)
\end{aligned}$$

so that we only need to bound the latter expression; note that the inequality shows Bob's optimal strategy is to output the most frequent symbol at the j^{th} coordinate in P_A.

Now, define μ as the uniform distribution over Γ and $\mu_{i,a}$ as the distribution of x_i when $x\sim P_a$ (i.e., the distribution of x_i when $x\sim\Gamma^p$ conditioned on $A(x)=a$). Then, by Pinsker's inequality (Eq. 2.10), for all $a\in \operatorname{Im} A$ and $i\in[p]$ we have

$$\|\mu_{i,a}-\mu\|^2 \le \frac{\mathrm{KL}\big(\mu_{i,a}\;||\;\mu\big)}{2\ln 2}$$

(where we use $\|\cdot\|$ as shorthand for the 2-norm $\|\cdot\|_2$). Since the inequality holds for all a and i, then it also holds for the convex combination corresponding to taking $A\sim\lambda$ and $j\sim[p]$ independently (i.e., whose coefficients are $\mathbb{P}[A=a, j=i] = \frac{|P_a|}{\gamma^p p}$). Therefore,

$$\begin{aligned}
\mathbb{E}_{\substack{A\sim\lambda\\ j\sim[p]}}\left[\|\mu_{j,A}-\mu\|^2\right] &= \frac{1}{p}\sum_{i=1}^{p}\mathbb{E}_{A\sim\lambda}\left[\|\mu_{i,A}-\mu\|^2\right] \\
&\le \frac{1}{2p\ln 2}\sum_{i=1}^{p}\mathbb{E}_{A\sim\lambda}\left[\mathrm{KL}(\mu_{i,A}\;||\;\mu)\right] \\
&= \frac{1}{2p\ln 2}\sum_{i=1}^{p} I(A:x_i),
\end{aligned}$$

where the last equality follows by the definition of mutual information (Eq. 2.11). By convexity of $z\mapsto z^2$, we have

$$\begin{aligned}
\mathbb{E}_{\substack{A\sim\lambda\\ j\sim[p]}}\left[\|\mu_{j,A}-\mu\|\right]^2 &\le \mathbb{E}_{\substack{A\sim\lambda\\ j\sim[p]}}\left[\|\mu_{j,A}-\mu\|^2\right] \\
&\le \frac{1}{2p\ln 2}\sum_{i=1}^{p} I(A:x_i).
\end{aligned}$$

Recall that $\mu_{i,a}(\alpha) = \mathbb{P}_{x\sim P_a}[x_i = \alpha]$. Comparing this value with the average mass $1/\gamma$, we have

$$\begin{aligned}
\mathbb{E}_{\substack{A\sim\lambda\\ j\sim[p]}}\left[\max_{\alpha\in\Gamma}\{\mathbb{P}_{x\sim P_A}[x_j=\alpha]\}\right] - \frac{1}{\gamma} &= \mathbb{E}_{\substack{A\sim\lambda\\ j\sim[p]}}\left[\max_{\alpha\in\Gamma}\left\{\mu_{j,A}(\alpha) - \frac{1}{\gamma}\right\}\right] \\
&\le \mathbb{E}_{\substack{A\sim\lambda\\ j\sim[p]}}\left[\max_{\alpha\in\Gamma}\left\{\left|\mu_{j,A}(\alpha) - \frac{1}{\gamma}\right|\right\}\right] \\
&\le \mathbb{E}_{\substack{A\sim\lambda\\ j\sim[p]}}\left[\|\mu_{j,A} - \mu\|\right] \\
&\le \sqrt{\frac{\sum_{i=1}^p I(A:x_i)}{2p\ln 2}},
\end{aligned}$$

so that using Eq. 6.1 and rearranging,

$$\mathbb{P}_{\substack{x\sim\Gamma^p\\ j\sim[p]}}[\text{Bob outputs } x_j] \le \frac{1}{\gamma} + \sqrt{\frac{\sum_{i=1}^p I(A:x_i)}{2p\ln 2}}.$$

The theorem thus reduces to showing $\sum_{i=1}^p I(A:x_i) \le s$. By standard information-theoretic equivalences and inequalities,

$$\begin{aligned}
\sum_{i=1}^p I(A:x_i) &= \sum_{i=1}^p \big(H(x_i) - H(x_i|A)\big) && \text{(by Eq. 2.11)}\\
&= H(x) - \sum_{i=1}^p H(x_j|A) && \text{(by Eq. 2.7)}\\
&\le H(x) - \sum_{i=1}^n H(x_i|x_1,\ldots,x_{i-1},A) && \text{(by Eq. 2.6)}\\
&= H(x) - H(x|A) && \text{(by Eq. 2.8)}\\
&= I(A:x) \le H(A) && \text{(by Eq. 2.11)}\\
&\le s && \text{(by Eq. 2.5)}
\end{aligned}$$

and the result follows. □

The commitment phase of our scheme exploits this hardness result directly: we take $\Gamma \hookrightarrow \mathbb{F}$ where $\mathbb{F}$ is a large enough finite field (which will allow us to use `pep` to decommit) and have P send the triple $(y, \alpha - y_k, k)$ for random y and k as a commitment to α. (In particular, the commitment key is a random string-coordinate pair (y,k)). Loosely speaking, the protocol has the sender communicate a random stream y with the message hidden at a random coordinate k, which is revealed after y.

The honest verifier keeps a (random) fingerprint of y, which it can use to validate the message at y_k (see Protocol 6.2), while the decommit stage simply instantiates pep appropriately (see Protocol 6.3). We note that the inputs listed in the description of the protocols are those available to the verifier.

Protocol 6.2: commit(α)

Input: explicit access to $p, d, m, q \in \mathbb{N}$ with $p \leq d^m$, $q > d$ and $\mathbb{F} = \mathbb{F}_q$. Streaming access to $y \sim \mathbb{F}^p$ followed by a correction $\gamma \in \mathbb{F}$ and a coordinate $k \sim [p]$.

$\boldsymbol{V}$: Sample $\boldsymbol{\rho} \sim \mathbb{F}^m$ and compute $\hat{y}(\boldsymbol{\rho}) = \sum_{i=1}^{p} \chi_i(\boldsymbol{\rho}) y_i$ while streaming y.

Store $\boldsymbol{\rho}, k, \gamma$ and $\hat{y}(\boldsymbol{\rho})$.

Protocol 6.3: decommit(α, y, k)

Input: $\alpha \in \mathbb{F}$, as well as the (parameters and) values stored in the commit stage: $k, \gamma, \boldsymbol{\rho}, \hat{y}(\boldsymbol{\rho})$.

$\boldsymbol{V}$: Compute and send the line $L : \mathbb{F} \to \mathbb{F}^m$ such that $L(0) = k$ and $L(\rho) = \boldsymbol{\rho}$ with $\rho \sim \mathbb{F}$.

$\boldsymbol{P}$: Send $\hat{y}_{|L}$.

$\boldsymbol{V}$: Compute $g(\rho)$ and $g(0)$, where $g : \mathbb{F} \to \mathbb{F}$ is the degree-dm extension of the sequence of evaluations sent by P.

Accept if $g(\rho) = \hat{y}(\boldsymbol{\rho})$ and $g(0) + \gamma = \alpha$, rejecting otherwise.

Now, we show that Protocols 6.2 and 6.3 form a streaming commitment protocol, i.e., they satisfy the hiding and binding properties of Definition 6.5 if p is large enough; these follow from the hardness of SEARCH-INDEX and the soundness of pep, respectively.

Theorem 6.3. *Protocols 6.2 and 6.3 form a streaming commitment protocol with space complexity $s = O(m \log q)$ when $p = q^3$ and $dm = \mathrm{polylog}(q)$. The protocol is secure against* $\mathrm{poly}(s)$*-space adversaries and communicates $O(q^3 \log q)$ bits.*

Proof. First, note that the communication complexity is dominated by the prover sending $p = q^3$ field elements in the commit step, for a total of $O(q^3 \log q)$ bits.

The binding property is an immediate consequence of the completeness and soundness of pep: if P is honest, i.e., sends the correction $\gamma = \alpha - y_k$ in the commit stage and the polynomial $\hat{y}_{|L}$ in the decommit stage, then V accepts, as $\hat{y}_{|L}(\rho) = \hat{y}(\boldsymbol{\rho})$ and $\hat{y}_{|L}(0) + \gamma = \alpha$. (Recall that the line L satisfies $L(0) = k$ and $L(\rho) = \boldsymbol{\rho}$.)

Now, suppose the prover replies with a polynomial g such that $g(0) \neq y_k = \hat{y}(k) = \hat{y}_{|L}(0)$; then the Schwartz-Zippel lemma (Lemma 6.1) implies $\hat{y}(\boldsymbol{\rho}) = \hat{y}_{|L}(\rho) \neq g(\rho)$ except with probability $dm/q = o(1)$, in which case V rejects.[7] Note that the verifier only needs to store $\boldsymbol{\rho} \in \mathbb{F}^m$, $k \in [p]$ and a constant number of additional field elements, for a space complexity of $O(m \log q + \log p) = O(m \log q)$.

To show the hiding property, assume towards contradiction that there exists a streaming algorithm D with space $\mathrm{poly}(s) = \mathrm{polylog}(q)$ that distinguishes commitments between some $\alpha \in \mathbb{F}$ and $\alpha' \in \mathbb{F} \setminus \{\alpha\}$ with constant bias:[8] that is,

$$\mathbb{P}_{\substack{y \sim \mathbb{F}^p \\ k \sim [p]}}\big[D(y, k, \alpha - y_k) \text{ accepts}\big] - \mathbb{P}_{\substack{y \sim \mathbb{F}^p \\ k \sim [p]}}\big[D(y, k, \alpha' - y_k) \text{ accepts}\big] \geq \varepsilon$$

for some $\varepsilon = \Omega(1)$. Now consider the following algorithm A for SEARCH-INDEX over the alphabet $\mathbb{F}$ with input (x, j): simulate D on the stream (x, γ, j) where $\gamma \sim \mathbb{F}$; output $\alpha - \gamma$ if D accepts, and otherwise output $\alpha' - \gamma$. Note that A outputs correctly exactly when $\gamma = \alpha - y_k$ and D accepts, or $\gamma = \alpha' - y_k$ and D rejects; moreover, A can simulate D with constant space overhead, so that its space complexity is also $\mathrm{polylog}(q)$. We will now show that A solves SEARCH-INDEX with a bias that is too large, contradicting Proposition 6.1.

$$\begin{aligned}
&\mathbb{P}_{\substack{x \sim \mathbb{F}^p \\ j \sim [p]}}\big[A(x, j) = x_j\big] \\
&\quad = \frac{1}{q} \cdot \mathbb{P}_{\substack{x \sim \mathbb{F}^p \\ j \sim [p]}}\big[D(x, j, \alpha - x_j) \text{ accepts}\big] + \frac{1}{q} \cdot \mathbb{P}_{\substack{x \sim \mathbb{F}^p \\ j \sim [p]}}\big[D(x, j, \alpha' - x_j) \text{ rejects}\big] \\
&\quad = \frac{1}{q}\left(1 + \mathbb{P}_{\substack{x \sim \mathbb{F}^p \\ j \sim [p]}}\big[D(x, j, \alpha - x_j) \text{ accepts}\big] - \mathbb{P}_{\substack{x \sim \mathbb{F}^p \\ j \sim [p]}}\big[D(x, j, \alpha' - x_j) \text{ accepts}\big]\right) \\
&\quad \geq \frac{1 + \varepsilon}{q} \\
&\quad = \frac{1}{q} + \Omega\left(\frac{1}{q}\right).
\end{aligned}$$

Since $1/q = \sqrt{q/p} = \omega\left(\sqrt{\mathrm{poly}(s)/p}\right)$, owing to $s = \mathrm{polylog}(q)$, the result follows. $\square$

[7]We remark that ρ need not be sampled from the entire field; the same result holds if $\rho \sim R \subset \mathbb{F}$ when R is large enough. This will be useful in proving that our protocols for PEP and SUMCHECK are zero-knowledge.

[8]Note that allowing $\mathrm{poly}(s)$ space for D will imply a space-robust indistinguishability property; bounding it by, say, $\tilde{O}(s)$ or $O(s^2)$ would prove a weaker but still nontrivial statement.

Remark 6.4. Just as in pep, the verifier learns much more than than the message $\hat{y}_{|L}(0) = \alpha \in \mathbb{F}$: it learns all of $\hat{y}_{|L}$. Crucially, however, the additional information consists of *random field elements uncorrelated with* α. This enables the commitment protocol laid out in this section to be proven zero-knowledge when the simulator has read-only access to a large random string t, as in Definition 6.3. (More accurately, such a simulator can perfectly generate the random variable that corresponds to the view resulting from the commit followed by the decommit steps.)

Indeed, a simulator with space $O(m \log q)$ and query access to $y \sim \mathbb{F}^p$ may sample $k \sim [p]$ and send $(y, \alpha - y_k, k)$ in the commit step; then, in decommit, after receiving the line L, it computes and sends $\hat{y}_{|L} = \big(\hat{y}_{|L}(i) : i \in \{0\} \cup [dm]\big)$ by reading the string y an additional $dm + 1$ times, computing and sending one LDE evaluation at a time.

However, this basic commitment protocol is not yet sufficient. As discussed in Section 3.3.2, it allows P to commit (and decommit) to a single field element; but the prover should be able to commit to a polynomial and decommit to a single evaluation thereof. In the next section we show how to accomplish this, by modifying our scheme to make it *algebraic*.

6.2.3 Making the commitment algebraic

In this section, we will show how to modify the commitment protocol laid out in Section 6.2.2 so that the prover can commit to ℓ messages and decommit to *a single linear combination* of the verifier's choosing. As we shall see, this can in fact be accomplished by adapting only the commitment step.

The idea behind this new protocol is simple, but has an important caveat. If the prover P wishes to commit to the messages $\boldsymbol{\alpha} = (\boldsymbol{\alpha}_1, \boldsymbol{\alpha}_2, \ldots, \boldsymbol{\alpha}_\ell)$, the obvious solution is to send $(y_i, \boldsymbol{\alpha}_i - y_{ik_i}, k_i)$ for all i, a sequence of commitments to each $\boldsymbol{\alpha}_i$. However, the indices k_i where each message is hidden are sampled independently, so that even though taking low-degree extensions is a linear operation (i.e., the LDE of $\sum_i \boldsymbol{\beta}_i y_i$ is $\sum \boldsymbol{\beta}_i \hat{y}_i$), a linear combination of the y_i does not yield a commitment to a linear combination of the $\boldsymbol{\alpha}_i$: evaluating it at k_i yields a sum where only the i^{th} summand is guaranteed to be correct.

We can fix this problem by hiding all the messages at the same coordinate k. Then, setting $\boldsymbol{\gamma} = \big(\boldsymbol{\alpha}_i - y_{ik} : i \in [\ell]\big)$ and $\gamma = \boldsymbol{\beta} \cdot \boldsymbol{\gamma} = \sum \boldsymbol{\beta}_i \boldsymbol{\gamma}_i$, we have

$$\gamma + \Big(\sum \boldsymbol{\beta}_i y_i\Big)_k = \sum \boldsymbol{\beta}_i (y_{ik} + \boldsymbol{\gamma}_i) = \boldsymbol{\alpha} \cdot \boldsymbol{\beta};$$

so a linear combination of commitments yields a commitment to a linear combination of the messages. Therefore, the prover may send $(y_1, \dots, y_\ell, \boldsymbol{\gamma}, k)$ and the new protocol will satisfy the binding property (a slightly stronger version of which, with respect to a random $\boldsymbol{\beta}$, will be necessary; we elaborate upon this later in the section).

More precisely, viewing $y \in \mathbb{F}^{\ell \times p}$ as a matrix whose i^{th} row is y_i, the prover may send y, say, column by column.[9] The resulting string, appended with $\boldsymbol{\gamma}$ and k, is a random INDEX instance whose alphabet is $\mathbb{F}^\ell$; and this enables us to show the hiding property for algebraic-commit as we did for commit.

The result is Protocol 6.4, which enables a prover to commit to multiple messages and decommit (via Protocol 6.3, using $\hat{y}(\boldsymbol{\rho}, \boldsymbol{\beta})$ as the fingerprint and $\boldsymbol{\beta} \cdot \boldsymbol{\gamma}$ as the correction) to an arbitrary linear combination of them.

Theorem 6.5. *Protocol 6.4 (algebraic-commit) and Protocol 6.3 (decommit) form a streaming commitment protocol with space complexity $s = O\big((\ell + m)\log q\big)$ if $p = q^{3\ell}$ and $dm = \operatorname{polylog}(q)$. The scheme is secure against $\operatorname{poly}(s)$-space adversaries and communicates $O(\ell q^{3\ell} \log q)$ bits.*

Furthermore, if each linear coefficient can be computed in $O(m \log q)$ space, then $s = O(m \log q)$.

We omit the proof, as it is a straightforward extension of Theorem 6.3.

Protocol 6.4: algebraic-commit$(\boldsymbol{\alpha})$

Input: explicit access to $p, m, d, q \in \mathbb{N}$ with $p \le d^m$, $q > d$ and $\mathbb{F} = \mathbb{F}_q$; as well as linear coefficients $\boldsymbol{\beta} \in \mathbb{F}^\ell$. Streaming access to $y \in \mathbb{F}^{\ell \times p}$ followed by $\boldsymbol{\gamma} \in \mathbb{F}^\ell$ and $k \in [p]$.

V: Sample $\boldsymbol{\rho} \sim \mathbb{F}^m$ and compute $\hat{y}(\boldsymbol{\rho}, \boldsymbol{\beta}) = \sum_{i=1}^{\ell} \boldsymbol{\beta}_i \hat{y}_i(\boldsymbol{\rho})$, a random linear fingerprint of y with coefficients $\boldsymbol{\beta}$, while streaming y.

Store $\boldsymbol{\rho}, k, \hat{y}(\boldsymbol{\rho}, \boldsymbol{\beta})$ and the correction $\gamma = \sum_{i=1}^{\ell} \boldsymbol{\beta}_i \gamma_i$.

We stress that the binding property of the linear commitment protocol has an important caveat: it is with respect to *the linear combination* $\boldsymbol{\alpha} \cdot \boldsymbol{\beta}$, rather than the entire tuple $\boldsymbol{\alpha}$. Therefore, if the prover has knowledge of the linear coefficients, it can easily commit to a set of messages $\boldsymbol{\alpha}' \neq \boldsymbol{\alpha}$ that nonetheless decommits to the same linear combination $\boldsymbol{\alpha} \cdot \boldsymbol{\beta}$, and P has many choices indeed: the equation $\sum \boldsymbol{\beta}_i \boldsymbol{\alpha}'_i = \sum \boldsymbol{\beta}_i \boldsymbol{\alpha}_i$ is satisfied by all $\boldsymbol{\beta}$ in the hyperplane (of size $q^{\ell-1}$) orthogonal to

[9] We remark that while sending y column by column naturally corresponds to an INDEX instance with a larger alphabet (where symbols are ℓ-tuples of field elements), since the hardness of INDEX holds for the stronger model of one-way communication protocols, the hiding property of the scheme is preserved regardless of the order in which y is sent. This is important in our sumcheck protocol, where a column cannot be sent all at once.

$\boldsymbol{\alpha}' - \boldsymbol{\alpha}$.

Since our applications require a stronger guarantee – that V should be able to detect when P commits to $\boldsymbol{\alpha}$ and a decommits according to $\boldsymbol{\alpha}' \neq \boldsymbol{\alpha}$ – this binding property is insufficient unless V chooses the coefficients $\boldsymbol{\beta}$ *at random*; then the linear combination of $\boldsymbol{\alpha}'$ matches that of $\boldsymbol{\alpha}$ only with probability $1/q$. While in our zero-knowledge protocol for PEP the coefficients are not *uniform*, they are a random evaluation of low-degree polynomials, and the same reasoning holds with a small loss (see Theorem 6.7).

However, an important issue still remains: the exponential dependency of Theorem 6.5 in the number ℓ of field elements that comprise the tuple P commits and decommits to. Concretely, in our applications we have $\ell = \omega(1)$ but can only afford to communicate $\mathrm{poly}(q)$ bits. To circumvent this issue, we shall use the following efficient reduction from INDEX over bits to the problem of distinguishing a commitment to a fixed element of $\mathbb{F}^\ell$ from a commitment to a random one.

Lemma 6.2. *Let (A, B) be a one-way protocol with s-bit messages that distinguishes between a length-p algebraic commitment to a fixed $\boldsymbol{\alpha} \in \mathbb{F}^\ell$ and a random commitment with advantage ε; that is, such that*

$$\left| \begin{array}{l} \mathbb{P}_{\substack{y\sim\mathbb{F}^{\ell\times p}\\ k\sim[p]}}\big[B\big(A(y), (\boldsymbol{\alpha}_i \oplus y_{ik} : i \in [\ell]), k\big)\ \textit{accepts}\big] \\ -\mathbb{P}_{\substack{y\sim\mathbb{F}^{\ell\times p}\\ k\sim[p]\\ \boldsymbol{\tau}\sim\mathbb{F}^\ell}}\big[B(A(y), \boldsymbol{\tau}, k)\ \textit{accepts}\big] \end{array} \right| = \varepsilon.$$

Then there exists an average-case one-way communication protocol for (binary) INDEX over p-bit strings that communicates $O(\ell^2 s \log^2 q/\varepsilon^2)$ bits and succeeds with probability $1 - \frac{1}{e} = \frac{1}{2} + \Omega(1)$.

Proof. Define, for ease of notation, $y^{(k)} := (y_{ik} : i \in [\ell])$ (i.e., the k^{th} column of y) and

$$a_{\boldsymbol{\tau}} := \mathbb{P}\left[B\left(A(y), \boldsymbol{\tau} \oplus y^{(k)}, k\right)\ \text{accepts}\right] = \mathbb{E}\left[B\left(A(y), \boldsymbol{\tau} \oplus y^{(k)}, k\right)\right],$$

where we interpret Bob's output as 1 (respectively 0) when he accepts (respectively rejects). Define, also, $\varepsilon_{\boldsymbol{\tau}} := a_{\boldsymbol{\alpha}} - a_{\boldsymbol{\tau}}$.

We first argue that, without loss of generality, we can assume $\mathbb{F} = \mathbb{F}_2 = \{0, 1\}$.

Note that, with $q = |\mathbb{F}|$,[10]

$$\varepsilon = a_{\boldsymbol{\alpha}} - \frac{1}{q^\ell} \sum_{\boldsymbol{\tau} \in \mathbb{F}^\ell} a_{\boldsymbol{\tau}} = \frac{1}{q^\ell} \sum_{\boldsymbol{\tau} \in \mathbb{F}^\ell} \varepsilon_{\boldsymbol{\tau}}.$$

Taking $\ell' := \lfloor \ell \log q \rfloor$ and $S \subseteq \mathbb{F}^\ell$ as the set of size $2^{\ell'}$ containing $\boldsymbol{\alpha}$ and the tuples $\boldsymbol{\tau}$ with the largest $\varepsilon_{\boldsymbol{\tau}}$, and viewing $\{0,1\}^{\ell'} \subseteq \mathbb{F}^\ell$ via a bijection between $\{0,1\}^{\ell'}$ and S, we have

$$\varepsilon' := \frac{1}{2^{\ell'}} \sum_{\boldsymbol{\tau} \in \{0,1\}^{\ell'}} \varepsilon_{\boldsymbol{\tau}} \geq \frac{\varepsilon}{3},$$

owing to $|S| \geq q^\ell/2$ and $\varepsilon_{\boldsymbol{\tau}} \geq \varepsilon_{\boldsymbol{\tau'}}$ when $\boldsymbol{\tau} \in S \setminus \{\boldsymbol{\alpha}\}$ and $\boldsymbol{\tau'} \in \mathbb{F}^\ell \setminus S$. Therefore, assuming $\mathbb{F} = \mathbb{F}_2$ incurs at most a constant factor in ε and a $\log q$ factor in ℓ; we shall use ε and ℓ (rather than ε' and ℓ') hereafter for simplicity of notation.

Finally, define, for each $0 \leq i < \ell$,

$$\varepsilon_i := \frac{1}{2^{\ell-i}} \sum_{\substack{\boldsymbol{\tau} \in \{0,1\}^\ell \\ \forall i' \leq i,\ \boldsymbol{\tau}_{i'} = \boldsymbol{\alpha}_{i'}}} \varepsilon_{\boldsymbol{\tau}}.$$

We divide the analysis into two cases: suppose, first, that $\varepsilon_i \geq \varepsilon_{i-1} \cdot \left(1 - \frac{1}{2\ell}\right)$ for all $i \in [\ell - 1]$. Then, by Bernoulli's inequality ($t \leq -1$ implies $(1+t)^\ell \geq 1 + t\ell$), we have

$$\varepsilon_{\ell-1} = \frac{1}{2}\left(a_{\boldsymbol{\alpha}} - a_{\boldsymbol{\alpha}^{\oplus \ell}}\right) \geq \left(1 - \frac{1}{2\ell}\right)^\ell \cdot \varepsilon \geq \frac{\varepsilon}{2},$$

where $\boldsymbol{\alpha}^{\oplus i} = (\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_{i-1}, 1 - \boldsymbol{\alpha}_i, \boldsymbol{\alpha}_{i+1}, \ldots, \boldsymbol{\alpha}_\ell)$. Consider the following one-way protocol (with shared randomness) for an INDEX instance $(x, j) \in \{0,1\}^p \times [p]$: Alice and Bob jointly sample $2/\varepsilon^2$ independent matrices $y' \sim \{0,1\}^{\ell \times p}$ and permutations $\sigma \sim S_p$; Alice sets $y_i = y'_i \oplus \mathbb{1}[i = \ell] \cdot \sigma(x)$ (where $\sigma(x)_k := x_{\sigma(k)}$), simulates $A(y)$ and sends the resulting messages in a $2s/\varepsilon^2$-bit string to Bob.

With knowledge of j, Bob finishes the simulations $B(A(y), \boldsymbol{\gamma}, k)$, using coordinate $k = \sigma^{-1}(j)$ and correction $\boldsymbol{\gamma} = \boldsymbol{\alpha} \oplus y'^{(k)}$; he computes their empirical mean μ, outputs $\boldsymbol{\alpha}_\ell$ if $\mu \geq a_{\boldsymbol{\alpha}} + \varepsilon/2$, and outputs $1 - \boldsymbol{\alpha}_\ell$ otherwise.

Correctness follows from the observation that, if $x_j = \sigma(x)_k = \boldsymbol{\alpha}_\ell$, then $\boldsymbol{\gamma} = \boldsymbol{\alpha} \oplus y^{(k)}$, so $\mathbb{E}[\mu] = a_{\boldsymbol{\alpha}}$; since the (y, k) pairs are uniform and independent,

$$\mathbb{P}\left[\mu \leq a_{\boldsymbol{\alpha}} - \frac{\varepsilon}{2}\right] \leq \frac{1}{e}$$

[10]This assumes the acceptance probability of a commitment to $\boldsymbol{\alpha}$ is larger than that of a random commitment, which is without loss of generality (otherwise Bob can simply flip his output bit).

by the Chernoff-Hoeffding bound (Lemma 2.1, with $2/\varepsilon^2$ samples and $\delta = \varepsilon/2$). Likewise, when $x_j = 1$ we have $\boldsymbol{\alpha} = \boldsymbol{\alpha}^{\oplus \ell} \oplus y^{(k)}$; then $\mathbb{E}[\mu] = a_{\boldsymbol{\alpha}^{\oplus \ell}} \leq a_{\boldsymbol{\alpha}} - \varepsilon$ and an application of the Chernoff-Hoeffding bound (with the same parameters) yields the same guarantee.

We now consider the second case: suppose $\varepsilon_i < \varepsilon_{i-1} \cdot \left(1 - \frac{1}{2\ell}\right)$ for some $i \in [\ell - 1]$; we take, without loss of generality, the minimal such i. Then

$$\begin{aligned}\frac{1}{2^{\ell-i}} \sum_{\substack{\boldsymbol{\tau} \in \{0,1\}^\ell \\ \forall i' \leq i,\ \boldsymbol{\tau}_{i'} = \boldsymbol{\alpha}_{i'}}} \varepsilon_{\boldsymbol{\tau}^{\oplus i}} &= \frac{1}{2^{\ell-i}} \sum_{\substack{\boldsymbol{\tau} \in \{0,1\}^\ell \\ \forall i' < i,\ \boldsymbol{\tau}_{i'} = \boldsymbol{\alpha}_{i'} \\ \boldsymbol{\tau}_i = 1 - \boldsymbol{\alpha}_i}} \varepsilon_{\boldsymbol{\tau}} \\ &= 2\varepsilon_{i-1} - \varepsilon_i \\ &> \varepsilon_{i-1}\left(1 + \frac{1}{2\ell}\right),\end{aligned}$$

and thus

$$\begin{aligned}\frac{1}{2^{\ell-i}} \left(\sum_{\substack{\boldsymbol{\tau} \in \{0,1\}^\ell \\ \forall i' \leq i,\ \boldsymbol{\tau}_{i'} = \boldsymbol{\alpha}_{i'}}} (\varepsilon_{\boldsymbol{\tau}^{\oplus i}} - \varepsilon_{\boldsymbol{\tau}}) \right) &= \frac{1}{2^{\ell-i}} \left(\sum_{\substack{\boldsymbol{\tau} \in \{0,1\}^\ell \\ \forall i' \leq i,\ \boldsymbol{\tau}_{i'} = \boldsymbol{\alpha}_{i'}}} (a_{\boldsymbol{\tau}^{\oplus i}} - a_{\boldsymbol{\tau}}) \right) \\ &> \frac{\varepsilon_{i-1}}{\ell} \\ &\geq \frac{\varepsilon}{2\ell}.\end{aligned}$$

We will use a similar strategy to the previous case, although the expression we must estimate involves many more terms (indeed, $2^{\ell-i+1}$ of them). Consider the following one-way protocol for an INDEX instance $(x, j) \in \{0,1\}^p \times [p]$: Alice and Bob jointly sample $64\ell^2/\varepsilon^2$ independent matrices $y' \sim \{0,1\}^{\ell \times p}$ and permutations $\sigma \sim S_p$; Alice sets $y_{i'} = y'_{i'} \oplus \mathbb{1}[i' = i] \cdot \sigma(x)$, computes and sends all messages $A(y)$ in a $64\ell^2 s/\varepsilon^2$-bit string to Bob.[11] (Recall that assuming $\mathbb{F} = \{0,1\}$ incurs constant and logarithmic factors in ε and ℓ, respectively, so that Alice's message is $O(\ell^2 s \log^2 q/\varepsilon^2)$ bits long.)

For each $A(y)$ sent by Alice, Bob simulates $B\left(A(y), \boldsymbol{\tau} \oplus y'^{(k)}, k\right)$ with $k = \sigma^{-1}(j)$ *for all* $\boldsymbol{\tau}$ satisfying $\boldsymbol{\tau}_{i'} = \boldsymbol{\alpha}_{i'}$ when $i' \leq i$. He computes the empirical mean μ

[11]Note that the only difference in Alice's strategy, as compared to the previous case, is the row where she inserts $\sigma(x)$ and the number of simulations of A.

of

$$\frac{1}{2^{\ell-i}}\left(\sum_{\substack{\boldsymbol{\tau}\in\{0,1\}^\ell\\ \forall i'\le i,\ \boldsymbol{\tau}_{i'}=\boldsymbol{\alpha}_{i'}}}\left(B\left(A(y),\boldsymbol{\tau}^{\oplus i}\oplus y'^{(k)},k\right)-B\left(A(y),\boldsymbol{\tau}\oplus y'^{(k)},k\right)\right)\right),$$

outputs 0 if the result is non-negative, and outputs 1 otherwise.

To prove correctness, first note that

$$\boldsymbol{\tau}\oplus y'^{(k)} = \begin{cases} \boldsymbol{\tau}\oplus y^{(k)}, & \text{when } x_j = 0\\ \boldsymbol{\tau}^{\oplus i}\oplus y^{(k)}, & \text{when } x_j = 1,\end{cases}$$

so that, when $x_j = 0$,

$$\mathbb{E}[\mu] = \frac{1}{2^{\ell-i}}\left(\sum_{\substack{\boldsymbol{\tau}\in\{0,1\}^\ell\\ \forall i'\le i,\ \boldsymbol{\tau}_{i'}=\boldsymbol{\alpha}_{i'}}}\left(a_{\boldsymbol{\tau}^{\oplus i}}-a_{\boldsymbol{\tau}}\right)\right) > \frac{\varepsilon}{2\ell},$$

and when $x_j = 1$ we have $\mathbb{E}[\mu] < -\varepsilon/2\ell$ (since the order of each pair of terms in the sum is flipped).

We conclude with an application of Hoeffding's inequality (Lemma 2.3, with $a=-1$, $b=1$, $\delta=1/2$ and $64\ell^2/\varepsilon^2$ samples): in the $x_j=0$ case,

$$\mathbb{P}\left[\mu \le \frac{\varepsilon}{4\ell}\right] \le \frac{1}{e};$$

and, likewise, in the $x_j = 1$ case we have $\mathbb{P}\left[\mu \ge -\frac{\varepsilon}{4\ell}\right] \le \frac{1}{e}$. □

6.2.4 A verifier-to-prover temporal commitment

The goal of this section is to construct the second main component towards our streaming zero-knowledge protocols. While it is not formally a commitment protocol (as per Definition 6.5), it is useful to conceptualise it as V committing to its internal randomness *before* the input is streamed (hence *temporal*).

Roughly speaking, we would like to ensure that a malicious verifier cannot choose the point $\boldsymbol{\rho}$ at which it (allegedly) computes its fingerprint *after it sees the input* $(x,\boldsymbol{\beta})$, as that would allow it to learn more than $f^x(\boldsymbol{\beta})$. (For example, in the INDEX case it could claim that $\boldsymbol{\rho} = j+1$ and learn $\hat{x}(j+1) = x_{j+1}$.) We will prove, in 3 steps, a lemma formalising the intuition that a space-s algorithm cannot remember the positions of significantly more than s elements, which will later enable the construction of a simulator. As in the case of algebraic commitments, we will in fact prove a stronger statement: that this holds not only in the case of streaming algorithms, but in the stronger model of one-way communication protocols.

We first define two variants of SEARCH-INDEX in the one-way communication complexity model, which we call RECONSTRUCT and PAIR (see Definitions 6.8 and 6.9). In RECONSTRUCT, Bob's task is to output the symbols at *every* coordinate of the input z (rather than receiving a single coordinate j and outputting only z_j, as in INDEX); in other words, Bob should reconstruct the input as best he can. In PAIR, as in SEARCH-INDEX, Bob's task is again to output the symbol at a single coordinate; but rather than receiving the index as part of the input, Bob is free to choose a coordinate-symbol pair (i, α) and succeeds if $\alpha = z_i$. (Note that in both RECONSTRUCT and PAIR, Bob does not receive any additional input besides Alice's message.)

Our first two steps are as follows. We first study RECONSTRUCT and show, in Lemma 6.3, that if Alice's message has s bits, Bob cannot reconstruct significantly more than s coordinates of the input. Then, in Lemma 6.4, we show how this bound for RECONSTRUCT implies a related bound for PAIR; more precisely, we prove that there exists a size-s set C of coordinates such that the probability Bob outputs a correct coordinate-symbol pair (i, z_i) where $i \notin C$ is arbitrarily small.

While Lemma 6.4 immediately implies an analogous statement for streaming algorithms, it is not yet enough for our purposes. The reason is that our verifier will read additional information, i.e., a fixed – but unknown – PEP instance $(x, \boldsymbol{\beta})$ between reading a PAIR input and writing its output. While it is intuitively clear that this should not help the verifier in any way (as the PEP and PAIR instances are uncorrelated), we still require a slight extension of Lemma 6.4.

To this end we define, for each fixed string $x \in \Gamma^n$, a variant of PAIR that we call PAIR(x) (Definition 6.10). The only difference between this one-way communication problem and PAIR is that Bob receives the string x in addition to Alice's message a. In Theorem 6.6, we show that the existence of a set capturing most of the correct outputs of PAIR implies such a set C also exists for PAIR(x); crucially, C is determined by a and *does not depend on* x. This last result then immediately implies an analogous one for streaming algorithms.

Let us begin with the definitions:

Definition 6.8. RECONSTRUCT is the following one-way communication problem: Alice receives a string $z \sim \Gamma^v$ and sends Bob an s-bit message a; after receiving a, Bob outputs a string $b \in \Gamma^v$. The *score* of an execution is the number of matching coordinates between z and b, i.e., $|\{i \in [v] : b_i = z_i\}|$.

Definition 6.9. *Let* PAIR denote the following one-way communication problem: Alice receives a string $z \sim \Gamma^v$ and sends Bob an s-bit message a; after receiving a, Bob outputs a pair $(\alpha, i) \in \Gamma \times [v]$. The execution succeeds if $\alpha = z_i$.

Note that both are definitionally average-case problems, as z is sampled uniformly. We now proceed to the first step towards the goal of this section: a proof that, in our parameter settings of interest for $|\Gamma|$ and s (as functions of v), the expected score of any protocol for RECONSTRUCT is tightly constrained by the message length s.

Lemma 6.3. *Any one-way protocol for* RECONSTRUCT *with alphabet size* $|\Gamma| = O(v/\log\log v)$, $|\Gamma| \geq 32v/\log\log v$ *and message length* s, *where* $\log v \leq s = \mathrm{polylog}(v)$, *achieves an expected score of at most* $s + o(s)$.

Proof. By the minimax theorem, we may assume Alice's and Bob's strategies are both deterministic, so that there exists a set of messages $A \subseteq \{0,1\}^s$ that partitions the set Γ^v of input strings by $\{P_a : a \in A\}$, where Bob outputs $b = b(a) \in \Gamma^v$ whenever $z \in P_a$.

Observe that Bob's optimal strategy is to set b_i as the most frequent symbol at the i^{th} coordinate among the strings of P_a; we can thus index the partition by $b \in B := \{b(a) : a \in A\}$, setting $P_b = P_{b(a)} = P_a$. (Note that while $\{P_b : b \in B\}$ may be a smaller partition than $\{P_a : a \in A\}$, the expected scores of the protocols induced by both partitions are the same.)

Define the random variable $M_b := \{i \in [v] : z_i = b_i\}$. For simplicity of notation, denote also $\gamma := |\Gamma|$. Note that the expected score of this one-way protocol is

$$\begin{aligned}
&\mathbb{E}_{z\sim\Gamma^v}\left[\sum_{b\in B} \mathbb{1}[z \in P_b] \cdot |M_b|\right] \\
&\quad= \sum_{b\in B} \mathbb{P}[z \in P_b] \cdot \mathbb{E}_{z\sim P_b}\left[|M_b|\right] \\
&\quad= \sum_{\substack{b\in B \\ |P_b| \geq \frac{s}{v}\cdot\frac{\gamma^v}{2^s}}} \mathbb{P}[z \in P_b] \cdot \mathbb{E}_{z\sim P_b}\left[|M_b|\right] + \sum_{\substack{b\in B \\ |P_b| < \frac{s}{v}\cdot\frac{\gamma^v}{2^s}}} \mathbb{P}[z \in P_b] \cdot \mathbb{E}_{z\sim P_b}\left[|M_b|\right].
\end{aligned}$$

We bound the first term by the largest expectation, and the second by observing that the union of sets P_b with $|P_b| \leq \frac{s}{v} \cdot \frac{\gamma^v}{2^s}$ contain at most an s/v fraction of all length-v strings:

$$
\begin{aligned}
\mathbb{E}_{z\sim\Gamma^v}\left[\sum_{b\in B}\mathbb{1}[z\in P_b]\cdot|M_b|\right] &\le \max_{\substack{b\in B\\ |P_b|\ge\frac{s\gamma^v}{v\cdot 2^s}}}\mathbb{E}_{z\sim P_b}\left[|M_b|\right]+\sum_{\substack{b\in B\\ |P_b|<\frac{s\gamma^v}{v\cdot 2^s}}}\mathbb{P}[z\in P_b]\cdot v\\
&\le \max_{\substack{b\in B\\ |P_b|\ge\frac{s\gamma^v}{v\cdot 2^s}}}\mathbb{E}_{z\sim P_b}\left[|M_b|\right]+s.
\end{aligned}
$$

Let $\delta\in(0,1)$ be such that the volume of Hamming balls of radius δ is $\mathcal{V}:=\frac{s\gamma^v}{v\cdot 2^s}\le\frac{s\gamma^v}{v^2}$. (Recall that $s\ge\log v$.) For any $b\in B$, the set P_b that maximises

$$
E_{z\sim P_b}\left[|M_b|\right]=|P_b|^{-1}\sum_{z\in P_b}|\{i\in[v]:z_i=b_i\}|
$$

is $P_b=B(b,\delta')$, the ball centered at b (whose radius δ' is determined by the equality $|B(b,\delta')|=|P_b|$). Since $|P_b|\ge\mathcal{V}$ implies $\delta'\ge\delta$, we have

$$
\frac{1}{|P_b|}\cdot\sum_{z\in B(b,\delta')}|\{i\in[v]:z_i=b_i\}|\le\frac{1}{\mathcal{V}}\cdot\sum_{z\in B(b,\delta)}|\{i\in[v]:z_i=b_i\}|,
$$

so it suffices to bound the right-hand side. (The inequality follows from the observation that the left-hand side is a weighted average between the right-hand side and the expectation over $z\sim B(b,\delta')\setminus B(b,\delta)$, which is smaller.)

Define $\varepsilon:=1-\delta$. We aim to upper bound $E_{z\sim P_b}\left[|M_b|\right]$, and set as an intermediate goal to prove upper and lower bounds for ε. To this end, we will use the following standard approximations (see, e.g., [GRS12]) for $H=H_2$ when σ (or $1-\sigma$) is small:

$$
H(\sigma)=H(1-\sigma)\in\left[\sigma\log\frac{1}{\sigma},\sigma\left(\log\frac{1}{\sigma}+\frac{2}{\ln 2}\right)\right] \tag{6.2}
$$

We begin with the lower bound on ε, which uses the lower bound of Eq. 6.2 and follows by showing that the volume of a ball with radius $1-\frac{\log\gamma}{v\log\log\gamma}$ is larger than $\mathcal{V}$; then $\delta<1-\frac{\log\gamma}{v\log\log\gamma}$, or, equivalently, $\varepsilon=1-\delta>\frac{\log\gamma}{v\log\log\gamma}$.

We have

$$\begin{aligned}
&H_\gamma\left(1-\frac{\log\gamma}{v\log\log\gamma}\right)\\
&=\frac{\left(1-\frac{\log\gamma}{v\log\log\gamma}\right)\log(\gamma-1)+H\left(1-\frac{\log\gamma}{v\log\log\gamma}\right)}{\log\gamma} && \text{(Eq. 2.1)}\\
&=\frac{\left(1-\frac{\log\gamma}{v\log\log\gamma}\right)\log(\gamma-1)+H\left(\frac{\log\gamma}{v\log\log\gamma}\right)}{\log\gamma} && \text{(Eq. 2.2)}\\
&\geq\frac{\left(1-\frac{\log\gamma}{v\log\log\gamma}\right)\left(\log\gamma+\log\left(1-\frac{1}{\gamma}\right)\right)}{\log\gamma}+\frac{\log\left(\frac{v\log\log\gamma}{\log\gamma}\right)}{v\log\log\gamma} && \text{(Eq. 6.2)}\\
&=1+\left(\frac{1}{\log\gamma}-\frac{1}{v\log\log\gamma}\right)\log\left(1-\frac{1}{\gamma}\right)+\frac{\log\frac{v}{\gamma}+\log\log\log\gamma}{v\log\log\gamma}-\frac{1}{v}\\
&\geq 1-\frac{\gamma+1}{\gamma^2\ln 2}\left(\frac{1}{\log\gamma}-\frac{1}{v\log\log\gamma}\right)+\frac{\log\frac{v}{\gamma}+\log\log\log\gamma}{v\log\log\gamma}-\frac{1}{v} && \text{(Eq. 2.3)}\\
&\geq 1-\frac{1}{\gamma\ln 2}\left(1+\frac{1}{\gamma}\right)\left(\frac{1}{\log\gamma}-\frac{1}{v\log\log\gamma}\right)-\frac{1}{v}\\
&\geq 1-\frac{3}{2v},
\end{aligned}$$

where the second-to-last inequality uses $v\geq\gamma$; and the last uses $\gamma=\Theta\left(\frac{v}{\log\log v}\right)$ to bound the first negative term to order $\Theta\left(\frac{\log\log v}{v\log v}\right)$, so the $1/v$ term dominates. Therefore,

$$\gamma^{H_\gamma\left(1-\frac{\log\gamma}{v\log\log\gamma}\right)v}\geq\gamma^v/\gamma^{3/2},$$

and thus, by Eq. 2.4, the volume of a ball (centered at any point b) of radius $1-\frac{\log\gamma}{v\log\log\gamma}=1-\frac{\mathrm{polylog}(v)}{v}$ satisfies

$$\begin{aligned}
\left|B\left(b,1-\frac{\log\gamma}{v\log\log\gamma}\right)\right| &\geq\frac{\gamma^{H_\gamma\left(1-\frac{\log\gamma}{v\log\log\gamma}\right)v}}{\sqrt{\log v}}\\
&\geq\frac{\gamma^v}{2^{\frac{3}{2}\log\gamma+\frac{1}{2}\log\log v}}\\
&\geq\frac{\gamma^v}{2^{\frac{7}{4}\log\gamma}}.
\end{aligned}$$

Then

$$\begin{aligned}|B(b,\delta)| = \mathcal{V} &= \frac{s\gamma^v}{v\cdot 2^s}\\ &\le \frac{\gamma^v \operatorname{polylog}(v)}{v^2}\\ &\le \frac{\gamma^v}{2^{\frac{15}{8}\log v}}\\ &\le \left|B\left(b, 1-\frac{\log\gamma}{v\log\log\gamma}\right)\right|,\end{aligned}$$

and we conclude that $\varepsilon = 1-\delta > \frac{\log\gamma}{v\log\log\gamma}$.

We now proceed to the upper bound on ε, which will use the upper bound of Eq. 6.2. Since $\gamma^{H_\gamma(\delta)v} \ge \mathcal{V} = \frac{s\gamma^v}{v\cdot 2^s}$ (Eq. 2.4), taking the logarithm of both sides and using Eq. 2.1 yields

$$\frac{(1-\varepsilon)\log(\gamma-1)+H(1-\varepsilon)}{\log\gamma} = H_\gamma(1-\varepsilon) = H_\gamma(\delta) \ge 1 - \frac{s+\log\frac{v}{s}}{v\log\gamma}. \qquad (6.3)$$

Note that the right-hand side is $1-o(1)$ because $s=o(v)$; then, δ is within $o(1)$ distance of the maximiser $1-1/\gamma = 1-o(1)$ of H_γ, so that $\delta = 1-o(1)$ and $\varepsilon = o(1)$.

This allows us to bound $H(\varepsilon) = H(1-\varepsilon)$ from above via Eq. 6.2, which, combined with Eq. 6.3 (multiplied by $\log\gamma$), implies

$$(1-\varepsilon)\log(\gamma-1)+\varepsilon\log\frac{1}{\varepsilon}+\frac{2\varepsilon}{\ln 2} \ge \log\gamma - \frac{s+\log v-\log s}{v}.$$

Rearranging yields

$$\varepsilon\left(\log\varepsilon+\log\gamma+\log\left(1-\frac{1}{\gamma}\right)-\frac{2}{\ln 2}\right) \le \frac{s+\log v-\log s}{v}+\log\left(1-\frac{1}{\gamma}\right).$$

The bounds $-\log(1-1/\gamma) = O(1/\gamma) = O(\log\log v/v)$ (Eq. 2.3) and $s \ge \log v$ show that the right-hand side is $O(s/v)$; and Eq. 2.3 along with $\log\gamma = \log v - \log\log\log v + \Theta(1) = \big(1-o(1)\big)\log v$ implies the left-hand side is $\Omega\big(\varepsilon(\log\varepsilon+\log v)\big)$. Therefore, the inequality above simplifies to

$$\varepsilon(\log\varepsilon+\log v) = O\left(\frac{s}{v}\right).$$

Now, if we had $\varepsilon = \Omega(s/v)$, then

$$\begin{aligned}\varepsilon(\log\varepsilon + \log v) &= \varepsilon\big(\log s - \log v + \log v + \Omega(1)\big)\\ &= \Omega(\varepsilon\log s) = \omega(s/v),\end{aligned}$$

a contradiction. We thus conclude that $\varepsilon = o(s/v)$ (and, in particular, that ε is both lower and upper bounded by $\operatorname{polylog}(v)/v$).

Returning to the goal of bounding the expected score, we now show that most of the volume of a Hamming ball of radius δ is close to its boundary. More precisely, consider the volume $\mathcal{V}'$ of a ball of radius $\delta' = 1 - 2\varepsilon$. As $\varepsilon = v^{-1}\operatorname{polylog}(v)$, Eq. 2.4 applies, giving $\mathcal{V}' \le \gamma^{H_\gamma(1-2\varepsilon)}$ and

$$\mathcal{V} = \Omega\left(\frac{\gamma^{H_\gamma(1-\varepsilon)}}{\sqrt{\varepsilon v}}\right) = \Omega\left(\frac{\gamma^{H_\gamma(1-\varepsilon)}}{\sqrt{s}}\right).$$

so that

$$\frac{\mathcal{V}'}{\mathcal{V}} = O\left(\sqrt{s}\cdot\gamma^{-(H_\gamma(1-\varepsilon)-H_\gamma(1-2\varepsilon))v}\right).$$

We can bound the coefficient in the exponent as follows:

$$\begin{aligned}H_\gamma(1-\varepsilon) - H_\gamma(1-2\varepsilon) &= \frac{\varepsilon\log(\gamma-1) + H(\varepsilon) - H(2\varepsilon)}{\log\gamma}\\ &\ge \frac{\varepsilon}{\log\gamma}\left(\log(\gamma-1) + \log\frac{1}{\varepsilon} - 2\log\frac{1}{2\varepsilon} - \frac{4}{\ln 2}\right) && \text{(by Eq. 6.2)}\\ &= \frac{\varepsilon}{\log\gamma}\left(\log(\varepsilon\gamma) + \log\left(1-\frac{1}{\gamma}\right) + 2 - \frac{4}{\ln 2}\right)\\ &\ge \frac{\varepsilon\log\log\gamma}{\log\gamma},\end{aligned}$$

where the last inequality follows from $\varepsilon\gamma > \frac{\gamma\log\gamma}{v\log\log\gamma} = \Theta\left(\frac{\log\gamma}{\log^2\log\gamma}\right)$ when the constant in $\Theta(\cdot)$ is large enough ($\gamma \ge 32v/\log\log v$ suffices, as $\log(1-1/\gamma)+2-4/\ln 2 > -5$).

Therefore,

$$\begin{aligned}
\sqrt{s} \cdot \gamma^{-(H_\gamma(1-\varepsilon) - H_\gamma(1-2\varepsilon))v} &\le \sqrt{s} \cdot \gamma^{-\frac{\varepsilon v \log\log\gamma}{\log\gamma}} \\
&= \sqrt{s} \cdot 2^{-\varepsilon v \log\log\gamma} \\
&< \sqrt{s} \cdot 2^{-\log\gamma} \\
&= \frac{\sqrt{s}}{\gamma} \\
&= \Theta\left(\frac{\sqrt{s}\log\log v}{v}\right) \\
&= o(s/v),
\end{aligned}$$

where the last line is due to $\sqrt{s} \ge \sqrt{\log v} = \omega(\log\log v)$ and the strict inequality to $\varepsilon > \frac{\log\gamma}{v\log\log\gamma}$. Therefore, $\mathcal{V}'/\mathcal{V} = o(s/v)$, showing that the volume of a ball of radius $1-\varepsilon$ is indeed concentrated in points of distance at least $1-2\varepsilon$.

Finally, we conclude that

$$\begin{aligned}
\mathbb{E}_{z\sim\Gamma^v}\left[\sum_{b\in B} \mathbb{1}[z \in P_b] \cdot |M_b|\right] &\le \max_{\substack{b\in B \\ |P_b| \ge \frac{s\gamma^v}{v\cdot 2^s}}} \mathbb{E}_{z\sim P_b}\left[|M_b|\right] + s \\
&\le s + \frac{1}{\mathcal{V}} \cdot \sum_{z\in B(b,\delta)} |\{i \in [v] : z_i = b_i\}| \\
&\le s + \frac{\mathcal{V}'}{\mathcal{V}} \cdot v + \left(1 - \frac{\mathcal{V}'}{\mathcal{V}}\right) \cdot 2\varepsilon v \\
&= s + o(s),
\end{aligned}$$

as desired. □

At this stage, we have an upper bound on the expected score of any one-way communication protocol for RECONSTRUCT. The next step is to show that it implies a similar bound for the communication problem PAIR; indeed, it seems intuitively clear that RECONSTRUCT is no harder than PAIR, as it allows Bob to output an independent guess for each coordinate. We formalise this intuition in the following lemma.

Lemma 6.4. *Any one-way protocol for* PAIR with alphabet size $\frac{32v}{\log\log v} \le |\Gamma| = O\left(\frac{v}{\log\log v}\right)$ and message length s, where $\log v \le s = \text{polylog}(v)$, satisfies the following: there exists an event E (depending only on z) with $\mathbb{P}[E] = 1 - o(1)$ and a set C of size s (depending only on Alice's message) such that

$$\mathbb{P}\big[\text{Bob outputs } (z_i, i) \text{ with } i \notin C \big| E\big] = o(1).$$

Proof. We will first show how to construct a protocol for RECONSTRUCT given one for PAIR, and then use Lemma 6.3 to conclude; as in that lemma, we define $\{P_a\}$ as the partition induced by Alice's messages $a = a(z) \in A$ (we can assume Alice to be deterministic, as before, by the minimax theorem; then a is a random variable determined by z).

Recall that in a protocol for PAIR, Bob's output is a random variable $b(a) \in \Gamma \times [v]$;[12] our goal is to construct, from this random variable, an entire string $y \in \Gamma^v$ and apply the expected score bound to it. For ease of notation, when the message a is fixed we write $b = (b_1, b_2) = b(a)$; note that b is independent of the conditional distribution $z \sim P_a$ of the input, since upon fixing a it is solely a function of Bob's internal randomness. We will denote its distribution by $\mu = \mu(a)$, and the conditional distribution of b_2 when $b_1 = i$ by μ_i.

The (PAIR) protocol's success probability, conditional on receiving a, is given by

$$\begin{aligned}\sum_{i=1}^{v} \mathbb{P}_{\substack{z\sim P_a\\ b\sim\mu}}[b = (z_i, i)] &= \sum_{i=1}^{v} \mathbb{P}_{b\sim\mu}[b_2 = i] \cdot \mathbb{P}_{\substack{z\sim P_a\\ b\sim\mu}}[b_1 = z_i \mid b_2 = i] \\ &= \sum_{i=1}^{v} \mathbb{P}_{b\sim\mu}[b_2 = i] \cdot \mathbb{P}_{\substack{z\sim P_a\\ b_1\sim\mu_i}}[b_1 = z_i].\end{aligned}$$

Define $y = y(a) \in \Gamma^v$ as the string whose i^{th} coordinate is the most frequent symbol at the i^{th} coordinate in P_a (as before, y is the best attempt at reconstructing the input z given to Alice). Now, consider the RECONSTRUCT protocol that outputs the string whose i^{th} coordinate is the random variable $b_1 \sim \mu_i$. Since, for each $i \in [v]$, the symbol $\alpha \in \Gamma$ maximising $\mathbb{P}_{z\sim P_a}[\alpha = z_i]$ is y_i, the expected score of the resulting protocol (conditioned on a) is

$$\begin{aligned}\sum_{i=1}^{v} \mathbb{P}_{z\sim P_a}[b_1 = z_i \mid b_2 = i] &= \sum_{i=1}^{v} \mathbb{P}_{\substack{z\sim P_a\\ b_1\sim\mu_i}}[b_1 = z_i] \\ &= \sum_{i=1}^{v} \sum_{\alpha\in\Gamma} \mathbb{P}_{b_1\sim\mu_i}[b_1 = \alpha] \cdot \mathbb{P}_{z\sim P_a}[\alpha = z_i] \\ &\le \sum_{i=1}^{v} \mathbb{P}_{z\sim P_a}[y_i = z_i] \\ &= \mathbb{E}_{z\sim P_a}\left[|M_a|\right],\end{aligned}$$

[12]Note that, in contrast with Alice, we cannot assume Bob is deterministic. We wish to bound the number of points in the support of b that aggregate all but a subconstant amount of probability weight in correct solutions to the problem. This is not a function of the *value* of b, but of its *distribution*, so the minimax principle does not apply.

where, as before, $M_a = \{i \in [v] : y_i = z_i\}$.

Recall that in Lemma 6.3 we showed that, as long as $|P_a| \geq \frac{s|\Gamma|^v}{v2^s}$, the above expectation is $o(s)$. We now conclude with the following claim, whose proof follows immediately afterwards.

Claim 6.1. *Let $p, q \in [0,1]^v$ be probability vectors and $t \leq v$ be an integer. There exists a set $C \subseteq [v]$ of size t such that $\sum_{i \in [v] \setminus C} p_i q_i \leq 1/t$.*

Note that while $r \in [0,1]^v$ defined by $r_i = \mathbb{P}[b_1 = z_i \mid b_2 = i]$ is not a probability vector, we may normalise it to obtain one: applying Claim 6.1 to $p = \big(\mathbb{P}[b_2 = i] : i \in [v]\big)$, $q = r/\|r\|_1$ and $t = s$, we obtain a set $C_a \subset [v]$ of size s such that

$$\begin{aligned}
\mathbb{P}_{\substack{z \sim P_a \\ b \sim \mu(a)}}\big[b = (z_i, i) \text{ with } i \notin C_a\big] &= \sum_{i \notin C_a}^{v} p_i r_i \\
&= \|r\|_1 \sum_{i \notin C_a}^{v} p_i q_i \\
&\leq \frac{\|r\|_1}{s} \\
&= \frac{\sum_{i=1}^{v} \mathbb{P}[b_1 = z_i \mid b_2 = i]}{s} \\
&= o(1)
\end{aligned}$$

whenever $|P_a| \geq \frac{s|\Gamma|^v}{v2^s}$. Finally, take C_a as given by the claim. Recall that the sets P_a of size less than $\frac{s|\Gamma|^v}{v2^s}$ cover at most a $s/v = o(1)$ fraction of length-v strings, so that the probability $z \sim \Gamma^v$ falls into the union of such sets is $o(1)$. In the complement of this event, we have

$$\begin{aligned}
&\mathbb{P}\left[b(a) = (z_i, i) \text{ with } i \notin C_a \;\middle|\; |P_a| \geq \frac{s|\Gamma|^v}{v2^s}\right] \\
&= \frac{1}{\mathbb{P}_{z \sim \Gamma^v}\left[|P_a| \geq \frac{s|\Gamma|^v}{v2^s}\right]} \sum_{\substack{a \in A \\ |P_a| \geq \frac{s|\Gamma|^v}{v2^s}}} \mathbb{P}_{z \sim \Gamma^v}[z \in P_a] \cdot \mathbb{P}_{\substack{z \sim P_a \\ b \sim \mu(a)}}[b = (z_i, i) \text{ with } i \notin C_a] \\
&= \frac{1}{1 - o(1)} \cdot o(1) = o(1),
\end{aligned}$$

which concludes the proof. □

The only step remaining to complete the proof of Lemma 6.4 is Claim 6.1, which we provide below.

Proof of Claim 6.1. We reduce the claim to proving an upper bound on a certain optimisation problem. Namely, let $\Delta = \{x \in [0,1]^v : \sum_i x_i = 1\}$ and $\Delta' =$

$\Delta \cap \{x \in [0,1]^v : x_1 \geq \cdots \geq x_v\}$ be the v-dimensional simplex and the simplex with ordered coordinates, respectively. Define the function $f : \Delta' \times \Delta \to \mathcal{R}_+$ by $f(p,q) = \sum_{i=1}^{v} i p_i q_i$.

Under the assumption that $f(p,q) \leq 1$ for all $p \in \Delta'$ and $q \in \Delta$, we conclude as follows: since $p_1 \geq p_2 \geq \cdots \geq p_v$ without loss of generality (permuting the vectors to satisfy the condition does not affect the truth of the claim), for any $t \in [v]$

$$1 \geq f(p,q) = \sum_{i=1}^{v} \left(\sum_{j=i}^{v} p_j q_j \right) \geq \sum_{i=1}^{t} \left(\sum_{j=i}^{v} p_j q_j \right)$$

implies the existence of $i \in [t]$ such that $\sum_{j=i}^{v} p_j q_j \leq 1/t$. Taking $C = [i-1]$ completes the proof.

We now proceed to show $f(p,q) \leq 1$. Since f is continuous with compact domain, there exists a pair (p^*, q^*) that maximises f. Let $\ell \in [v]$ be the largest nonzero coordinate of p^*. Then $q_i^* > 0$ for all $i \leq \ell$, as otherwise moving the mass p_i^* onto p_1^* would contradict maximality; and $q_i^* = 0$ for all $i > \ell$, or moving q_i^* onto (say) q_1^* likewise leads to a contradiction.

Now, suppose (towards contradiction) $\ell > 1$, take $1 < j \leq \ell$ and consider the pair (p^*, q') with $q'_1 = 0$, $q'_i = q_1^* + q_i^*$ and $q'_j = q_j^*$ otherwise. Then $f(p^*, q') \leq f(p^*, q^*)$ implies

$$ip_i^*(q_1^* + q_i^*) \leq p_1^* q_1^* + ip_i^* q_i^*,$$

and thus $ip_i^* \leq p_1^*$ (since $q_1^* \neq 0$). But then

$$f(p^*, q^*) = \sum_{i=1}^{\ell} ip_i^* q_i^* \leq p_1^* \sum_{i=1}^{\ell} q_i^* = p_1^* < 1,$$

a contradiction, as the delta distributions at 1 achieve value 1.

We thus conclude that $\ell = 1$, so the maximisers p^*, q^* are the delta distributions at 1 and $f(p,q) \leq f(p^*, q^*) = 1$, as desired. □

With the second step of our proof finished, we already have a nontrivial result by the known implication from hardness for one-way communication complexity: any streaming algorithm that streams a uniformly random string $z \in \Gamma^v$ *and immediately outputs* a pair (α, i) has a small set $C \subset [v]$ capturing most of the probability that it outputs correctly. However, the verifier in our zero-knowledge streaming protocol will stream an INDEX instance between streaming z and outputting a pair. To capture this behaviour, we define a (slight) variant of PAIR and prove that the result of Lemma 6.4 carries over to it.

Definition 6.10. *For each string $x \in \Gamma^n$, let* $\textsc{pair}(x)$ *denote the following one-way communication problem: Alice receives a string $z \sim \Gamma^v$ and sends Bob an s-bit message a; Bob reads x and a and outputs a pair $(\alpha, i) \in \Gamma \times [v]$. The protocol succeeds if $\alpha = z_i$.*

We have now reached the end goal of this section:

Lemma 6.5. *Fix a (single) one-way communication protocol for* $\textsc{pair}(x)$ *for all $x \in \Gamma^n$ with alphabet size $32v/\log\log v \le |\Gamma| = O(v/\log\log v)$ and message length $\log v \le s = \operatorname{polylog}(v)$. Then, for any $x \in \Gamma^n$, there exists an event E (that depends only on z) with $\mathbb{P}[E] = 1 - o(1)$ and a set C of size s (that depends only on Alice's message) satisfying*

$$\mathbb{P}\big[b(a,x) = (z_i, i) \text{ with } i \notin C_a \big| E\big] = o(1).$$

Proof. We will make a small adaptation in one of the steps of Lemma 6.4 to show there is a size-s set C *independent of* x that captures most of the probability of Bob's correct outputs.

Following the notation of Lemma 6.4, $\{P_a\}$ is the partition induced by Alice's messages and Bob's output is a random variable $b(a(z), x) = b(a, x) \in \Gamma \times [v]$. We also denote the distribution of $b = b(a, x)$ by $\mu(a, x)$ and the conditional distribution of b_1 when $b_2 = i$ by $\mu_i(a, x)$.

For every x and a, the protocol's success probability conditioned on $z \in P_a$ is

$$\begin{aligned}\sum_{i=1}^{v} \mathbb{P}_{\substack{z\sim P_a \\ b\sim\mu(a,x)}}[b = (z_i, i)] &= \sum_{i=1}^{v} \mathbb{P}_{b\sim\mu(a,x)}[b_2 = i] \cdot \mathbb{P}_{\substack{z\sim P_a \\ b\sim\mu(a,x)}}[b_1 = z_i \mid b_2 = i] \\ &= \sum_{i=1}^{v} \mathbb{P}_{b\sim\mu(a,x)}[b_2 = i] \cdot \mathbb{P}_{\substack{z\sim P_a \\ b_1\sim\mu_i(a,x)}}[b_1 = z_i].\end{aligned}$$

With $y = y(a) \in \Gamma^v$ as the string whose i^{th} coordinate is the most frequent symbol at the i^{th} coordinate in P_a, we know that $\mathbb{P}_{z\sim P_a}[\alpha = z_i]$ is maximal when $\alpha = y_i$. This holds also if α is a random variable (independent from z), so that, in particular, with $r \in [0,1]^v$ defined by

$$r_i := \max_{x\in\Gamma^n} \left\{ \mathbb{P}_{\substack{z\sim P_a \\ b_1\sim\mu_i(a,x)}}[b_1 = z_i] \right\} \le \mathbb{P}_{z\sim P_a}[y_i = z_i]$$

we have $\|r\|_1 = o(s)$ when $|P_a|$ is sufficiently large. Defining $p \in [0,1]^v$ by $p_i = \mathbb{P}_{b\sim\mu(a,x)}[b_2 = i]$, $p' \in [0,1]^v$ by $q_i = r_i/\|r\|_1$ and using Claim 6.1, we obtain a set

$C_a \subset [v]$ of size s such that *for every* $x \in \Gamma^n$,

$$\mathbb{P}_{\substack{z\sim P_a\\ b\sim\mu(a,x)}}\left[b = (z_i, i) \text{ and } i \notin C_a\right] = \sum_{i=1}^{v} \mathbb{P}_{b\sim\mu(a,x)}[b_2 = i] \cdot \mathbb{P}_{\substack{z\sim P_a\\ b_1\sim\mu_i(a,x)}}[b_1 = z_i]$$
$$\leq \|r\|_1 \sum_{i\notin C_a}^{v} p_i q_i = o(1),$$

and we conclude with same calculation of Lemma 6.4. □

As an immediate corollary (by taking C to be a set of symbol-coordinate pairs, rather than only coordinates; and setting, say, $C = \varnothing$ in the complement of the event E), we have:

Theorem 6.6. *Let* Γ *be an alphabet of size* $32v/\log\log v \leq |\Gamma| = \Theta(v/\log\log v)$ *and fix* $x \in \Gamma^n$*. Let* $\widetilde{V}$ *be a streaming space-*s *algorithm with* $\log v \leq s = \mathrm{polylog}(v)$ *that streams* $z \sim \Gamma^v$ *followed by* x*, and outputs a pair* $(\alpha, i) \in \Gamma \times [v]$*.*

There exists a set $C \subset \Gamma \times [v]$ *of size* s*, determined by the snapshot of* $\widetilde{V}$ *at the end of the stream* z*, such that*

$$\mathbb{P}\left[\widetilde{V}(z,x) \text{ outputs } (z_i, i) \notin C\right] = o(1).$$

The theorem above attains what we set out for in this section: since $\widetilde{V}$ cannot remember many pairs (z_i, i), we may prepend to any protocol a step where P sends z to the verifier. Then, whenever $\widetilde{V}$ sends an allegedly random $\alpha \in \Gamma$ to the prover, we ask that it *also* send the coordinate i such that $\alpha = z_i$ as evidence that α was indeed sampled in the past, i.e., before it finished streaming z. In other words, this step provides a *temporal commitment* by means of which $\widetilde{V}$ can show that its internal randomness is uncorrelated with the input.

6.3 A zero-knowledge SIP for polynomial evaluation

Our goal in this section will be to combine the components constructed in Sections 6.2.3 and 6.2.4 – *algebraic* and *temporal* commitment protocols – into a zero-knowledge protocol for PEP. It is useful to keep in mind that PEP is a generalisation of INDEX, and thus a protocol for the former yields one for the latter; in other words, for concreteness one may replace PEP by INDEX throughout this section. A formal definition of PEP follows.

Definition 6.11. *Let* $\alpha \in \mathbb{F}$ *and* $f = \{f^x : x \in \Gamma^n\}$ *be a mapping such that* $f^x : \mathbb{F}^m \to \mathbb{F}$ *is a degree-*d *polynomial.* PEP(f, α) *is the language* $\{(x, \boldsymbol{\beta}) \in \Gamma^n \times \mathbb{F}^m : f^x(\boldsymbol{\beta}) = \alpha\}$.

We remark that the parameters of the problem generally increase as a function of n; in particular, the field size is always assumed to satisfy $q = |\mathbb{F}| = \omega(1)$.

6.3.1 The protocol

For any mapping f and field element α, Protocol 6.5 lays out $\textsf{zk-pep}(f, \alpha)$, our zero-knowledge SIP for $\textsc{pep}(f, \alpha)$. Theorems 6.7 and 6.8 prove, respectively, the correctness (i.e., completeness and soundness) and the zero-knowledge properties of $\textsf{zk-pep}$.

The protocol uses commitment (sub)protocols to allow each party to only reveal key information after the other party gives evidence that it is being honest; this is achieved by interspersing the commit-decommit steps of one party with those of the other. More precisely, in the setup (Step 0) the verifier performs its (temporal) commitment; after the input is streamed (Step 1), the prover makes its (algebraic) commitment in Step 2. Then follow decommitments in the same order: verifier and prover decommit at Steps 3 and 4, respectively.

For ease of notation, we use $\mathbb{F}^\times$ to denote $\mathbb{F} \setminus \{0\}$, the multiplicative group of the field $\mathbb{F}$. Recall, moreover, that for a matrix y, we use $\hat{y}(\boldsymbol{\rho}, \boldsymbol{\theta}) \in \mathbb{F}$ to denote an evaluation of the low-degree extension of the string $\boldsymbol{\theta} \cdot y$ over $\mathbb{F}$ (see Section 2.5), and that $\boldsymbol{\chi}(\rho)$ denotes a vector of Lagrange polynomials (see Section 6.2.1); in the following protocol, the vector contains all but the first point of the interpolating set $\{0\} \cup [dm]$ for a univariate degree-dm polynomial over $\mathbb{F}$, i.e., $\boldsymbol{\chi}(\rho) = \big(\chi_i(\rho) : i \in [dm]\big) \in \mathbb{F}^{dm}$.

Protocol 6.5: $\textsf{zk-pep}(f, \alpha)$

Input: Explicit access to field $\mathbb{F}$, element $\alpha \in \mathbb{F}$, degree d, dimension m and a mapping $x \mapsto f^x$; streaming access to $x \in \Gamma^n$ followed by $\boldsymbol{\beta} \in \mathbb{F}^m$.

Parameters:

Field size $q = |\mathbb{F}|$ satisfying $dm = o(q)$;

Commitment lengths $v = q^m(\log m + \log\log q)/32$ and $p = m(dmq)^3$;

Step 0: Temporal commitment

$\boldsymbol{P}$: Send a string $z \sim \left(\mathbb{F}^m\right)^v$.

$\boldsymbol{V}$: Sample $\boldsymbol{\rho} \sim \mathbb{F}^m$ and stream z. For each $i \in [v]$, check if $z_i = \boldsymbol{\rho}$ and store $\ell = i$ if so.

Reject if $\boldsymbol{\rho} \neq z_i$ for all $i \in [v]$.

Step 1: Input streaming

$\boldsymbol{V}$: Stream x and compute $f^x(\boldsymbol{\rho}) \in \mathbb{F}$. Store $\boldsymbol{\beta} \in \mathbb{F}^m$.
If $\boldsymbol{\rho} = \boldsymbol{\beta}$, check that $f^x(\boldsymbol{\rho}) = \alpha$, accepting if so and rejecting otherwise.

Step 2: Algebraic commitment

$\boldsymbol{V}$: Sample $\rho \sim \mathbb{F}^\times \setminus [dm]$ and send the line $L : \mathbb{F} \to \mathbb{F}^m$ with $L(0) = \boldsymbol{\beta}$ and $L(\rho) = \boldsymbol{\rho}$.

$\boldsymbol{P}$: Send an algebraic commitment $(y, \boldsymbol{\gamma}, k)$ to $f^x_{|L}$, i.e., $(y, k) \sim \mathbb{F}^{dm \times p} \times [p]$ and $\boldsymbol{\gamma} \in \mathbb{F}^{dm}$ with $\boldsymbol{\gamma}_i = f^x_{|L}(i) - y_{ik}$ for all $i \in [dm]$.

$\boldsymbol{V}$: Sample $\boldsymbol{\sigma} \sim \mathbb{F}^m$ and, while streaming y, compute $\hat{y}\big(\boldsymbol{\sigma}, \boldsymbol{\chi}(\rho)\big)$.
Compute the correction $\gamma = \boldsymbol{\chi}(\rho) \cdot \boldsymbol{\gamma}$ and save (the identification of) $k \in \mathbb{F}^m$.

Step 3: Temporal decommitment

$\boldsymbol{V}$: Send $\boldsymbol{\rho}$ and ℓ.

$\boldsymbol{P}$: Check that $z_\ell = \boldsymbol{\rho} \in L$ and $\rho \coloneqq L^{-1}(\boldsymbol{\rho}) \notin \{0\} \cup [dm]$, aborting otherwise.

Step 4: Algebraic decommitment

$\boldsymbol{V}$: Run $\mathsf{decommit}\big(f^x(\boldsymbol{\rho}) - \chi_0(\rho)\alpha, \boldsymbol{\chi}(\rho) \cdot y, k\big)$, with correction γ and fingerprint $\hat{y}(\boldsymbol{\sigma}, \boldsymbol{\chi}(\rho))$. Accept if $\mathsf{decommit}$ accepts and reject otherwise.

6.3.2 Analysis of the protocol

We now show that $\mathsf{zk\text{-}pep}$ is a valid (i.e., complete and sound) streaming interactive proof, as well as compute its space and communication complexities.

Theorem 6.7. *Let f be such that an evaluation of the $\mathbb{F}_q$-polynomial f^x can be computed by streaming x in $O(m \log q)$ space. Then, for any $\alpha \in \mathbb{F}_q$, Protocol 6.5 is an SIP for* PEP(f, α) *with $s = O(m \log q)$ space complexity. Its communication complexity is $O(q^m m \log^2 q)$ in the setup and $O(d^4 m^5 q^3 \log q)$ in the interactive phase.*

Proof. We will prove completeness then soundness, and compute the complexities last.

Completeness. The verifier only aborts in Step 0 (the setup) if $\boldsymbol{\rho}$ is not among the $v > q^m \log \log q$ random tuples sent by the prover, an event with probability

$(1-1/q^m)^v \le e^{-v/q^m} = o(1)$. Otherwise, since the prover behaves honestly, in Step 2 (the algebraic commitment) we have

$$y_{ik} = f^x_{|L}(i) - \boldsymbol{\gamma}_i$$

for all $i \in [dm]$.

Let $w = \boldsymbol{\chi}(\rho) \cdot y = \sum_{i=1}^{dm} \chi_i(\rho) y_i \in \mathbb{F}^p$ and $\hat{w} : \mathbb{F}^m \to \mathbb{F}$ be its m-variate LDE. Recall that, in $\mathsf{decommit}\big(f^x(\boldsymbol{\rho}) - \chi_0(\rho)\alpha, w, k\big)$ (Protocol 6.3), with correction γ and fingerprint $\hat{y}(\boldsymbol{\sigma}, \boldsymbol{\chi}(\rho))$, the verifier sends a line $L' : \mathbb{F} \to \mathbb{F}^m$ with $L'(0) = k$, $L'(\sigma) = \boldsymbol{\sigma}$, receives $\hat{w}_{|L'}$ and makes two checks: that $\hat{w}_{|L'}(\sigma)$ matches the fingerprint and that $\hat{w}(0) + \gamma = f^x(\boldsymbol{\rho}) - \chi_0(\rho)\alpha$. Since

$$\hat{w}_{|L'}(\sigma) = \hat{w}(\boldsymbol{\sigma}) = \sum_{i=1}^{dm} \chi_i(\rho)\hat{y}_i(\boldsymbol{\sigma}) = \hat{y}\big(\boldsymbol{\sigma}, \boldsymbol{\chi}(\rho)\big)$$

and

$$\begin{aligned}
\hat{w}(0) + \gamma = w_k + \gamma &= \sum_{i=1}^{dm} \chi_i(\rho)\big(y_{ik} + \boldsymbol{\gamma}_i\big) \\
&= \sum_{i=1}^{dm} \chi_i(\rho) f^x_{|L}(i) \\
&= f^x(\boldsymbol{\rho}) - \chi_0(\rho) f^x(\boldsymbol{\beta}) \\
&= f^x(\boldsymbol{\rho}) - \chi_0(\rho)\alpha,
\end{aligned}$$

the verifier accepts when P is honest except with probability $o(1)$.

Soundness. First, note that if $\boldsymbol{\rho} \notin \{z_i : i \in [v]\}$, the verifier rejects already in Step 0. We can thus assume the tuple $\boldsymbol{\rho}$ equals some coordinate in z, and, since the string and tuple are independent random variables, the distribution of $\boldsymbol{\rho}$ is still uniform conditioned on this event. (We may also assume that $\boldsymbol{\rho} \neq \boldsymbol{\beta}$, since otherwise V also rejects regardless of the prover's behaviour.)

The only other point where V may reject is Step 4 (the algebraic decommitment). Once again, recall that V sends the prover a line L' with $L'(0) = k$, $L'(\sigma) = \boldsymbol{\sigma}$ where $\sigma \sim \mathbb{F}$ and P replies with a degree-dm polynomial $g : \mathbb{F} \to \mathbb{F}$ that is allegedly $\hat{w}_{|L'}$. The verifier then checks that $g(\sigma) = \hat{y}\big(\boldsymbol{\sigma}, \boldsymbol{\chi}(\rho)\big) = \hat{w}_{|L'}(\sigma)$ and $g(0) + \gamma = f^x(\boldsymbol{\rho}) - \chi_0(\rho)\alpha$, rejecting if either equality fails to hold.

We now analyse three cases: first, suppose that $g = \hat{w}_{|L'}$. Then the first check passes but

$$\begin{aligned} g(0) + \gamma &= w_k + \gamma \\ &= f^x(\boldsymbol{\rho}) - \chi_0(\rho) f^x(\boldsymbol{\beta}) \\ &\neq f^x(\boldsymbol{\rho}) - \chi_0(\rho)\alpha, \end{aligned}$$

so the verifier rejects (with probability 1).

Suppose, now, that $g(0) \neq \hat{w}_{|L'}(0)$. Then Lemma 6.1 (Schwartz-Zippel) implies $g(\sigma) \neq \hat{w}_{|L'}(\sigma)$, so the verifier rejects, except with probability $dm/q = o(1)$.

Finally, suppose that $g \neq \hat{w}_{|L'}$ but $g(0) = \hat{w}(0) = \sum_{i=1}^{dm} \chi_i(\rho) y_{ik}$. Then either the first check fails, i.e., $g(\sigma) \neq \hat{y}\big(\boldsymbol{\sigma}, \boldsymbol{\chi}(\rho)\big)$, and V rejects; or $g(\sigma) = \hat{y}\big(\boldsymbol{\sigma}, \boldsymbol{\chi}(\rho)\big)$, and the second check passes if

$$g(0) + \gamma = \sum_{i=1}^{dm} \chi_i(\rho)\big(y_{ik} + \boldsymbol{\gamma}_i\big)$$

is equal to

$$f^x(\boldsymbol{\rho}) - \chi_0(\rho)\alpha = \chi_0(\rho)\big(f^x(\boldsymbol{\beta}) - \alpha\big) + \sum_{i=1}^{dm} \chi_i(\rho) f^x_{|L}(i).$$

Rearranging, the second check corresponds to the following equation:

$$\chi_0(\rho)\big(f^x(\boldsymbol{\beta}) - \alpha\big) + \sum_{i=1}^{dm} \chi_i(\rho) \left(f^x_{|L}(i) - \boldsymbol{\gamma}_i - y_{ik}\right) = 0.$$

Now, consider the left-hand side of the equation as a polynomial in ρ: plugging in 0 for the variable ρ evaluates to $f^x(\boldsymbol{\beta}) - \alpha \neq 0$, so that it is a nonzero polynomial; and, crucially, ρ was sampled uniformly (from $\mathbb{F}^\times \setminus [dm]$) and independently of the communication (in particular, of y and $\boldsymbol{\gamma}$) by V. By Lemma 6.1 lemma once again, the equation is satisfied with probability at most $dm/(q - dm - 1) = o(1)$ and soundness follows.

Communication complexity. Most of the communication occurs in Steps 0 and 2 (the commitments), which communicate

$$O\big(q^m(\log m + \log\log q) m \log q\big) = O\left(q^m m \log^2 q\right) \qquad \text{and}$$
$$O\left(pdm \log q\right) = O\left(d^4 m^5 q^3 \log q\right)$$

bits, respectively. (The communication in other steps is significantly smaller: Step 1 has none, while Steps 3 and 4 communicate $m \log q + \log v = O(m \log q)$ and $O(dm \log q)$ bits, respectively.)

Space complexity. Apart from a constant number of elements of $\mathbb{F}$ (requiring $O(\log q)$ bits), the verifier stores $\ell \in [v]$, $k \in [p]$ and $\boldsymbol{\rho}, \boldsymbol{\sigma} \in \mathbb{F}^m$. Since $v \geq p$, the space complexity is dominated by ℓ and $\boldsymbol{\rho}, \boldsymbol{\sigma}$. Since storing ℓ requires $\log v = O(m \log q)$ bits (as does computing $f^x(\boldsymbol{\rho})$) while $\boldsymbol{\rho}$ and $\boldsymbol{\sigma}$ require $m \log q$ bits each, the space complexity follows. □

6.3.3 Zero-knowledge

Having shown that zk-pep is a valid streaming interactive proof, we now show it is also zero-knowledge.

Theorem 6.8. *Protocol 6.5 is zero-knowledge, secure against distinguishers with space $dm^2 \operatorname{polylog}(q)$. The simulator runs in $O\big((d+m \log q)m \log q\big) = O\big(dm^2 \log^2 q\big)$ space.*

Proof. Recall that a space-s SIP is zero-knowledge against $dm^2 \operatorname{polylog}(q)$-space distinguishers if there exists a streaming simulator S that satisfies the following. For any space-s (honest or malicious) verifier $\widetilde{V}$ and input $(x, \boldsymbol{\beta})$ where $f^x(\boldsymbol{\beta}) = \alpha$, given whitebox access to $\widetilde{V}$ the simulator S produces a view that is indistinguishable to a $dm^2 \operatorname{polylog}(q)$-space (streaming) algorithm from the view generated by an interaction of $\widetilde{V}$ with the honest prover. Note that $\widetilde{V}$ can be simulated in space $O(s)$, so the space complexity of the statement suffices to simulate the verifier of Protocol 6.5 since $s = O(m \log q)$.

The simulator interprets its read-only random bit string as (z, y) with $z \sim \mathbb{F}^v$ and $y \sim \mathbb{F}^{dm \times p}$ (so that $vm \log q + pdm \log q \leq q^{m+8}$ bits suffice and an algorithm with $(m+8) \log q$ space can address into this string). This pair will be used to simulate prover messages, whereas the simulation of $\widetilde{V}$ will use a source of randomness that cannot be reread (but has unbounded length). In the description that follows, as well as the more succinct one in Algorithm 6.1, recall that $\widetilde{V}$ is assumed to only output a decision at the end of the protocol (so that, if it decides to reject in the middle, it continues the protocol with dummy messages); and likewise if S (or P) aborts.

In the setup, Step 0 (the temporal commitment), S simulates $\widetilde{V}(z)$. Then, using the snapshot of the verifier's memory and its whitebox access to $\widetilde{V}$, the simulator finds the set C of s elements of $\mathbb{F}^m$ that $\widetilde{V}$ may successfully decommit to with the largest probabilities. More precisely, S calls the whitebox oracle $\mathcal{W}$ (see Definition 6.4) on the algorithm that corresponds to the verifier immediately before streaming x, with initial memory state equal to the current snapshot $b \in \{0,1\}^s$, and whose output is a pair $(\boldsymbol{\rho}, \ell)$ at Step 3 (ignoring L, the intermediate output at Step 2).

S initialises a sorted list of message-probability pairs in $\mathbb{F}^m \times [v] \times [0,1]$, and, for all $\ell \in [v]$, uses its oracle access to both z and $\mathcal{W}$ to find $\mu_\ell := \mathcal{W}\big(b, (z_\ell, \ell)\big)$. If the size of the list is smaller than s, or μ_ℓ is larger than the smallest probability in it, S adds (z_ℓ, ℓ, μ_ℓ) to it (and removes the tuple with the smallest $\mu_{\ell'}$ if the size of the resulting would have exceeded s).

This yields the set $C \subset \mathbb{F}^m \times [v]$ with the s most likely correct decommitments of $\widetilde{V}$. Since the string z is over the alphabet $\mathbb{F}^m$, whose size satisfies

$$\frac{v}{\log\log v} = \frac{q^m(\log m + \log\log q)}{32\log\big(m\log q + \log(\log m + \log\log q) - 5\big)} \leq \frac{q^m}{32},$$

$q^m = \Theta(v/\log\log v)$ as well as $s \geq \log p = \Theta(\log q)$ and $s = \mathrm{polylog}(p)$, Theorem 6.6 applies for this parameter setting. This ensures that, except with probability $o(1)$, the verifier $\widetilde{V}$ will output either $(z_\ell, \ell) \in C$ or an incorrect $(\boldsymbol{\rho}, \ell)$ with $z_\ell \neq \boldsymbol{\rho}$ in its decommitment at Step 3.

Then S proceeds to Step 1, simulating $\widetilde{V}(x)$ and, with $F := \{z_i : (z_i, i) \in C\}$, computing $f^x(\boldsymbol{\rho})$ *for every* $\boldsymbol{\rho} \in F$. At the start of Step 2 (the algebraic commitment), $\widetilde{V}$ sends a line L. The simulator inspects the intersection of L (viewed as a set) with the set of fingerprints F and computes a random degree-dm polynomial g subject to the constraints $g(\beta) = f^x_{|L}(\beta) = f^x\big(L(\beta)\big)$ for all $\beta \in L^{-1}(F)$.[13] Note that the description of g is comprised of $O(dm)$ field elements.

S samples $k \sim [p]$ then simulates $\widetilde{V}$ streaming y followed by $\boldsymbol{\gamma}_i = g(i) - y_{ik}$ for all $i \in [dm]$ and k; note that S is able to compute all $\boldsymbol{\gamma}_i$ from the description of g combined with its oracle access to y.

There is no prover-to-verifier communication in Step 3 (the temporal decommitment), so S simulates $\widetilde{V}$ until the verifier sends a tuple $\boldsymbol{\rho} \in \mathbb{F}^m$ and an index $\ell \in [v]$. The simulator then checks that $z_\ell = \boldsymbol{\rho} \in L$ and $\rho := L^{-1}(\boldsymbol{\rho}) \in \mathbb{F}^\times \setminus [dm]$; if not, then S aborts (as P would).

Finally, in Step 4 (the algebraic decommitment), S simulates $\widetilde{V}$ until it sends a line $L' : \mathbb{F} \to \mathbb{F}^m$. The only remaining part of the verifier's view left to generate are the evaluations of of the polynomial $\sum_{i\in[dm]} \chi_i(\rho)\hat{y}_i \circ L'$ for all points in $[dm]$. These are computed by S in a streaming fashion using its oracle access to y.

The space complexity of S is dominated by the description of the polynomial g, which requires $O(dm\log q)$ bits, and by the set C of $s = O(m\log q)$ elements of

[13] Knowledge of $f^x(\boldsymbol{\rho})$ for all $\boldsymbol{\rho} \in F$ enables the simulator to sample from this distribution: F fixes $|L \cap F|$ evaluations, and the simulator sets the $dm - |L \cap F|$ remaining ones uniformly.

$\mathbb{F}^m \times [v]$. Since each element requires $m \log q + \log v = O(m \log q)$ bits, the total space complexity is

$$O(dm \log q + sm \log q) = O\left((d + m \log q)m \log q\right) = O(dm^2 \log^2 q),$$

as claimed. (Apart from C, the simulator stores $f^x(\boldsymbol{\rho}) \in \mathbb{F}$ for every $\boldsymbol{\rho} \in C$, which requires $s \log q$ bits; and the lines L, L' as well as k, which require $O(\log q)$ bits each.)

Algorithm 6.1: Simulator for Protocol 6.5

Input: Whitebox access to $\widetilde{V}$; oracle access to a length-q^{m+8} random bit string interpreted as $(z, y) \in \left(\mathbb{F}^m\right)^v \times \mathbb{F}^{dm \times p}$; streaming access to $(x, \boldsymbol{\beta}) \in \Gamma^n \times \mathbb{F}^m$.

Output: View $\left(z, x, \boldsymbol{\beta}, y, \boldsymbol{\gamma}, k, (h(i) : i \in [dm])\right)$ with $k \in [p]$, $\boldsymbol{\gamma} \in \mathbb{F}^{dm}$ and $h : \mathbb{F} \to \mathbb{F}$.

Step 0: Temporal commitment

- $\boldsymbol{S}$: Send z.
- $\widetilde{\boldsymbol{V}}$: Simulate until the end of this step and let $b \in \{0,1\}^s$ be the resulting snapshot of $\widetilde{V}$. Use the whitebox oracle $\mathcal{W}$ to determine the set $C \subset \{(z_i, i) : i \in [v]\}$ of size s with the largest $\mathcal{W}(b, (z_i, i))$.

Step 1: Input streaming

- $\widetilde{\boldsymbol{V}}$: Stream x, computing and storing $f^x(\boldsymbol{\rho})$ for every $\boldsymbol{\rho} \in \{z_i : (z_i, i) \in C\}$ while simulating the verifier.
- $\boldsymbol{S}$: Store $\boldsymbol{\beta}$.

Step 2: Algebraic commitment

- $\widetilde{\boldsymbol{V}}$: Simulate until $\widetilde{V}$ sends a line L, aborting if $L(0) \neq \boldsymbol{\beta}$.
- $\boldsymbol{S}$: Sample a random polynomial $g : \mathbb{F} \to \mathbb{F}$ of degree at most dm subject to $g(0) = \alpha$ and $g(\beta) = f^x\big(L(\beta)\big)$ for all β such that $(i, L(\beta)) \in C$ for some $i \in [v]$.
  Send y followed by $\boldsymbol{\gamma} = \big(g(i) - y_{ik} : i \in [dm]\big)$ and $k \sim [p]$.
- $\widetilde{\boldsymbol{V}}$: Simulate until the end of the step.

Step 3: Temporal decommitment

$\boldsymbol{\widetilde{V}}$: Simulate until $\widetilde{V}$ sends $\boldsymbol{\rho} \in \mathbb{F}^m$ and $\ell \in [v]$.

$\boldsymbol{S}$: Check that $z_\ell = \boldsymbol{\rho} \in L$ and $\rho \in \mathbb{F}^\times \setminus [dm]$, aborting if either check fails or $(\boldsymbol{\rho}, \ell) \notin C$.

Step 4: Algebraic decommitment

$\boldsymbol{\widetilde{V}}$: Simulate until $\widetilde{V}$ sends a line $L' : \mathbb{F} \to \mathbb{F}^m$, aborting if $L'(0) \neq k$.

$\boldsymbol{S}$: Set $\rho \coloneqq L^{-1}(\boldsymbol{\rho})$ and send $\left(\sum_{i=1}^{dm} \chi_i(\rho) \cdot \hat{y}_i \circ L'(j) : j \in [dm]\right)$.

Now, all that remains is to prove indistinguishability by space-s' streaming algorithms between the output of S and a real transcript, for some s' comparable to the space complexities of the verifier and simulator. The following claim proves this with $s' = dm^2 \operatorname{polylog}(q)$ (which is larger than both).

Claim 6.2. *Fix* $\alpha \in \mathbb{F}$ *and* f *as in the definition of* PEP, an input $(x, \boldsymbol{\beta}) \in \mathbb{F}^n \times \mathbb{F}^m$, a bit string r of arbitrary length and a $O(m \log q)$-space verifier algorithm $\widetilde{V}$. Let D be a streaming algorithm with space $dm^2 \operatorname{polylog}(q)$ such that

$$\mathbb{P}\left[D\left(\mathrm{View}_{P,\widetilde{V}}(x,r)\right) \text{ accepts}\right] - \mathbb{P}\left[D\left(S\left(\widetilde{V}, x, r\right)\right) \text{ accepts}\right] = \varepsilon,$$

with $\mathrm{View}_{P,\widetilde{V}}(x,r)$ a view of Protocol 6.5 and $S\left(\widetilde{V}, x, r\right)$ output by Algorithm 6.1. Then $\varepsilon = o(1)$.

Assume, towards contradiction, that there exist α, f, an input $(x, \boldsymbol{\beta}) \in \mathbb{F}^n \times \mathbb{F}^m$, a streaming verifier $\widetilde{V}$ with $O(m \log q)$ space and a (streaming) distinguisher D with $dm^2 \operatorname{polylog}(q)$ space such that D distinguishes real transcripts of $\mathsf{zk\text{-}pep}(f, \alpha)$ from simulations with bias $\varepsilon = \Omega(1)$ when the input is $(x, \boldsymbol{\beta})$.

Recall that we assume that $\widetilde{V}$ rejects only after receiving all messages from P; therefore, the algebraic commitment $(y, \boldsymbol{\gamma}, k)$ is always present in both real and simulated views. Our goal is to show D implies a one-way protocol for INDEX over the binary alphabet with a small message and a large bias, using Lemma 6.2. We do so by constructing, from D, a one-way communication protocol that distinguishes algebraic commitments to a fixed message $\boldsymbol{\alpha} \in \mathbb{F}^\ell$ from algebraic commitments to a random $\boldsymbol{\alpha}' \in \mathbb{F}^\ell$, where $\ell \leq dm$.

As both the real and simulated transcripts are identically distributed up to (and including) the verifier's message in Step 2, the expected distinguishing advantage and probability of a simulation failure (i.e., of an abortion in Step 3 due to $(\boldsymbol{\rho}, \ell) \notin C$) are ε and $o(1)$, respectively (over z and the bits of the verifier randomness r used until then). Therefore, there exists a fixed prefix of the transcript that retains distinguishing advantage $\varepsilon/2$ and whose probability of a simulation failure is $o(1)$; indeed, at least an $\varepsilon/2$ fraction of prefixes retains advantage $\varepsilon/2$ and at most an $o(1)$ fraction yields simulation failures with $\Omega(1)$ probability, so an $\varepsilon/2 - o(1)$ fraction of prefixes work. We thus assume, in the one-way protocol we define next, not only x and $\boldsymbol{\beta}$ to be fixed, but also the line L and z – and, consequently, the set $C \subset \{(z_i, i) : i \in [v]\}$ (as well as the corresponding $f^x(z_i)$) that captures most of the weight of correct tuples $\widetilde{V}$ may decommit to, as given by Theorem 6.6.

Viewing L as the set of pairs $\{(L(\sigma), \sigma) : \sigma \in \mathbb{F}\}$, define $\ell := dm - |L \cap C|$ and assume,[14] without loss of generality, that $L \cap C = [dm] \setminus [\ell]$. Consider the following one-way communication protocol with shared randomness (for strings w of length p) that distinguishes a commitment $(w, k, \boldsymbol{\eta})$ to $\left(f^x(i) : i \in [\ell]\right)$ from a commitment to a random message: Alice uses S to simulate an interaction between P and $\widetilde{V}$ with input $(x, \boldsymbol{\beta})$ and verifier randomness r, executing D on the (partial and fixed) transcript thus obtained, until $\widetilde{V}$ sends a line $L : \mathbb{F} \to \mathbb{F}^m$ in Step 2.

Alice samples $\rho' \sim \mathbb{F}^\times \setminus [dm]$ and continues the simulation of D by feeding it $y \in \mathbb{F}^{dm \times p}$ defined as follows: $y_i := w_i$ for $i \in [\ell]$, $y_i \sim \mathbb{F}^p$ for $\ell < i < dm$ and

$$y_{dm} := \chi_{dm}(\rho')^{-1} \cdot \left(t - \sum_{i=1}^{dm-1} \chi_i(\rho') y_i \right),$$

where $y_{\ell+1}, \dots, y_{dm-1}$ and t are random strings (in $\mathbb{F}^p$) shared with Bob. Note that $\rho' \notin \{0\} \cup [dm]$ implies $\chi_{dm}(\rho') \neq 0$, so that y_{dm} is well-defined. (See Figure 6.1 for a diagram of the reduction.)

After simulating $\widetilde{V}$, D and S in Step 2 with y, she sends Bob all three snapshots as well as L and ρ' in a $dm^2 \operatorname{polylog}(q)$-bit message.[15] (The space complexities of $\widetilde{V}$ and S are both dominated by the distinguisher's.)

Bob, in turn, finishes the simulation of Step 2 with his (random) index $k \in [p]$ and the correction tuple $\boldsymbol{\gamma}$ defined as follows:[16]

$$\boldsymbol{\gamma}_i = \begin{cases} \boldsymbol{\eta}_i, & \text{if } i \leq \ell \\ \hat{x}(i) - y_{ik}, & \text{if } \ell < i < dm \end{cases}$$

[14]Note that when $|L \cap C| \geq dm$ the simulator knows the entirety of $f^x_{|L}$, in which case the distinguishing bias is 0. Nonzero bias thus implies $dm > |L \cap C|$.

[15]We assume Bob receives the tuple $\boldsymbol{\eta}$ and reads C along with the corresponding evaluations from the simulator's snapshot; alternatively, Alice could send this information in a message that is asymptotically no larger.

[16]Recall that all y_i for all $\ell < i < dm$ are contained in Alice and Bob's shared randomness.

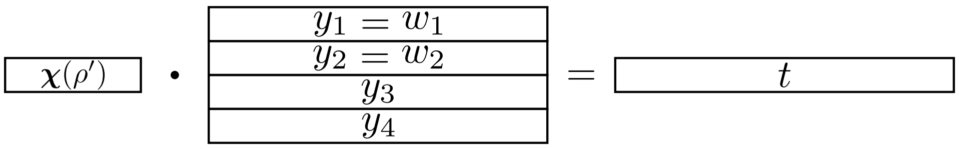


Figure 6.1: Reduction from INDEX to distinguishability of views when $\ell = 3$ and $dm = 4$. The instance w is inserted into the first 2 rows of y, while y_3 is filled in with joint randomness and y_4 is the solution of the linear system shown in the diagram.

and

$$\gamma_{dm} \coloneqq \chi_{dm}(\rho')^{-1} \left(f^x_{|L}(\rho') - \chi_0(\rho')\alpha - t_k - \sum_{i=1}^{dm-1} \chi_i(\rho')\gamma_i \right).$$

Bob proceeds to simulate Steps 3 and 4, using S to generate the remainder of the view. Note that in the former step $(\boldsymbol{\rho}, \ell) \notin C$ is the only case in which S aborts when P would not, which identifies a simulated transcript with certainty; but this is a small-probability event. When S fails (i.e., $(\boldsymbol{\rho}, \ell) \notin C$) or the field element $\rho = L^{-1}(\boldsymbol{\rho})$ is not equal to ρ', Bob halts the simulations and accepts or rejects uniformly; otherwise, he finishes the transcript by sending the low-degree polynomial that comprises the last round. This is possible because, while Bob does *not* know all $\hat{y}_i$, he does know the required linear combination:

$$\begin{aligned}\sum_{i=1}^{dm} \chi_i(\rho) \cdot y_i &= \sum_{i=1}^{dm-1} \chi_i(\rho) \cdot y_i + \chi_{dm}(\rho) \cdot \chi_{dm}(\rho)^{-1} \left(t - \sum_{i=1}^{dm-1} \chi_i(\rho) y_i \right) \\ &= t,\end{aligned}$$

and since t is a (random) string known to both Alice and Bob, in particular he can compute $\hat{t}_{L'}$ for any line $L' : \mathbb{F}^m \to \mathbb{F}$.

Finally, Bob inspects the output of D and chooses his output accordingly, accepting if and only if D accepts. Note that this one-way protocol succeeds

- with probability $1/2$ (and thus bias 0) either when S fails or when S succeeds and $\rho' \neq \rho$;
- with bias $\varepsilon/2$ when S succeeds and $\rho' = \rho$.

The latter follows from the fact that, if S succeeds and $\rho' = \rho$, it produces a full transcript where $\boldsymbol{\gamma}$ is a correction for the (unique) degree-dm polynomial g such that $g(0) = \alpha$, $g(i) = \boldsymbol{\eta}_i + y_{ik}$ for $i \in [\ell]$ and $g(i) = f^x(i)$ for $i \in [dm] \setminus [\ell] = L \cap C$.[17] Therefore, if $\boldsymbol{\eta} = \big(f^x(i) - y_{ik} : i \in [\ell]\big)$, then $\boldsymbol{\gamma}$ is a correction to f^x; while if $\boldsymbol{\eta}$ is random, then $\boldsymbol{\gamma}$ is a random degree-dm polynomial that matches f^x in (0 and) $L \cap C$.

[17]When $L \cap C \neq [dm] \setminus [\ell]$, the set still fixes $|L \cap C|$ values of g and leaves $dm - |L \cap C|$ to be chosen randomly.

Since D distinguishes between the two cases with bias $\varepsilon/2$, then so does the one-way protocol. Therefore,

$$\begin{aligned}
&\mathbb{P}_{\substack{w\sim\mathbb{F}^{\ell\times p}\\ k\sim[p]}}\left[B\left(A(w),\left(f^x(i)-w_{ik}:i\in[\ell]\right),k\right)\ \text{accepts}\right]\\
&\qquad-\mathbb{P}_{\substack{w\sim\mathbb{F}^{\ell\times p}\\ k\sim[p]\\ \boldsymbol{\eta}\sim\mathbb{F}^{\ell}}}\left[B\left(A(w),\boldsymbol{\eta},k\right)\ \text{accepts}\right]\\
&\qquad=o(1)\cdot\left(\frac{1}{2}-\frac{1}{2}\right)+\left(1-o(1)\right)\cdot\left(1-\frac{1}{q}\right)\cdot\left(\frac{1}{2}-\frac{1}{2}\right)+\left(1-o(1)\right)\cdot\frac{1}{q}\cdot\frac{\varepsilon}{2}\\
&\qquad\geq\frac{\varepsilon}{3q}\ .
\end{aligned}$$

Finally, applying Lemma 6.2, we obtain a one-way binary INDEX protocol for strings of length $p=m(dmq)^3$ with messages of length $\frac{dm^2q^2\ell^2\log^2 q}{\varepsilon^2}\,\mathrm{polylog}(q)\leq d^3m^4q^{2.01}$ and constant bias, a contradiction with $\sqrt{\frac{d^3m^4q^{2.01}}{p}}=o(1)$. □

6.3.4 Applications: INDEX, POINT-QUERY, RANGE-COUNT and SELECTION

From the general $\mathsf{zk\text{-}pep}(f,\alpha)$ protocol, we immediately obtain a zero-knowledge streaming interactive proof for the DECISION-INDEX(α) problem (Definition 6.7) as a corollary:

Corollary 6.1. *Fix $\delta\in(0,1]$. For any $\alpha\in\mathbb{F}_q$ where $q=\Theta\left(\log^{1+\frac{2}{\delta}}n\right)$, the language* DECISION-INDEX(α) *admits a zkSIP with space complexity $O(\log n)$ and communication complexities $O(n^{1+\delta})$ and* polylog(n) *in the setup and interactive stages, respectively. The protocol is secure against $\tilde{O}\left(\log^{2+\frac{2}{\delta}}n\right)$-space distinguishers.*

Proof. Set $d=\log^{\frac{2}{\delta}}n$ and $m=\delta\log n/2\log\log n$, so that $d^m=n$ and $dm/q=o(1)$. Note, moreover, that DECISION-INDEX(α) is the polynomial evaluation problem where $f_x=\hat{x}$, the low-degree extension of x (which can be computed in $O(m\log q)$ space) and $\boldsymbol{\beta}$ is the identification of a coordinate $j\in[n]$. Thus, applying Protocol 6.5 to the mapping $x\mapsto\hat{x}$ with the aforementioned parameters, we obtain a protocol with verifier space complexity

$$\begin{aligned}
O(m\log q)&=O\left(\frac{\log n}{\log\log n}\cdot\log\log n\right)\\
&=O(\log n)
\end{aligned}$$

and communication complexities

$$\begin{aligned}O(q^m m \log^2 q) &= \left(\log^{1+\frac{2}{\delta}} n\right)^{\frac{\delta \log n}{2 \log\log n}} \operatorname{polylog}(n)\\ &= n^{1+\frac{\delta}{2}} \operatorname{polylog}(n)\\ &= O(n^{1+\delta})\end{aligned}$$

in the setup and $O(d^4 m^5 q^3 \log q) = \operatorname{polylog}(n)$ in the interactive stage; moreover, it is secure against distinguishers with $dm^2 \operatorname{polylog}(q) = \tilde{O}\left(\log^{2+\frac{2}{\delta}} n\right)$ space. $\square$

Remark 6.9. Inspecting the proof of Claim 6.2, we see that increasing the prover's commitment length allows us to achieve significantly stronger indistinguishability: with $p = \log^{\omega(1)} n$, we have $\sqrt{s' q^2 \ell^2 / p} = o(1)$ for any $s' = \operatorname{polylog}(n)$. This setting of p increases only the communication complexity of the interactive phase (Steps 2 to 4) – which can still be bounded by $n^{o(1)}$ – and makes the protocol secure against $\operatorname{polylog}(n)$-space distinguishers.

We now select a few applications of the zk-pep protocol to solve other streaming problems; the remainder of this section follows reductions to pep due to [CCM+19].

In the POINT-QUERY problem, the input is a stream of updates $(u, i) \in \mathbb{Z} \times [\ell]$ to an ℓ-dimensional vector y initialised to zero, followed by an index j, and the task is to output y_j. A formal definition follows.[18]

Definition 6.12. *Let $\ell, M \in \mathbb{N}$ and $t \in [-M, M] \cap \mathbb{Z}$. The language* POINT-QUERY(t) *is defined as*

$$\left\{ (u_1, k_1, \ldots, u_n, k_n, j) : \begin{array}{c} \forall i, u_i \in [-M, M] \cap \mathbb{Z} \text{ and } k_i, j \in [\ell], \\ \forall k, \left|\sum_{i \in [n], k_i = k} u_i\right| \leq M \text{ and} \\ \sum_{i \in [n], k_i = j} u_i = t \end{array} \right\}.$$

Corollary 6.2. *Fix $\delta \in (0, 1]$. Let $\ell, M \in \mathbb{N}$ with $\ell \in [n]$, $M = \operatorname{poly}(n)$ and $t \in [-M, M] \cap \mathbb{Z}$. There exists a zkSIP for* POINT-QUERY(t) *with space complexity $O(\log^2 n)$ and communication complexities $O(n^{1+\delta})$ and $\operatorname{polylog}(n)$ in the setup and interactive stages, respectively.*

Proof. We first note that, by an application of the Chinese Remainder Theorem (see, e.g., [GR15]), we may assume $M = O(\log n)$ at the cost of a logarithmic blowup to the space complexity: the verifier runs Protocol 6.5 in parallel with $O(\log n)$ fields $\mathbb{F}_q \supset \mathbb{F}_p$ for distinct primes $p = O(\log n)$, so that any integer in $[-M, M]$ can be

[18]We remark that POINT-QUERY is formally a promise problem: the condition that coordinatewise sums are bounded by M is assumed to hold for no-instances of the language too. However, a polynomial bound is often trivially true (as in the applications that follow).

uniquely represented by logarithmically many field elements.

We set the same parameters as in Corollary 6.1: degree $d = \log^{\frac{2}{\delta}} n$ and $m = \delta \log n / 2 \log\log n$, but also ensure $q = \Theta\left(\log^{1+\frac{2}{\delta}} n\right)$ is the power of a prime larger than $2M + 1$ (so that elements of $[-M, M] \cap \mathbb{Z}$ map to distinct field elements).

Viewing integers in $[-M, M]$ as elements of $\mathbb{F}$, we define $y \in \mathbb{F}^\ell$ by

$$y_k := \sum_{\substack{i \in [n] \\ k_i = k}} u_i,$$

and the mapping $x = \big((u_i, k_i) : i \in [n]\big) \mapsto f^x$ by $f^x := \hat{y}$. Note that the verifier can compute

$$\hat{y}(\boldsymbol{\rho}) = \sum_{k \in [\ell]} \left(\sum_{\substack{i=1 \\ k_i = k}}^{n} u_i \right) \chi_k(\boldsymbol{\rho})$$

by recording the running sum of $u_i \chi_{k_i}(\boldsymbol{\rho})$, a task for which $O(m \log q) = O(\log n)$ space suffices.

Applying Protocol 6.5 with the mapping and parameters above, we obtain a zero-knowledge SIP with space complexity $O(\log^2 n)$ (due to the aforementioned logarithmic overhead), communication complexity $O(n^{1+\delta})$ in the setup and polylog(n) in the interactive stage. $\square$

With the protocol of Corollary 6.2, we obtain a zero-knowledge SIP for the RANGE-COUNT problem, where the stream consists of a sequence x of elements in a set $[\ell]$ followed by a subset $R \subseteq [\ell]$, and the task is to return the number of times an element of R appeared in the stream. Formally,

Definition 6.13. *Let $\mathcal{R} \subseteq 2^{[\ell]}$. The language* RANGE-COUNT(t) *is defined as*

$$\{(x, R) \in [\ell]^n \times \mathcal{R} : |\{i \in [n] : x_i \in R\}| = t\}.$$

Corollary 6.3. *Fix $\delta \in (0, 1]$. For every $\mathcal{R} \subseteq 2^{[\ell]}$ of size* poly(n)*, the language* RANGE-COUNT(t) *admits a zkSIP with space complexity $O(\log^2 n)$ and communication complexities $O(n^{1+\delta})$ and* polylog(n) *in the setup and interactive stages, respectively.*

Proof sketch. We run the protocol for POINT-QUERY (Corollary 6.2) on the stream obtained by concatenating $(R' \in \mathcal{R} : x_i \in R')$ for every $i \in [n]$ (which the verifier can simulate while streaming x), followed by R (viewed as an element of $[|\mathcal{R}|]$). More precisely, we redefine the mapping $x \mapsto f^x$ as what would be obtained by processing the derived stream, which avoids the length overhead (to $n|\mathcal{R}| = \text{poly}(n)$, rather than n) incurred otherwise.

Since $M = n$ is an upper bound for the number of points in any subset of $[\ell]$, we obtain a protocol with the complexities as claimed. □

We conclude with an application of the RANGE-COUNT protocol to solve SELECTION (and MEDIAN in particular). For $x \in [\ell]^n$ and $i \in [\ell]$, we call $\varphi(x)$ the *frequency vector* of x, defined as $\varphi_i(x) = |\{j \in [n] : x_j = i\}|$ (see, also, Definition 6.16). A word in the language SELECTION consists of x along with a *rank* $r \in [n]$ the integer $k \in [\ell]$ with this rank and offsets $\phi \in [n]$, $\phi' \in \{0\} \cup [n]$. (We remark that the additional parameters take into account what the verifier learns in the search version of the SIP: not only the element k with rank r, but the values of the cumulative frequencies up to $k-1$ and up to k.)

Definition 6.14. *For $\ell \in [n]$, the language* SELECTION *is defined as*

$$\left\{ (x,k,r,\phi,\phi') \in [\ell]^{n+1} \times [n]^2 \times \{0\} \cup [n] : \begin{array}{l} \sum_{i=1}^{k-1} \varphi_i(x) = r - \phi \textit{ and} \\ \sum_{i=1}^{k} \varphi_i(x) = r + \phi' \end{array} \right\}.$$

Corollary 6.4. *Fix $\delta \in (0,1]$. There exists a zkSIP for* SELECTION *with space complexity $O(\log^2 n)$ and communication complexities $O(n^{1+\delta})$ and* polylog(n) *in the setup and interactive stages, respectively.*

Proof sketch. We execute the protocol for RANGE-COUNT twice (by temporally committing and streaming x only once; this can be done by saving two independent fingerprints for f^x, and only running zk-pep twice from Step 2 onwards). The class of ranges is $\mathcal{R} = \{[n] \setminus [i] : 0 \leq i \leq n\}$, of size $O(n)$, and the verifier checks that the number of hits in the ranges $[n] \setminus [k-1]$ and $[n] \setminus [k]$ are $r - \phi$ and $r + \phi'$, respectively. □

6.4 A zero-knowledge sumcheck SIP

In the previous section we showed how Protocol 6.1, the polynomial evaluation protocol of [CTY11], can be made zero-knowledge with the careful addition of algebraic and temporal commitment protocols. Although PEP is a foundational problem for streaming algorithms – generalising INDEX, for example – it is not immediately clear whether the same techniques enable us to construct a zero-knowledge version of the second widely used tool in SIPs: the *sumcheck* protocol. In this section, we prove that they do: Protocol 6.7 leys out zk-sumcheck, a zkSIP for the SUMCHECK problem (Definition 6.15) with the same components, namely, the algebraic and temporal commitments that enabled zk-pep.

Sumcheck protocols are extremely useful building blocks for the construction of interactive proofs; indeed, some of the most celebrated results of the last two decades rely on them, most notably the GKR [GKR15] and subsequent delegation-of-computation protocols (e.g., [RVW13, RRR21, RR20b]). Roughly speaking, they allow a verifier to check that the sum, over a subcube, of the evaluations of a polynomial yields a prescribed field element; they save *exponentially* in the communication (and time) complexity as compared to sending the entire description of the polynomial. In particular, they enable the (exact) computation of frequency moments of a stream via an interactive protocol in sublinear space [CCMT14], which is impossible without interaction [AMS99].

More precisely, let $f : \mathbb{F}^m \to \mathbb{F}$ be a polynomial of (individual) degree d and $H \subset \mathbb{F}$ be an evaluation domain. One obvious way to check that $\sum_{\boldsymbol{\beta} \in H^m} f(\boldsymbol{\beta})$ is equal to some $\alpha \in \mathbb{F}$ is via the description of f (say, as a list of sufficiently many evaluations), from which the sum can be computed directly. This requires not only the entire description of f, which has size $(d+1)^m$; but also entails evaluating f over $|H|^m$ many points, implying an even larger runtime.

The standard sumcheck protocol (Protocol 6.6) enables a verifier V to offload this costly computation to a powerful prover P and check the claim by communicating $O(dm)$ field elements in $O(|H|md)$ time steps, with *a single random evaluation of* f.[19]

Protocol 6.6: $\mathsf{sumcheck}(f, \alpha)$

Input: Explicit access to $\mathbb{F} = \mathbb{F}_q$, evaluation domain $H \subset \mathbb{F}$, degree d, dimension m and $\alpha \in \mathbb{F}$ as well as $f(\boldsymbol{\rho})$ with $\boldsymbol{\rho} \sim \mathbb{F}^m$, where $f : \mathbb{F}^m \to \mathbb{F}$ is a degree-d polynomial.

Repeat, from $i = 1$ to m:

- ***P***: Send $f_i(T) = \sum_{\beta_{i+1},\dots,\beta_m \in H} f(\boldsymbol{\rho}_1, \dots, \boldsymbol{\rho}_{i-1}, T, \beta_{i+1}, \dots, \beta_m)$.
- ***V***: Send $\boldsymbol{\rho}_i$.

V: Check that $\sum_{\beta_1 \in H} f_1(\beta_1) = \alpha$, $f(\boldsymbol{\rho}) = f_m(\boldsymbol{\rho}_m)$ and the intermediate polyomials satisfy $\sum_{\beta_i \in H} f_i(\beta_i) = f_{i-1}(\boldsymbol{\rho}_{i-1})$ for all $2 \le i < m$, accepting if so and rejecting otherwise.

It is well known that the protocol above (always) accepts if $\sum_{\boldsymbol{\beta} \in H^m} f(\boldsymbol{\beta}) = \alpha$, and rejects with probability at least $1 - dm/q$ otherwise (see, e.g., [AB09]). As

[19] Protocol 6.6 is laid out in a somewhat non-standard (but equivalent) form, with checks deferred to the end, that more closely resembles the streaming version we construct.

sums of polynomials can be performed in a streaming fashion, the verifier only needs $O(m \log q)$ bits of space.

6.4.1 The protocol

We now show that the techniques of Section 6.2 enable us to construct a streaming zero-knowledge variant of $\mathsf{sumcheck}(f, \alpha)$, which solves the problem defined next.

Definition 6.15. *Let $\alpha \in \mathbb{F}$, $H \subseteq \mathbb{F}$ and $f = \{f^x : x \in \Gamma^n\}$ be a mapping such that $f^x : \mathbb{F}^m \to \mathbb{F}$ is a degree-d polynomial.* $\textsc{sumcheck}(f, \alpha)$ *is the language* $\left\{ x \in \Gamma^n : \sum_{\boldsymbol{\beta} \in H^m} f^x(\boldsymbol{\beta}) = \alpha \right\}$.

The techniques need to be adapted, however, with one key distinction between $\mathsf{zk\text{-}sumcheck}$ and $\mathsf{zk\text{-}pep}$: the prover now must make many (algebraic) commitments, each of which is used in a pair of decommitments; moreover, the commitments cannot be sent in parallel anymore, owing to dependencies between messages in contiguous rounds. Intuitively, neither of these should pose too great a challenge: computing fingerprints of a set of messages whose commitment is sent sequentially should be no easier than when they are sent in parallel (indeed, for one-way communication protocols they are exactly equivalent); and if one algebraic decommitment does not leak a significant amount of information, two should not do so either.

The protocol follows. We note that (differently from Section 6.3) $\boldsymbol{\chi}(\rho)$ denotes the vector of Lagrange polynomials over $\mathbb{F}$ for degree-d univariate polynomials with interpolating set $[d+1]$, i.e., $\boldsymbol{\chi}(\rho) = \big(\chi_i(\rho) : i \in [d+1]\big) \in \mathbb{F}^{d+1}$.

Protocol 6.7: $\mathsf{zk\text{-}sumcheck}(f, \alpha)$

Input: Explicit access to $\mathbb{F}$, element $\alpha \in \mathbb{F}$, degree d, dimension m, evaluation domain $H \subset \mathbb{F}$ and mapping $x \mapsto f^x$; streaming access to $x \in \Gamma^n$.

Parameters:

Field size $q = |\mathbb{F}|$ satisfying $dm = o(q)$;

Commitment lengths $v = q^m(\log m + \log\log q)/96$ and $p = q^{\log\log q}$.

Step 0: Temporal commitment

P: Send a string $z \sim \big((\mathbb{F} \setminus [d+1])^m\big)^v$.

V: Sample $\boldsymbol{\rho} \sim (\mathbb{F} \setminus [d+1])^m$ and stream z. Check if $z_i = \boldsymbol{\rho}$ for each i, storing $\ell = i$ if so.

Reject if $\boldsymbol{\rho} \neq z_i$ for all $i \in [v]$.

Step 1: Input streaming

$\boldsymbol{V}$: Stream x and compute $f^x(\boldsymbol{\rho}) \in \mathbb{F}$.

Step 2: Algebraic commitments

$\boldsymbol{P}$: Compute $f_1(T) = \sum_{\beta_2,\dots,\beta_m \in H} f^x(T, \beta_2 \dots, \beta_m)$ and sample $k \sim [p]$.

$\boldsymbol{V}$: Sample $\boldsymbol{\sigma}^{(1)}, \dots, \boldsymbol{\sigma}^{(m+1)} \sim \mathbb{F}^m$. Compute $\boldsymbol{\chi}(\boldsymbol{\rho}_1) \dots, \boldsymbol{\chi}(\boldsymbol{\rho}_m)$ and the linear coefficients $\boldsymbol{\theta}$ such that $\sum_{\beta \in H} g(\beta) = \sum_i \boldsymbol{\theta}_i g(i)$ when g is a degree-d univariate polynomial.

Repeat, from $i = 1$ to m:

$\boldsymbol{P}$: Send $y^{(i)} \sim \mathbb{F}^{(d+1)\times p}$ and $\boldsymbol{\gamma}^{(i)} = \big(f_i(j) - y^{(i)}_{jk} : j \in [d+1]\big)$.

$\boldsymbol{V}$: Compute the fingerprints $\hat{y}^{(i)}\left(\boldsymbol{\sigma}^{(i)}, \boldsymbol{\chi}(\boldsymbol{\rho}_i)\right)$ and $\hat{y}^{(i)}\left(\boldsymbol{\sigma}^{(i+1)}, \boldsymbol{\theta}\right)$, as well as the dot products $\boldsymbol{\chi}(\boldsymbol{\rho}_i) \cdot \boldsymbol{\gamma}^{(i)}$ and $\boldsymbol{\theta} \cdot \boldsymbol{\gamma}^{(i)}$.
Send $\boldsymbol{\rho}_i$.

$\boldsymbol{P}$: Compute $f_{i+1}(T) = \sum_{\beta_{i+2},\dots,\beta_m \in H} f^x(\boldsymbol{\rho}_1, \dots, \boldsymbol{\rho}_i, T, \beta_{i+2}, \dots, \beta_m)$ if $i < m$.

$\boldsymbol{P}$: Send k.

Step 3: Temporal decommitment

$\boldsymbol{V}$: Send ℓ.

$\boldsymbol{P}$: Check that $z_\ell = \boldsymbol{\rho} \in \big(\mathbb{F} \setminus [d+1]\big)^m$, aborting otherwise.

Step 4: Algebraic decommitments

$\boldsymbol{V}$: For all $1 < i \le m$, run

$$\mathsf{decommit}\left(0, \boldsymbol{\theta} \cdot y^{(i)} - \boldsymbol{\chi}(\boldsymbol{\rho}_{i-1}) \cdot y^{(i-1)}, k\right), \text{ with}$$

fingerprint $\hat{y}^{(i)}\left(\boldsymbol{\sigma}^{(i)}, \boldsymbol{\theta}\right) - \hat{y}^{(i-1)}\left(\boldsymbol{\sigma}^{(i)}, \boldsymbol{\chi}(\boldsymbol{\rho}_{i-1})\right)$ and correction $\boldsymbol{\theta} \cdot \boldsymbol{\gamma}^{(i)} - \boldsymbol{\chi}(\boldsymbol{\rho}_{i-1}) \cdot \boldsymbol{\gamma}^{(i-1)}$.

Run $\mathsf{decommit}\left(\alpha, \boldsymbol{\theta} \cdot y^{(1)}, k\right)$ with fingerprint $\hat{y}^{(1)}\left(\boldsymbol{\sigma}^{(1)}, \boldsymbol{\theta}\right)$ and correction $\boldsymbol{\theta} \cdot \boldsymbol{\gamma}^{(1)}$.

Run $\mathsf{decommit}\left(f^x(\boldsymbol{\rho}), \boldsymbol{\chi}(\boldsymbol{\rho}_m) \cdot y^{(m)}, k\right)$ using $\hat{y}^{(m)}\left(\boldsymbol{\sigma}^{(m+1)}, \boldsymbol{\chi}(\boldsymbol{\rho}_m)\right)$ as the fingerprint and $\boldsymbol{\chi}(\boldsymbol{\rho}_m) \cdot \boldsymbol{\gamma}^{(m)}$ as the correction.

Accept if all decommitments accept, and reject otherwise.

6.4.2 Analysis of the protocol

We now show that zk-sumcheck is a valid (i.e., complete and sound) streaming interactive proof, and compute its space and communication complexities.

Theorem 6.10. *Let f be such that an evaluation of the $\mathbb{F}_q$-polynomial f^x can be computed by streaming x in $O(m^2 \log q)$ space. For any $\alpha \in \mathbb{F}_q$, Protocol 6.7 is an SIP for* SUMCHECK(f, α) *with space complexity $s = O(m^2 \log q)$, communication complexity $O(q^m m \log^2 q)$ in the setup and $O(q^{\log\log q} dm \log q) = q^{\log\log q} \operatorname{poly}(q)$ in the interactive phase.*

Proof. As in Theorem 6.7, we first show completeness and soundness, then compute the complexities.

Completeness. Recall that decommit(β, w, k) with correction γ accepts if (the fingerprint matches the LDE of w and) $\gamma + w_k = \beta$. Therefore, when P and V are both honest, the first $m-1$ decommitments of Step 4 accept, since

$$
\begin{aligned}
&\boldsymbol{\theta} \cdot \boldsymbol{\gamma}^{(i)} - \boldsymbol{\chi}(\boldsymbol{\rho}_{i-1}) \cdot \boldsymbol{\gamma}^{(i-1)} + \left(\boldsymbol{\theta} \cdot y^{(i)} - \boldsymbol{\chi}(\boldsymbol{\rho}_{i-1}) \cdot y^{(i-1)}\right)_k \\
&\qquad = \sum_{j=1}^{d+1} \left(\boldsymbol{\theta}_j \big(\boldsymbol{\gamma}_j^{(i)} + y_{jk}^{(i)}\big) - \chi_j(\boldsymbol{\rho}_{i-1}) \big(\boldsymbol{\gamma}_j^{(i-1)} + y_{jk}^{(i-1)}\big)\right) \\
&\qquad = \sum_{j=1}^{d+1} \boldsymbol{\theta}_j f_i(j) - \sum_{j=1}^{d+1} \chi_j(\boldsymbol{\rho}_{i-1}) f_{i-1}(j) \\
&\qquad = \left(\sum_{\beta \in H} f_i(\beta)\right) - f_{i-1}(\boldsymbol{\rho}_{i-1}) \\
&\qquad = 0.
\end{aligned}
$$

Likewise, the last two decommitments accept because

$$\begin{aligned}
\boldsymbol{\theta} \cdot \boldsymbol{\gamma}^{(1)} + \left(\boldsymbol{\theta} \cdot y^{(1)}\right)_k &= \sum_{j=1}^{d+1} \boldsymbol{\theta}_j (\boldsymbol{\gamma}_j^{(1)} + y_{jk}^{(1)}) \\
&= \sum_{j=1}^{d+1} \boldsymbol{\theta}_j f_1(j) \\
&= \sum_{\beta \in H} f_1(\beta) \\
&= \sum_{\boldsymbol{\beta} \in H^m} f(\boldsymbol{\beta}) \\
&= \alpha
\end{aligned}$$

and

$$\begin{aligned}
\boldsymbol{\chi}(\boldsymbol{\rho}_m) \cdot \boldsymbol{\gamma}^{(m)} + \left(\boldsymbol{\chi}(\boldsymbol{\rho}_m) \cdot y^{(m)}\right)_k &= \sum_{j=1}^{d+1} \chi_j(\boldsymbol{\rho}_m)(\boldsymbol{\gamma}_j^{(m)} + y_{jk}^{(m)}) \\
&= \sum_{j=1}^{d+1} \chi_j(\boldsymbol{\rho}_m) f_m(j) \\
&= f_m(\boldsymbol{\rho}_m) \\
&= f^x(\boldsymbol{\rho}),
\end{aligned}$$

respectively. The verifier thus accepts unless $\boldsymbol{\rho} \neq \{z_i : i \in [v]\}$ in Step 0, an event with probability

$$\left(1 - \frac{1}{q-d-1}\right)^v \leq e^{-v/(q-d-1)^m} \leq e^{-v/q^m} = o(1).$$

Soundness. We divide the behaviour of a malicious prover into three cases. The first (and simplest) is when $\widetilde{P}$ commits to f_i for all i and decommits with polynomials whose evaluations at 0 yield the same values as the honest prover (i.e., in $\mathsf{decommit}(\beta, w, k)$ with γ as the correction, $\widetilde{P}$ replies with a polynomial g such that $g(0) = w_k + \gamma$). Then, since $\sum_{\boldsymbol{\beta} \in H^m} f(\boldsymbol{\beta}) \neq \alpha$, the verifier rejects in $\mathsf{decommit}\left(\alpha, \boldsymbol{\theta} \cdot y^{(1)}, k\right)$ with probability 1.

The second case is when $\widetilde{P}$ commits to a sequence of polynomials $g_1, \ldots, g_m$ such that $g_i \neq f_i$ for some i, and decommits honestly. Then V accepts if and only if the set $\{g_i\}$ leads the verifier in the standard sumcheck protocol to accept; by the soundness of that protocol, V accepts with probability at most $dm/(q-d-1) = o(1)$.

The only remaining case is when $\widetilde{P}$ commits to a sequence of polynomials $\{g_i\}$ (which may or may not coincide with $\{f_i\}$) and, in at least one decommitment with respect to a string w where $\widetilde{P}$ receives the line L, the prover replies with a degree-dm polynomial g such that $g(0) \neq w_k = \hat{w}_{|L}(0)$. Then, since V has a fingerprint $\hat{w}(\boldsymbol{\sigma})$ with $\boldsymbol{\sigma} \sim \mathbb{F}^m$ and a field element $\sigma \sim \mathbb{F}$ such that $L(\sigma) = \boldsymbol{\sigma}$, we have $g(\sigma) \neq \hat{w}(\boldsymbol{\sigma}) = \hat{w}_{|L}(\sigma)$ with probability $dm/q = o(1)$ by Lemma 6.1 (Schwartz-Zippel), and soundness follows.

Space and communication complexities. The communication of the setup (Step 0, the temporal commitment) is $q^m(\log m + \log\log q)m \log q = O(q^m m \log^2 q)$ bits. The communication of the interactive phase (Steps 2 to 4) is dominated by the m algebraic commitments to elements of $\mathbb{F}^{d+1}$ with length $p = q^{\log\log q}$ each, for a total of $O(q^{\log\log q} dm \log q) \leq q^{\log\log q + 2}$ bits.

The verifier's space complexity is dominated by computing $f^x(\boldsymbol{\rho})$ and storing $O(m)$ elements of $\mathbb{F}^m$ (i.e., $\boldsymbol{\rho}$ and $\boldsymbol{\sigma}^{(i)}$ for $i \in [m+1]$), so that it is bounded by $O(m^2 \log q)$. ☐

6.4.3 Zero-knowledge

Having shown that zk-sumcheck is a valid streaming interactive proof, we now show it is also zero-knowledge.

Theorem 6.11. *Protocol 6.7 is zero-knowledge against* poly(q)*-space streaming distinguishers. The simulator has space complexity* poly(q).

Proof. We shall prove indistinguishability as we have done earlier: with the simulator S shown in Algorithm 6.2, we assume towards contradiction that there exists $\alpha \in \mathbb{F}$, an input $x \in \mathbb{F}^n$, internal randomness r, a space-$O(m^2 \log q)$ verifier $\widetilde{V}$ and a poly(q)-space distinguisher D that accepts $\mathrm{View}_{P,\widetilde{V}}(x, r)$ with probability $\varepsilon = \Omega(1)$ above that with which D accepts $S\big(\widetilde{V}, x, r\big)$. Then, via Lemma 6.2, we construct a one-way protocol for INDEX with impossibly large success probability.

The space complexity of S is dominated by its storing of $O(m^2 \log q) = \mathrm{poly}(q)$ elements of $\mathbb{F}^m \times [v]$ and by the computation of the partial sums $(g_i : i \in [m])$. Note that the naive strategy of sampling g and computing the corresponding partial sums requires $\Omega(d^m)$ space; however, [BCF+17] constructs an algorithm that can sample from the same distribution in poly(q) time, and thus space.[20] Note, moreover, that the alphabet over which z is taken has size

[20]More precisely, the algorithm of [BCF+17] allows us to sample from the distributions $g_i(\beta)$ for any β and i under the uniform distribution of g satisfying a set of constraints. To sample $(g_1, \ldots, g_m)$, we begin with the set of constraints induced by C and, after sampling $g_i(j)$, include the corresponding constraint before the next sample.

$$\begin{aligned}(q-d-1)^m &= q^m\left(1-\frac{d+1}{q}\right)^m\\ &\geq q^m\left(1-\frac{1}{m}\right)^m\\ &\geq \frac{q^m}{3}\\ &\geq \frac{32v}{\log\log v},\end{aligned}$$

so that Theorem 6.6 applies. (The conditions $(q-d-1)^m = \Theta\left(\frac{v}{\log\log v}\right)$ and $\log q \leq s = \mathrm{polylog}(q)$ are also clearly satisfied.)

Algorithm 6.2: Simulator for Protocol 6.7

Input: Whitebox access to $\widetilde{V}$; oracle access to a random bit string of length $q^{m+\log\log q}\,\mathrm{poly}(q)$ interpreted as the concatenation of $z \in (\mathbb{F}^m)^v$ and $y^{(i)} \in \mathbb{F}^{(d+1)\times p}$ for all $i \in [m]$.

Output: View $\left(z, x, \left(y^{(i)}, \boldsymbol{\gamma}^{(i)} : i \in [m]\right), k, \left(h_i : i \in [m+1]\right)\right)$, where $z \in \left((\mathbb{F}\setminus[d+1])^d\right)^v$, $y^{(i)} \in \mathbb{F}^{(d+1)\times p}$, $\boldsymbol{\gamma}^{(i)} \in \mathbb{F}^{d+1}$, $k \in [p]$ and $h_i : \mathbb{F} \to \mathbb{F}$ has degree dm.

Step 0: Temporal commitment

- $\boldsymbol{S}$: Send $z \in \left((\mathbb{F}\setminus[d+1])^m\right)^v$.
- $\widetilde{\boldsymbol{V}}$: Simulate until the end of this step and let $b \in \{0,1\}^s$ be the resulting snapshot of $\widetilde{V}$. Use the whitebox oracle $\mathcal{W}$ to determine the set $C \subset \{(z_i, i) : i \in [v]\}$ of size s with the largest $\mathcal{W}(b,(z_i,i))$.

Step 1: Input streaming

- $\widetilde{\boldsymbol{V}}$: Stream x, simulating the verifier while computing and storing $f^x(z_i)$ for all $(z_i, i) \in C$.

Step 2: Algebraic commitments

- $\boldsymbol{S}$: Take $g_1 : \mathbb{F} \to \mathbb{F}$ of degree (at most) d under the distribution determined by sampling $g : \mathbb{F}^m \to \mathbb{F}$ subject to the constraints $\sum_{\boldsymbol{\beta}\in H^m} g(\boldsymbol{\beta}) = \alpha$ and $g(z_i) = f^x(z_i)$ for all $(z_i, i) \in C$, then outputting $g_1(T) = \sum_{\beta_2,\ldots,\beta_m \in H} g(T, \beta_2\ldots,\beta_m)$.
  Sample $k \sim [p]$.

$\widetilde{\boldsymbol{V}}$: Simulate until the end of the step.

Repeat, from $i = 1$ to m:

- $\boldsymbol{S}$: Send $y^{(i)}$ and $\boldsymbol{\gamma}^{(i)} = \left(g_i(j) - y^{(i)}_{jk} : j \in [d+1]\right)$.
- $\widetilde{\boldsymbol{V}}$: Simulate until $\boldsymbol{\rho}_i$ is sent (or until the end of the step when $i = m$).
- $\boldsymbol{S}$: If $i < m$, sample g_{i+1} under the distribution that samples g and outputting $g_{i+1}(T) = \sum_{\beta_{i+2},\ldots,\beta_m \in H} g(\boldsymbol{\rho}_1, \ldots, \boldsymbol{\rho}_i, T, \beta_{i+2}, \ldots, \beta_m)$.

$\boldsymbol{S}$: Compute $\boldsymbol{\theta}$ such that $\sum_{\beta \in H} h(\beta) = \sum_i \boldsymbol{\theta}_i h(i)$ when h is a degree-d univariate polynomial, and send k.

Step 3: Temporal decommitment

$\widetilde{\boldsymbol{V}}$: Simulate until $\widetilde{V}$ sends $\ell \in [v]$.

$\boldsymbol{S}$: Abort if $z_\ell \neq \boldsymbol{\rho}$, $\boldsymbol{\rho} \notin \left(\mathbb{F} \setminus [d+1]\right)^m$ or $(\boldsymbol{\rho}, \ell) \notin C$.

Step 4: Algebraic decommitments

For all $1 < i \leq m$,

- $\widetilde{\boldsymbol{V}}$: Simulate until $\widetilde{V}$ sends a line $L_i : \mathbb{F} \to \mathbb{F}^m$.
- $\boldsymbol{S}$: Abort if $L_i(0) \neq k$, and otherwise send

$$\left(\left(\boldsymbol{\theta} \cdot \hat{y}^{(i)} - \boldsymbol{\chi}(\boldsymbol{\rho}_i) \cdot \hat{y}^{(i-1)}\right) \circ L_i(j) : j \in [dm+1]\right).$$

$\widetilde{\boldsymbol{V}}$: Simulate until $\widetilde{V}$ sends a line $L_1 : \mathbb{F} \to \mathbb{F}^m$.

$\boldsymbol{S}$: Abort if $L_1(0) \neq k$, and otherwise send

$$\left(\left(\boldsymbol{\theta} \cdot \hat{y}^{(1)}\right) \circ L_1(j) : j \in [dm+1]\right).$$

$\widetilde{\boldsymbol{V}}$: Simulate until $\widetilde{V}$ sends a line $L_{m+1} : \mathbb{F} \to \mathbb{F}^m$.

$\boldsymbol{S}$: Abort if $L_{m+1}(0) \neq k$, and otherwise send

$$\left(\left(\boldsymbol{\chi}(\boldsymbol{\rho}_m) \cdot \hat{y}^{(m)}\right) \circ L_{m+1}(j) : j \in [dm+1]\right).$$

We fix a string z (and thus the set C of the verifier's likely decommitments) along with bits of the verifier's random string r that ensure distinguishing bias at least $\varepsilon/2$ and $o(1)$ probability of simulation failure (recall that failure corresponds to the event $(\boldsymbol{\rho}, \ell) \notin C$). Consider the following (linear) mapping between $\mathbb{F}$-vector spaces: from polynomials $g : \mathbb{F}^m \to \mathbb{F}$ of degree at most d that satisfy the $s+1$ linear constraints of the fingerprints and subcube sum (i.e., $g(\boldsymbol{\rho}) = z_i$ for all $(z_i, i) \in C$ and $\sum_{\boldsymbol{\beta} \in H^m} g(\boldsymbol{\beta}) = \alpha$) to the sequence of univariate (partial sum) polynomials $\sum_{\beta_{i+1},\ldots,\beta_m \in H} g(\boldsymbol{\rho}_1, \ldots, \boldsymbol{\rho}_{i-1}, T, \beta_{i+1}, \ldots, \beta_m)$ for all $i \in [m]$ and evaluation points $\boldsymbol{\rho}$ in C.

Let $\ell \leq (d+1)m$ be the dimension of the image of this mapping, and let $\boldsymbol{\xi} = \boldsymbol{\xi}(\boldsymbol{\rho}) \in \mathbb{F}^{(d+1)m \times \ell}$ be the linear coefficients that map vectors in $\mathbb{F}^\ell$ to partial sums (given by $d+1$ evaluations) with respect to $\boldsymbol{\rho}$. We now proceed to Alice's strategy, who receives $w \in \mathbb{F}^{\ell \times p}$ as input and uses a random $y'^{(i)}$ shared with Bob for each commitment string $y^{(i)}$. She will also use $t^{(i)}$ for each pair $y^{(i-1)}, y^{(i)}$; additionally, $t^{(1)}$ and $t^{(m+1)}$ will be used for $y^{(1)}$ and $y^{(m)}$, respectively. The $t^{(i)}$ will ensure Bob knows the linear combination of every algebraic decommitment.

More precisely, Alice runs S (with the fixed string z and partially fixed r) until the end of Step 0, determines the set C, samples $\boldsymbol{\rho}' \sim F = \{z_i : (z_i, i) \in C\}$ and sets $\boldsymbol{\xi} = \boldsymbol{\xi}(\boldsymbol{\rho}')$. For every $(i,j) \in [m-1] \times [d]$ and $(i,j) \in \{m\} \times [d-1]$, she sets $y^{(i)}_j = y'^{(i)}_j + (\boldsymbol{\xi} \cdot w)_{(i-1)d+j}$. She also sets the remaining rows (i.e., $y^{(i)}_{d+1}$ for all i as well as $y^{(m)}_d$) to satisfy

$$\begin{aligned} \boldsymbol{\theta} \cdot y^{(1)} &= t^{(1)}, \\ \boldsymbol{\theta} \cdot y^{(i)} - \boldsymbol{\chi}(\boldsymbol{\rho}'_{i-1}) \cdot y^{(i-1)} &= t^{(i)} \qquad \text{for } 1 < i \leq m \text{ and} \\ \boldsymbol{\chi}(\boldsymbol{\rho}'_m) \cdot y^{(m)} &= t^{(m+1)}. \end{aligned}$$

Note that these are $m+1$ linear constraints on $m+1$ row vectors of dimension p, and since $\boldsymbol{\theta}_{d+1}$ and $\chi_d(\boldsymbol{\rho}'_m)$ are nonzero, there is at least one solution.[21] (If some constraint is not independent from the others, Alice replaces it with a "canonical" constraint to ensure a unique solution, e.g., setting the linear coefficients for $y^{(i)}_{d+1}$ with the smallest bit representation that makes the constraint independent.) She then simulates Step 1 and the part of Step 2 until $\widetilde{V}$ (and D) finish streaming the $y^{(i)}$, sending the resulting snapshots of S, $\widetilde{V}$ and D to Bob along with $\boldsymbol{\rho}'$ in a poly(q)-bit message.

Bob reads his input $(\boldsymbol{\eta}, k)$ and sets the correction tuples $\boldsymbol{\gamma}^{(i)} \in \mathbb{F}^{d+1}$ so as to satisfy constraints with the same linear coefficients as $y^{(i)}$: he sets $\boldsymbol{\gamma}^{(i)}_j =$

[21]The condition $\chi_d(\boldsymbol{\rho}'_m) \neq 0$ follows from choosing $\boldsymbol{\rho}'_m \notin [d+1]$, and we assume the last entry of $\boldsymbol{\theta}$ is nonzero without loss of generality. Note that if $\boldsymbol{\theta}$ is the zero vector the problem trivialises: in this case the verifier does not need assistance from a prover (or even to stream x), accepting if and only if $\alpha = 0$.

$(\boldsymbol{\xi}\cdot\boldsymbol{\eta})_{(i-1)d+j} - y'^{(i)}_{jk}$ for $(i,j)\in[m-1]\times[d]$ and $(i,j)\in\{m\}\times[d-1]$; then sets the coordinates $i=d+1$ and $j\in[m]$ as well as $(i,j)=(d,m)$ to satisfy

$$\begin{aligned}
\boldsymbol{\theta}\cdot\boldsymbol{\gamma}^{(1)} &= \alpha - t^{(1)}_k,\\
\boldsymbol{\theta}\cdot\boldsymbol{\gamma}^{(i)} - \boldsymbol{\chi}(\boldsymbol{\rho}'_{i-1})\cdot\boldsymbol{\gamma}^{(i-1)} &= -t^{(i)}_k \quad \text{for } 1<i\le m \text{ and}\\
\boldsymbol{\chi}(\boldsymbol{\rho}'_m)\cdot\boldsymbol{\gamma}^{(m)} &= f^x(\boldsymbol{\rho}') - t^{(m+1)}_k.
\end{aligned}$$

Bob then finishes the simulation of Step 2 with the coordinate $k\in[p]$.

In Step 3, if $\boldsymbol{\rho}'\neq\boldsymbol{\rho}$ or the simulation fails (i.e., $z_\ell=\boldsymbol{\rho}$ but $(\boldsymbol{\rho},\ell)\notin C$), Bob accepts or rejects uniformly at random. Otherwise, he simulates Step 4 until the protocol terminates (which his access to the shared random strings $t^{(i)}$ enables him to). At the end of the simulation, Bob accepts if and only if D accepts.

Note that, when $\boldsymbol{\eta}=\boldsymbol{\tau}-(w_{ik}: i\in[\ell])$ for a vector $\boldsymbol{\tau}$ that maps to the polynomials $(g_i: i\in[m])$ via $\boldsymbol{\xi}=\boldsymbol{\xi}(\boldsymbol{\rho})$, then $\boldsymbol{\gamma}^{(i)}_j$ satisfies

$$\begin{aligned}
\boldsymbol{\gamma}^{(i)}_j &= (\boldsymbol{\xi}\cdot\boldsymbol{\eta})_{(i-1)d+j} - y'^{(i)}_j\\
&= g_i(j) - (\boldsymbol{\xi}\cdot w)_{(i-1)d+j,k} - y'^{(i)}_{jk}\\
&= g_i(j) - y^{(i)}_{jk}
\end{aligned}$$

for all i,j in $[m-1]\times[d]$ and $\{m\}\times[d-1]$ (equivalently, for all i,j such that $y^{(i)}_j$ includes a linear combination of the rows of w). Then the linear constraints satisfied by the other $m+1$ pairs ensures the equality extends to all (i,j): for $i\in[m], j=d+1$ and $(i,j)=(m,d)$, we have

$$\begin{aligned}
\boldsymbol{\theta}\cdot\boldsymbol{\gamma}^{(1)} &= \alpha - t^{(1)}_k\\
&= \sum_{j=1}^{d+1}\boldsymbol{\theta}_j g(j) - t^{(1)}_k\\
&= \sum_{j=1}^{d+1}\boldsymbol{\theta}_j\left(g(j) - y^{(1)}_{jk}\right),\\
\boldsymbol{\chi}(\boldsymbol{\rho}_m)\cdot\boldsymbol{\gamma}^{(m)} &= f^x(\boldsymbol{\rho}) - t^{(m+1)}_k\\
&= g_m(\boldsymbol{\rho}_m) - t^{(m+1)}_k\\
&= \sum_{j=1}^{d+1}\chi_j(\boldsymbol{\rho}_m)\left(g_m(j) - y^{(m)}_{jk}\right),
\end{aligned}$$

and, for $1 < i \leq m$,

$$\begin{aligned}\boldsymbol{\theta} \cdot \boldsymbol{\gamma}^{(i)} - \boldsymbol{\chi}(\boldsymbol{\rho}_{i-1}) \cdot \boldsymbol{\gamma}^{(i-1)} &= -t_k^{(i)} \\ &= \sum_{j=1}^{d+1} \left(\chi_j(\boldsymbol{\rho}_{i-1}) y_{jk}^{(i-1)} - \boldsymbol{\theta}_j y_{jk}^{(i)} \right) \\ &= \sum_{j=1}^{d+1} \boldsymbol{\theta}_j \left(g_i(j) - y_{jk}^{(i)} \right) + \sum_{j=1}^{d+1} \chi_j(\boldsymbol{\rho}_{i-1}) \left(g_{i-1}(j) - y_{jk}^{(i-1)} \right).\end{aligned}$$

That is, since the $\boldsymbol{\gamma}^{(i)}$ satisfy the same linear constraints as the vectors $\left(g_i(j) - y_{jk}^{(i)} : j \in [d+1]\right)$, it follows that they are equal. Therefore the resulting view is distributed *exactly* as $\text{View}_{P,\widetilde{V}}(x,r)$ when $\boldsymbol{\tau}$ maps to the partial sums of f^x (and thus $\boldsymbol{\xi}(\boldsymbol{\rho}) \cdot \tau$ maps to the partial sums with respect to $\boldsymbol{\rho}$); and if $\boldsymbol{\eta} \sim \mathbb{F}^\ell$, it is distributed as $S(\widetilde{V}, x, r)$ (unless the simulation fails or $\boldsymbol{\rho} \neq \boldsymbol{\rho}'$).

This one-way protocol achieves bias 0 when the simulation fails (an $o(1)$-probability event) or the verifier's temporal decommitment $\boldsymbol{\rho}$ is in C (i.e., the simulation succeeds) but $\boldsymbol{\rho} \neq \boldsymbol{\rho}'$, an event with conditional probability $1 - \frac{1}{|C|} = 1 - \frac{1}{s}$. Otherwise, it achieves a bias of $\varepsilon/2$. We thus have

$$\begin{aligned}&\mathbb{P}_{\substack{w \sim \mathbb{F}^{\ell \times p} \\ k \sim [p]}} \left[B\left(A(w), \left(f^x(i) - w_{ik} : i \in [\ell]\right), k\right) \text{ accepts} \right] \\ &\qquad - \mathbb{P}_{\substack{w \sim \mathbb{F}^{\ell \times p} \\ k \sim [p] \\ \boldsymbol{\eta} \sim \mathbb{F}^\ell}} \left[B\left(A(w), \boldsymbol{\eta}, k\right) \text{ accepts} \right] \\ &\qquad = o(1) \cdot 0 + \left(1 - o(1)\right) \cdot \left(1 - \frac{1}{s}\right) \cdot 0 + \left(1 - o(1)\right) \cdot \frac{1}{s} \cdot \frac{\varepsilon}{2} \\ &\qquad \geq \frac{\varepsilon}{3s}.\end{aligned}$$

Applying Lemma 6.2 yields a one-way binary INDEX protocol for strings of length $p = q^{\log \log q}$ with messages of length $\frac{s^2 \ell^2 \log^2 q}{\varepsilon^2} \text{poly}(q) = \text{poly}(q)$ and constant bias, a contradicting Proposition 6.1's upper bound of $O\left(\sqrt{\text{poly}(q)/p}\right) = o(1)$. □

6.4.4 Applications: FREQUENCY-MOMENT and INNER-PRODUCT

We now proceed to applications of zk-sumcheck. The first is a zkSIP that (exactly) computes frequency moments of order $k > 1$ (commonly denoted F_k) for a stream over an alphabet of size ℓ, a problem known to require $\Omega(\ell)$ space without a prover [AMS99]. For the definition below, we set $\varphi_i(x) := |\{j \in [n] : x_j = i\}|$.

Definition 6.16. *Fix $k \in \mathbb{N}$. For every $\ell \in [n]$ and $t \in [n^k]$, the language* FREQUENCY-MOMENT$_k(t)$ *is* $\left\{x \in [\ell]^n : \sum_{i \in [\ell]} \varphi_i(x)^k = t\right\}$.

Corollary 6.5. *Fix $1 < k \in \mathbb{N}$ and $\delta \in (0,1]$. For every $\ell \in [n]$ and $t \in [n^k]$, there exists a zero-knowledge SIP for* FREQUENCY-MOMENT$_k(t)$ *with space complexity $O(\log^2 n/\log\log n)$. The communication complexity is $O(n^{1+\delta})$ in the setup and $n^{o(1)}$ in the interactive phase, and the protocol is secure against* $\operatorname{polylog}(n)$*-space distinguishers.*

Proof. We set the same parameters of Corollary 6.1: degree $d = \log^{\frac{2}{\delta}} n$, dimension $m = \frac{\delta \log n}{2 \log\log n}$, and take a field $\mathbb{F}$ of size $|\mathbb{F}| = q = \Theta\left(\log^{1+\frac{2}{\delta}} n\right)$. The mapping $x \mapsto f^x$ is defined as follows: viewing $[\ell] \hookrightarrow [d+1]^m \hookrightarrow \mathbb{F}^m$ and defining the frequency vector $\varphi = \varphi(x) := \big(\varphi_i(x) : i \in [\ell]\big)$, set $f^x(\boldsymbol{\alpha}) := \sum_{i\in[d+1]} \hat{\varphi}(i,\boldsymbol{\alpha})^k$ for $\boldsymbol{\alpha} \in \mathbb{F}^{m-1}$. Note that f^x is a $(m-1)$-variate degree-dk polynomial.

Using $O(dm\log q) = O(m^2 \log q)$ bits of space, the verifier can compute all the low-degree extensions $\hat{\varphi}(i,\boldsymbol{\rho}) \in \mathbb{F}$ (by adding $\chi_{x_j}(i,\boldsymbol{\rho})$ to each running sum upon reading x_j); then, after the stream, V raises each LDE to the k^{th} power and adds the results to obtain $f^x(\boldsymbol{\rho})$.

Applying Protocol 6.7, the verifier checks whether

$$\sum_{\boldsymbol{\alpha}\in[d+1]^{m-1}} f^x(\boldsymbol{\alpha}) = \sum_{\boldsymbol{\beta}\in[d+1]^m} \hat{\varphi}(\boldsymbol{\beta})^k = \sum_{i\in[\ell]} \varphi_i^k$$

is equal to t. The space complexity is $O(m^2\log q) = O(\log^2 n/\log\log n)$; the communication complexity of the setup step is of order

$$q^m m \log^2 q = n^{1+\frac{\delta}{2}} \operatorname{polylog}(n) = O\left(n^{1+\delta}\right),$$

and $q^{\log\log q}\operatorname{poly}(q) = n^{o(1)}$ in the interactive phase. Lastly, the protocol is secure against distinguishers with space $\operatorname{poly}(q) = \operatorname{polylog}(n)$. □

Our second and last last application is a small modification of the F_2 protocol that allows us to compute inner products.

Definition 6.17. *For every $\ell \in [n]$, $t \in [n^2\ell]$ and field $\mathbb{F}$,* INNER-PRODUCT(t) *is defined as $\left\{(x,y) \in \mathbb{F}^n \times \mathbb{F}^n : \varphi(x)\cdot\varphi(y) = \sum_{i\in[\ell]} \varphi_i(x)\varphi_i(y) = t\right\}$.*

Corollary 6.6. *For every $\delta \in (0,1]$, $\ell \in [n]$, $t \in [n^2\ell]$ and field $\mathbb{F}_q$ with $q = \Theta\left(\log^{1+\frac{2}{\delta}} n\right)$, there exists a zkSIP for* INNER-PRODUCT(t) *with space complexity $O(\log^2 n/\log\log n)$ and communication complexities $O(n^{1+\delta})$ and $n^{o(1)}$ in the setup and communication phases, respectively.*

Proof. We use the same parameter settings as Corollary 6.5 and define

$$f^{x,y}(\boldsymbol{\alpha}) = \sum_{i\in[d+1]} \widehat{\varphi(x)}(i,\boldsymbol{\alpha})\widehat{\varphi(y)}(i,\boldsymbol{\alpha}),$$

a polynomial of degree $2d = 2\log^{\frac{2}{\delta}} n$ whose evaluation the verifier computes by saving $\widehat{\varphi(x)}(i, \boldsymbol{\rho})$ and $\widehat{\varphi(y)}(i, \boldsymbol{\rho})$ for $i \in [d+1]$. Protocol 6.6 enables the verifier to check that $\sum_{i\in[\ell]} \varphi_i(x)\varphi_i(y)$ equals t, as desired, with complexities of the same order as in Corollary 6.5. $\square$

We remark that while one might reduce inner product to F_2, by taking the difference between the second moment of $\varphi(x) + \varphi(y)$ and the second moments of $\varphi(x)$ and $\varphi(y)$, the resulting protocol leaks these values, and is therefore not zero-knowledge.

Bibliography

[Aar09] S. Aaronson. **On Perfect Completeness for QMA**. In: *Quantum Information and Computation* 9.1&2 (Jan. 2009), pp. 81–89. ISSN: 15337146. DOI: 10.26421/QIC9.1-2-5. 11

[Aar10] Scott Aaronson. **BQP and the Polynomial Hierarchy**. In: *Proceedings of the 42nd ACM Symposium on Theory of Computing, STOC 2010, Cambridge, Massachusetts, USA, 5-8 June 2010*. Ed. by Leonard J. Schulman. 2010, pp. 141–150. DOI: 10.1145/1806689.1806711. 120

[Aar12] Scott Aaronson. **Impossibility of Succinct Quantum Proofs for Collision-Freeness**. In: *Quantum Inf. Comput.* 12.1-2 (2012), pp. 21–28. DOI: 10.26421/QIC12.1-2-3. 13, 15

[Aar21] Scott Aaronson. **Open Problems Related to Quantum Query Complexity**. In: *ACM Transactions on Quantum Computing* 2.4 (Dec. 2021), pp. 1–9. ISSN: 2643-6809, 2643-6817. DOI: 10.1145/3488559. 32

[AA18] Scott Aaronson and Andris Ambainis. **Forrelation: A Problem That Optimally Separates Quantum from Classical Computing**. In: *SIAM J. Comput.* 47.3 (2018), pp. 982–1038. DOI: 10.1137/15M1050902. 3, 12, 119, 121

[ABK$^+$21] Scott Aaronson, Shalev Ben-David, Robin Kothari, Shravas Rao, and Avishay Tal. **Degree vs. Approximate Degree and Quantum Implications of Huang's Sensitivity Theorem**. In: *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing, Virtual Event, Italy, June 21-25, 2021*. Ed. by Samir Khuller and Virginia Vassilevska Williams. ACM, 2021, pp. 1330–1342. DOI: 10.1145/3406325.3451047. 2

[AK07] Scott Aaronson and Greg Kuperberg. **Quantum versus Classical Proofs and Advice**. In: *Theory Comput.* 3.1 (2007), pp. 129–157. DOI: 10.4086/toc.2007.v003a007. 11, 15, 16, 146

[AMS99] Noga Alon, Yossi Matias, and Mario Szegedy. **The Space Complexity of Approximating the Frequency Moments**. In: *Journal of Computer and System Sciences* 58.1 (1999), pp. 137–147. DOI: 10.1006/jcss.1997.1545. 2, 24, 70, 196, 206

[Amb07] Andris Ambainis. **Quantum Walk Algorithm for Element Distinctness**. In: *SIAM J. Comput.* 37.1 (2007), pp. 210–239. DOI: 10.1137/S0097539705447311. 58, 143, 145

[ABRW16] Andris Ambainis, Aleksandrs Belovs, Oded Regev, and Ronald de Wolf. **Efficient Quantum Algorithms for (Gapped) Group Testing and Junta Testing**. In: *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*. Ed. by Robert Krauthgamer. SIAM, 2016, pp. 903–922. DOI: 10.1137/1.9781611974331.ch65. 12

[ACL11] Andris Ambainis, Andrew M. Childs, and Yi-Kai Liu. **Quantum Property Testing for Bounded-Degree Graphs**. In: *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques - 14th International Workshop, APPROX 2011, and 15th International Workshop, RANDOM 2011, Princeton, NJ, USA, August 17-19, 2011. Proceedings*. Ed. by Leslie Ann Goldberg, Klaus Jansen, R. Ravi, and José D. P. Rolim. Vol. 6845. Lecture Notes in Computer Science. Springer, 2011, pp. 365–376. DOI: 10.1007/978-3-642-22935-0_31. 12, 58, 143, 144

[AB09] Sanjeev Arora and Boaz Barak. **Computational Complexity - A Modern Approach**. Cambridge University Press, 2009. ISBN: 978-0-521-42426-4. DOI: 10.1017/CBO9780511804090. 196

[ALM+98] Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy. **Proof Verification and the Hardness of Approximation Problems**. In: *J. ACM* 45.3 (1998), pp. 501–555. DOI: 10.1145/278298.278306. 2

[AS98] Sanjeev Arora and Shmuel Safra. **Probabilistic Checking of Proofs: A New Characterization of NP**. In: *J. ACM* 45.1 (1998), pp. 70–122. DOI: 10.1145/273865.273901. 2

[AAB+19] Frank Arute, Kunal Arya, Ryan Babbush, Dave Bacon, Joseph C Bardin, Rami Barends, Rupak Biswas, Sergio Boixo, Fernando GSL Brandão, David A Buell, et al. **Quantum Supremacy Using a Programmable**

Superconducting Processor. In: *Nature* 574.7779 (2019), pp. 505–510. DOI: 10.1038/s41586-019-1666-5. 3

[AS21] Vahid R. Asadi and Igor Shinkar. **Relaxed Locally Correctable Codes with Improved Parameters**. In: *48th International Colloquium on Automata, Languages, and Programming, ICALP 2021, July 12-16, 2021, Glasgow, Scotland (Virtual Conference)*. Ed. by Nikhil Bansal, Emanuela Merelli, and James Worrell. Vol. 198. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021, 18:1–18:12. DOI: 10.4230/LIPIcs.ICALP.2021.18. 8, 54, 110, 115

[ABFR94] James Aspnes, Richard Beigel, Merrick L. Furst, and Steven Rudich. **The Expressive Power of Voting Polynomials**. In: *Comb.* 14.2 (1994), pp. 135–148. DOI: 10.1007/BF01215346. 126

[BBD$^+$02] Brian Babcock, Shivnath Babu, Mayur Datar, Rajeev Motwani, and Jennifer Widom. **Models and Issues in Data Stream Systems**. In: *Proceedings of the Twenty-First ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*. 2002, pp. 1–16. DOI: 10.1145/543613.543615. 2

[BOW19] Costin Badescu, Ryan O'Donnell, and John Wright. **Quantum State Certification**. In: *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC 2019, Phoenix, AZ, USA, June 23-26, 2019*. Ed. by Moses Charikar and Edith Cohen. ACM, 2019, pp. 503–514. DOI: 10.1145/3313276.3316344. 12

[BBC$^+$01] Robert Beals, Harry Buhrman, Richard Cleve, Michele Mosca, and Ronald de Wolf. **Quantum Lower Bounds by Polynomials**. In: *Journal of the ACM (JACM)* 48.4 (2001), pp. 778–797. DOI: 10.1145/502090.502097. 2

[BBHR18] Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, and Michael Riabzev. **Scalable, Transparent, and Post-Quantum Secure Computational Integrity**. Cryptology ePrint Archive, Report 2018/046. 2018. 19

[BCF$^+$17] Eli Ben-Sasson, Alessandro Chiesa, Michael A. Forbes, Ariel Gabizon, Michael Riabzev, and Nicholas Spooner. **Zero Knowledge Protocols from Succinct Constraint Detection**. In: *Theory of Cryptography - 15th International Conference, TCC 2017, Baltimore, MD, USA, November 12-15, 2017, Proceedings, Part II*. Ed. by Yael Kalai and Leonid Reyzin. Vol. 10678. Lecture Notes in Computer Science. Springer, 2017, pp. 172–206. DOI: 10.1007/978-3-319-70503-3_6. 201

[BCG$^+$14] Eli Ben-Sasson, Alessandro Chiesa, Christina Garman, Matthew Green, Ian Miers, Eran Tromer, and Madars Virza. **Zerocash: Decentralized Anonymous Payments from Bitcoin**. In: *2014 IEEE Symposium on Security and Privacy*. 2014, pp. 459–474. DOI: 10.1109/SP.2014.36. 19

[BCG$^+$13] Eli Ben-Sasson, Alessandro Chiesa, Daniel Genkin, Eran Tromer, and Madars Virza. **SNARKs for C: Verifying Program Executions Succinctly and in Zero Knowledge**. In: *Advances in Cryptology – CRYPTO 2013*. Ed. by Ran Canetti and Juan A. Garay. Vol. 8043. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 90–108. ISBN: 978-3-642-40083-4 978-3-642-40084-1. DOI: 10.1007/978-3-642-40084-1_6. 19

[BCG$^+$19] Eli Ben-Sasson, Alessandro Chiesa, Lior Goldberg, Tom Gur, Michael Riabzev, and Nicholas Spooner. **Linear-Size Constant-Query IOPs for Delegating Computation**. In: *Theory of Cryptography*. Ed. by Dennis Hofheinz and Alon Rosen. Vol. 11892. Cham: Springer International Publishing, 2019, pp. 494–521. ISBN: 978-3-030-36032-0 978-3-030-36033-7. DOI: 10.1007/978-3-030-36033-7_19. 19

[BCR$^+$19] Eli Ben-Sasson, Alessandro Chiesa, Michael Riabzev, Nicholas Spooner, Madars Virza, and Nicholas P. Ward. **Aurora: Transparent Succinct Arguments for R1CS**. In: *Advances in Cryptology – EUROCRYPT 2019*. Ed. by Yuval Ishai and Vincent Rijmen. Cham: Springer International Publishing, 2019, pp. 103–128. DOI: 10.1007/978-3-030-17653-2_4. 19

[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**. In: *SIAM J. Comput.* 36.4 (2006), pp. 889–974. DOI: 10.1137/S0097539705446810. 7, 8, 10, 54, 78, 79, 104, 110, 115

[BHLM09] Eli Ben-Sasson, Prahladh Harsha, Oded Lachish, and Arie Matsliah. **Sound 3-Query PCPPs Are Long**. In: *ACM Transactions on Computation Theory* 1.2 (Sept. 2009), pp. 1–49. ISSN: 1942-3454, 1942-3462. DOI: 10.1145/1595391.1595394. 10

[BRV18] Itay Berman, Ron D. Rothblum, and Vinod Vaikuntanathan. **Zero-Knowledge Proofs of Proximity**. In: *9th Innovations in Theoretical Computer Science Conference, ITCS 2018, January 11-14, 2018, Cambridge, MA, USA*. Ed. by Anna R. Karlin. Vol. 94. LIPIcs. Schloss

Dagstuhl - Leibniz-Zentrum für Informatik, 2018, 19:1–19:20. DOI: 10.4230/LIPIcs.ITCS.2018.19. 9

[BMR19a] Piotr Berman, Meiram Murzabulatov, and Sofya Raskhodnikova. **Testing Convexity of Figures under the Uniform Distribution**. In: *Random Structures & Algorithms* 54.3 (May 2019), pp. 413–443. ISSN: 1042-9832, 1098-2418. DOI: 10.1002/rsa.20797. 6

[BMR19b] Piotr Berman, Meiram Murzabulatov, and Sofya Raskhodnikova. **The Power and Limitations of Uniform Samples in Testing Properties of Figures**. In: *Algorithmica* 81.3 (Mar. 2019), pp. 1247–1266. ISSN: 0178-4617, 1432-0541. DOI: 10.1007/s00453-018-0467-9. 6

[BGS15] Arnab Bhattacharyya, Elena Grigorescu, and Asaf Shapira. **A Unified Framework for Testing Linear-Invariant Properties**. In: *Random Structures & Algorithms* 46.2 (Mar. 2015), pp. 232–260. ISSN: 10429832. DOI: 10.1002/rsa.20507. 6

[Bla09] Eric Blais. **Testing Juntas Nearly Optimally**. In: *Proceedings of the Forty-First Annual ACM Symposium on Theory of Computing*. STOC '09: Symposium on Theory of Computing. Bethesda MD USA: ACM, May 31, 2009, pp. 151–158. ISBN: 978-1-60558-506-2. DOI: 10.1145/1536414.1536437. 12

[BBM12] Eric Blais, Joshua Brody, and Kevin Matulef. **Property Testing Lower Bounds via Communication Complexity**. In: *Comput. Complex.* 21.2 (2012), pp. 311–358. DOI: 10.1007/s00037-012-0040-x. 51, 52

[Blu83] Manuel Blum. **Coin Flipping by Telephone a Protocol for Solving Impossible Problems**. In: *ACM SIGACT News* 15.1 (Jan. 1983), pp. 23–27. ISSN: 0163-5700. DOI: 10.1145/1008908.1008911. 154

[BLR93] Manuel Blum, Michael Luby, and Ronitt Rubinfeld. **Self-Testing/Correcting with Applications to Numerical Problems**. In: *Journal of Computer and System Sciences* 47.3 (1993), pp. 549–595. DOI: 10.1016/0022-0000(93)90044-W. 12

[BLM13] Stéphane Boucheron, Gábor Lugosi, and Pascal Massart. **Concentration Inequalities: A Nonasymptotic Theory of Independence**. Oxford University Press, 2013. DOI: 10.1093/acprof:oso/9780199535255.001.0001. 36

[BHMT02] Gilles Brassard, Peter Hoyer, Michele Mosca, and Alain Tapp. **Quantum Amplitude Amplification and Estimation**. In: *Contemporary Mathematics* 305 (2002), pp. 53–74. DOI: 10.1090/conm/305/05215. 12, 55, 127

[BHH11] Sergey Bravyi, Aram W. Harrow, and Avinatan Hassidim. **Quantum Algorithms for Testing Properties of Distributions**. In: *IEEE Transactions on Information Theory* 57.6 (June 2011), pp. 3971–3981. ISSN: 0018-9448, 1557-9654. DOI: 10.1109/TIT.2011.2134250. 13

[BCL20] Sébastien Bubeck, Sitan Chen, and Jerry Li. **Entanglement Is Necessary for Optimal Quantum Property Testing**. In: *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020, Durham, NC, USA, November 16-19, 2020*. Ed. by Sandy Irani. IEEE, 2020, pp. 692–703. DOI: 10.1109/FOCS46700.2020.00070. 12

[BFNR08] Harry Buhrman, Lance Fortnow, Ilan Newman, and Hein Röhrig. **Quantum Property Testing**. In: *SIAM J. Comput.* 37.5 (2008), pp. 1387–1400. DOI: 10.1137/S0097539704442416. 16, 53

[BW02] Harry Buhrman and Ronald de Wolf. **Complexity Measures and Decision Tree Complexity: A Survey**. In: *Theoretical Computer Science* 288.1 (Oct. 2002), pp. 21–43. ISSN: 03043975. DOI: 10.1016/S0304-3975(01)00144-X. 32

[CGG$^+$19] Clément L. Canonne, Elena Grigorescu, Siyao Guo, Akash Kumar, and Karl Wimmer. **Testing k-Monotonicity: The Rise and Fall of Boolean Functions**. In: *Theory Comput.* 15 (2019), pp. 1–55. DOI: 10.4086/toc.2019.v015a001. 136, 138

[CG18] Clément L. Canonne and Tom Gur. **An Adaptivity Hierarchy Theorem for Property Testing**. In: *Computational Complexity* 27.4 (Dec. 2018), pp. 671–716. ISSN: 1016-3328, 1420-8954. DOI: 10.1007/s00037-018-0168-4. 7

[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*. Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms. Society for Industrial and Applied Mathematics, Jan. 5, 2014, pp. 687–706. ISBN: 978-1-61197-338-9 978-1-61197-340-2. DOI: 10.1137/1.9781611973402.52. 19

[CCMT14] Amit Chakrabarti, Graham Cormode, Andrew McGregor, and Justin Thaler. **Annotations in Data Streams**. In: *ACM Trans. Algorithms* 11.1 (2014), 7:1–7:30. DOI: 10.1145/2636924. 196

[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, June 17-19, 2015, Portland, Oregon, USA. Vol. 33. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, Germany, 2015, pp. 217–243. DOI: 10.4230/LIPICS.CCC.2015.217. 19, 23, 59, 158

[CCM$^+$19] Amit Chakrabarti, Graham Cormode, Andrew McGregor, Justin Thaler, and Suresh Venkatasubramanian. **Verifiable Stream Computation and Arthur–Merlin Communication**. In: *SIAM Journal on Computing* 48.4 (Jan. 2019), pp. 1265–1299. ISSN: 0097-5397, 1095-7111. DOI: 10.1137/17M112289X. 59, 157, 193

[CG19] Amit Chakrabarti and Prantar Ghosh. **Streaming Verification of Graph Computations via Graph Structure**. In: *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2019, September 20-22, 2019, Massachusetts Institute of Technology, Cambridge, MA, USA*. Ed. by Dimitris Achlioptas and László A. Végh. Vol. 145. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019, 70:1–70:20. DOI: 10.4230/LIPIcs.APPROX-RANDOM.2019.70. 19

[CGT20] Amit Chakrabarti, Prantar Ghosh, and Justin Thaler. **Streaming Verification for Graph Problems: Optimal Tradeoffs and Nonlinear Sketches**. In: *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2020, August 17-19, 2020, Virtual Conference*. Ed. by Jaroslaw Byrka and Raghu Meka. Vol. 176. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020, 22:1–22:23. DOI: 10.4230/LIPIcs.APPROX/RANDOM.2020.22. 19

[CM13] Kaushik Chakraborty and Subhamoy Maitra. **Improved Quantum Test for Linearity of a Boolean Function**. In: *CoRR* abs/1306.6195 (2013). DOI: 10.48550/arXiv.1306.6195. 12

[CFMW10] Sourav Chakraborty, Eldar Fischer, Arie Matsliah, and Ronald de Wolf. **New Results on Quantum Property Testing**. In: *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2010, December 15-18, 2010, Chennai, India*. Ed. by Kamal Lodaya and Meena Mahajan. Vol. 8. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2010, pp. 145–156. DOI: 10.4230/LIPIcs.FSTTCS.2010.145. 12, 13

[Che16] Lijie Chen. **A Note on Oracle Separations for BQP**. In: *CoRR* abs/1605.00619 (2016). DOI: 10.48550/arXiv.1605.00619. 120

[CGW13] Victor Chen, Elena Grigorescu, and Ronald de Wolf. **Error-Correcting Data Structures**. In: *SIAM J. Comput.* 42.1 (2013), pp. 84–111. DOI: 10.1137/110834949. 7

[CFSS17] Xi Chen, Adam Freilich, Rocco A. Servedio, and Timothy Sun. **Sample-Based High-Dimensional Convexity Testing**. In: *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2017, August 16-18, 2017, Berkeley, CA, USA*. Ed. by Klaus Jansen, José D. P. Rolim, David Williamson, and Santosh S. Vempala. Vol. 81. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017, 37:1–37:20. DOI: 10.4230/LIPIcs.APPROX-RANDOM.2017.37. 6

[CGS22] Alessandro Chiesa, Tom Gur, and Igor Shinkar. **Relaxed Locally Correctable Codes with Nearly-Linear Block Length and Constant Query Complexity**. In: *SIAM J. Comput.* 51.6 (2022), pp. 1839–1865. DOI: 10.1137/20m135515x. 110, 115

[CG04] Hana Chockler and Dan Gutfreund. **A Lower Bound for Testing Juntas**. In: *Information Processing Letters* 90.6 (June 2004), pp. 301–305. ISSN: 00200190. DOI: 10.1016/j.ipl.2004.01.023. 12

[CDGH23] Graham Cormode, Marcel Dall'Agnol, Tom Gur, and Chris Hickey. **Streaming Zero-Knowledge Proofs**. In: *CoRR* abs/2301.02161 (2023). DOI: 10.48550/arxiv.2301.02161. viii

[CH18] Graham Cormode and Chris Hickey. **Cheap Checking for Cloud Computing: Statistical Analysis via Annotated Data Streams**. In: *AISTATS*. 2018. 19

[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*. ITCS '12: Innovations in Theoretical Computer Science. Cambridge Massachusetts: ACM, Jan. 8, 2012, pp. 90–112. ISBN: 978-1-4503-1115-1. DOI: 10.1145/2090236.2090245. 19, 20

[CMT13] Graham Cormode, Michael Mitzenmacher, and Justin Thaler. **Streaming Graph Computations with a Helpful Advisor**. In: *Algorithmica* 65.2 (Feb. 2013), pp. 409–442. ISSN: 0178-4617, 1432-0541. DOI: 10.1007/s00453-011-9598-y. 19

[CTY11] Graham Cormode, Justin Thaler, and Ke Yi. **Verifying Computations with Streaming Interactive Proofs**. In: *Proc. VLDB Endow.* 5.1 (2011), pp. 25–36. DOI: 10.14778/2047485.2047488. 19, 150, 157, 195

[DGL21] Marcel Dall'Agnol, Tom Gur, and Oded Lachish. **A Structural Theorem for Local Algorithms with Applications to Coding, Testing, and Privacy**. In: *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms, SODA 2021, Virtual Conference, January 10 - 13, 2021*. Ed. by Dániel Marx. SIAM, 2021, pp. 1651–1665. DOI: 10.1137/1.9781611976465.100. viii

[DGRT22] Marcel Dall'Agnol, Tom Gur, Subhayan Roy Moulik, and Justin Thaler. **Quantum Proofs of Proximity**. In: *Quantum* 6 (Oct. 2022), p. 834. ISSN: 2521-327X. DOI: 10.22331/q-2022-10-13-834. viii

[DS23] Marcel Dall'Agnol and Nicholas Spooner. **On the Necessity of Collapsing for Post-Quantum and Quantum Commitments**. In: *18th Conference on the Theory of Quantum Computation, Communication and Cryptography, TQC 2023, July 24-28, 2023, Aveiro, Portugal*. Ed. by Omar Fawzi and Michael Walter. Vol. 266. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023, 2:1–2:23. DOI: 10.4230/LIPIcs.TQC.2023.2. viii

[DTV15] Samira Daruki, Justin Thaler, and Suresh Venkatasubramanian. **Streaming Verification in Data Analysis**. In: *Algorithms and Computation*. Ed. by Khaled Elbassioni and Kazuhisa Makino. Vol. 9472. Berlin, Heidelberg: Springer Berlin Heidelberg, 2015, pp. 715–726. ISBN: 978-3-662-48970-3 978-3-662-48971-0. DOI: 10.1007/978-3-662-48971-0_60. 19

[DGG19] Irit Dinur, Oded Goldreich, and Tom Gur. **Every Set in P is Strongly Testable under a Suitable Encoding**. In: *10th Innovations in Theoretical Computer Science Conference, ITCS 2019, January 10-12, 2019, San Diego, California, USA*. Ed. by Avrim Blum. Vol. 124. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019, 30:1–30:17. DOI: 10.4230/LIPIcs.ITCS.2019.30. 80

[DH13] Irit Dinur and Prahladh Harsha. **Composition of Low-Error 2-Query PCPs Using Decodable PCPs**. In: *SIAM Journal on Computing* 42.6 (Jan. 2013), pp. 2452–2486. ISSN: 0097-5397, 1095-7111. DOI: 10.1137/100788161. 7

[Efr12] Klim Efremenko. **3-Query Locally Decodable Codes of Subexponential Length**. In: *SIAM J. Comput.* 41.6 (2012), pp. 1694–1703. DOI: 10.1137/090772721. 7

[FGL$^+$91] U. Feige, S. Goldwasser, L. Lovasz, S. Safra, and M. Szegedy. **Approximating Clique Is Almost NP-complete**. In: 32nd Annual Symposium of Foundations of Computer Science. San Juan, Puerto Rico: IEEE Comput. Soc. Press, 1991, pp. 2–12. ISBN: 978-0-8186-2445-2. DOI: 10.1109/SFCS.1991.185341. 2

[Fey86] Richard Feynman. **Quantum Mechanical Computers**. In: *Optics News* 16 (June 1986), pp. 507–531. DOI: 10.1007/BF01886518. 2

[Fey82] Richard P. Feynman. **Simulating Physics with Computers**. In: *International Journal of Theoretical Physics* 21.6-7 (June 1982), pp. 467–488. ISSN: 0020-7748, 1572-9575. DOI: 10.1007/BF02650179. 2

[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*. Ed. by Moni Naor. ACM, 2014, pp. 483–500. DOI: 10.1145/2554797.2554841. 6, 15, 80, 128

[FLV15] Eldar Fischer, Oded Lachish, and Yadu Vasudev. **Trading Query Complexity for Sample-Based Testing and Multi-testing Scalability**. In: 2015 IEEE 56th Annual Symposium on Foundations of Computer Science (FOCS). Berkeley, CA, USA: IEEE, Oct. 2015, pp. 1163–1182. ISBN: 978-1-4673-8191-8. DOI: 10.1109/FOCS.2015.75. 4, 6, 7, 37, 40, 43, 81, 103

[Gam10] João Gama. **Knowledge Discovery from Data Streams**. Chapman and Hall / CRC Data Mining and Knowledge Discovery Series. CRC Press, 2010. ISBN: 978-1-4398-2611-9. 2

[GL20] András Gilyén and Tongyang Li. **Distributional Property Testing in a Quantum World**. In: *11th Innovations in Theoretical Computer Science Conference, ITCS 2020, January 12-14, 2020, Seattle, Washington, USA*. Ed. by Thomas Vidick. Vol. 151. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020, 25:1–25:19. DOI: 10.4230/LIPIcs.ITCS.2020.25. 12

[Gol02] Oded Goldreich. **Zero-Knowledge Twenty Years after Its Invention**. In: *IACR Cryptol. ePrint Arch.* 2002 (2002), p. 186. 19

[Gol08] Oded Goldreich. **Computational Complexity: A Conceptual Perspective**. 1st ed. Cambridge University Press, Apr. 28, 2008. ISBN: 978-0-521-88473-0 978-0-511-80410-6. DOI: 10.1017/CBO9780511804106. 19

[Gol11] Oded Goldreich. **Short Locally Testable Codes and Proofs**. In: *Studies in Complexity and Cryptography. Miscellanea on the Interplay between Randomness and Computation - In Collaboration with Lidor Avigad, Mihir Bellare, Zvika Brakerski, Shafi Goldwasser, Shai Halevi, Tali Kaufman, Leonid Levin, Noam Nisan, Dana Ron, Madhu Sudan, Luca Trevisan, Salil Vadhan, Avi Wigderson, David Zuckerman*. Ed. by Oded Goldreich. Vol. 6650. Lecture Notes in Computer Science. Springer, 2011, pp. 333–372. DOI: 10.1007/978-3-642-22670-0_25. 8

[Gol14] Oded Goldreich. **On Multiple Input Problems in Property Testing**. In: *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2014, September 4-6, 2014, Barcelona, Spain*. Ed. by Klaus Jansen, José D. P. Rolim, Nikhil R. Devanur, and Cristopher Moore. Vol. 28. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2014, pp. 704–720. DOI: 10.4230/LIPIcs.APPROX-RANDOM.2014.704. 134

[Gol17] Oded Goldreich. **Introduction to Property Testing**. Cambridge University Press, 2017. DOI: 10.1017/9781108135252. 5, 32, 137

[Gol20] Oded Goldreich. **On the Communication Complexity Methodology for Proving Lower Bounds on the Query Complexity of Property Testing**. In: *Computational Complexity and Property Testing - On the Interplay Between Randomness and Computation*. Vol. 12050.

Lecture Notes in Computer Science. Springer, 2020, pp. 87–118. DOI: 10.1007/978-3-030-43662-9_7. 53

[GGL$^+$00] Oded Goldreich, Shafi Goldwasser, Eric Lehman, Dana Ron, and Alex Samorodnitsky. **Testing Monotonicity**. In: *Comb.* 20.3 (2000), pp. 301–337. DOI: 10.1007/s004930070011. 138

[GGR98] Oded Goldreich, Shari Goldwasser, and Dana Ron. **Property Testing and Its Connection to Learning and Approximation**. In: *Journal of the ACM* 45.4 (July 1998), pp. 653–750. ISSN: 0004-5411, 1557-735X. DOI: 10.1145/285055.285060. 1

[GG18] Oded Goldreich and Tom Gur. **Universal Locally Testable Codes**. In: *Chicago Journal of Theoretical Computer Science* 24.1 (2018), pp. 1–21. ISSN: 1073-0486. DOI: 10.4086/cjtcs.2018.003. 79

[GGK19] Oded Goldreich, Tom Gur, and Ilan Komargodski. **Strong Locally Testable Codes with Relaxed Local Decoders**. In: *ACM Trans. Comput. Theory* 11.3 (2019), 17:1–17:38. DOI: 10.1145/3319907. 115

[GGR18] Oded Goldreich, Tom Gur, and Ron D. Rothblum. **Proofs of Proximity for Context-Free Languages and Read-Once Branching Programs**. In: *Inf. Comput.* 261 (2018), pp. 175–201. DOI: 10.1016/j.ic.2018.02.003. 17, 129, 130

[GR99] Oded Goldreich and Dana Ron. **A Sublinear Bipartiteness Tester for Bounded Degree Graphs**. In: *Comb.* 19.3 (1999), pp. 335–373. DOI: 10.1007/s004930050060. 145

[GR02] Oded Goldreich and Dana Ron. **Property Testing in Bounded Degree Graphs**. In: *Algorithmica* 32.2 (2002), pp. 302–343. DOI: 10.1007/s00453-001-0078-7. 145

[GR11] Oded Goldreich and Dana Ron. **On Testing Expansion in Bounded-Degree Graphs**. In: *Studies in Complexity and Cryptography. Miscellanea on the Interplay between Randomness and Computation*. Ed. by Oded Goldreich. Vol. 6650. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 68–75. ISBN: 978-3-642-22669-4 978-3-642-22670-0. DOI: 10.1007/978-3-642-22670-0_9. 13

[GR16] Oded Goldreich and Dana Ron. **On Sample-Based Testers**. In: *ACM Transactions on Computation Theory* 8.2 (May 5, 2016), pp. 1–54. ISSN: 1942-3454, 1942-3462. DOI: 10.1145/2898355. 6

[GS10] Oded Goldreich and Or Sheffet. **On The Randomness Complexity of Property Testing**. In: *Computational Complexity* 19.1 (Mar. 2010), pp. 99–133. ISSN: 1016-3328, 1420-8954. DOI: 10.1007/s00037-009-0282-4. 86

[GS06] Oded Goldreich and Madhu Sudan. **Locally Testable Codes and PCPs of Almost-Linear Length**. In: *Journal of the ACM* 53.4 (July 2006), pp. 558–655. ISSN: 0004-5411, 1557-735X. DOI: 10.1145/1162349.1162351. 1, 76, 80

[GKR15] Shafi Goldwasser, Yael Tauman Kalai, and Guy N. Rothblum. **Delegating Computation: Interactive Proofs for Muggles**. In: *Journal of the ACM* 62.4 (Sept. 11, 2015), pp. 1–64. ISSN: 0004-5411, 1557-735X. DOI: 10.1145/2699436. 20, 153, 196

[GM21] Mika Göös and Gilbert Maystre. **A Majority Lemma for Randomised Query Complexity**. In: *36th Computational Complexity Conference, CCC 2021, July 20-23, 2021, Toronto, Ontario, Canada (Virtual Conference)*. Ed. by Valentine Kabanets. Vol. 200. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021, 18:1–18:15. DOI: 10.4230/LIPIcs.CCC.2021.18. 125

[GL21] Tom Gur and Oded Lachish. **On the Power of Relaxed Local Decoding Algorithms**. In: *SIAM J. Comput.* 50.2 (2021), pp. 788–813. DOI: 10.1137/19M1307834. 4, 7, 8, 37, 38, 40, 43, 81, 104, 107

[GLR21] Tom Gur, Yang P. Liu, and Ron D. Rothblum. **An Exponential Separation between MA and AM Proofs of Proximity**. In: *Comput. Complex.* 30.2 (2021), p. 12. DOI: 10.1007/s00037-021-00212-3. 16, 147

[GR15] Tom Gur and Ran Raz. **Arthur-Merlin Streaming Complexity**. In: *Information and Computation* 243 (Aug. 2015), pp. 145–165. ISSN: 08905401. DOI: 10.1016/j.ic.2014.12.011. 193

[GR17] Tom Gur and Ron D. Rothblum. **A Hierarchy Theorem for Interactive Proofs of Proximity**. In: *8th Innovations in Theoretical Computer Science Conference, ITCS 2017, January 9-11, 2017, Berkeley, CA, USA*. Ed. by Christos H. Papadimitriou. Vol. 67. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017, 39:1–39:43. DOI: 10.4230/LIPIcs.ITCS.2017.39. 7

[GR18] Tom Gur and Ron D. Rothblum. **Non-Interactive Proofs of Proximity**. In: *Comput. Complex.* 27.1 (2018), pp. 99–207. DOI: 10.1007/s00037-016-0136-9. 9, 15, 16, 56, 57, 80, 102, 108–110, 131, 132, 136, 138, 142, 143, 145

[GRS12] Venkatesan Guruswami, Atri Rudra, and Madhu Sudan. **Essential Coding Theory**. 2012. 35, 172

[HS70] András Hajnal and E. Szemerédi. **Proof of a Conjecture of P. Erdős**. In: *Colloq Math Soc János Bolyai* 4 (1970). 84

[HHT97] Yenjo Han, Lane A. Hemaspaandra, and Thomas Thierauf. **Threshold Computation and Cryptographic Security**. In: *SIAM J. Comput.* 26.1 (1997), pp. 59–78. DOI: 10.1137/S0097539792240467. 120

[HILL99] Johan Håstad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. **A Pseudorandom Generator from Any One-way Function**. In: *SIAM Journal on Computing* 28.4 (Jan. 1999), pp. 1364–1396. ISSN: 0097-5397, 1095-7111. DOI: 10.1137/S0097539793244708. 22

[Hic21] Christopher J. A. Hickey. **Streaming Interactive Proofs**. PhD thesis. University of Warwick, Coventry, UK, 2021. viii

[HA11] Mark Hillery and Erika Andersson. **Quantum Tests for the Linearity and Permutation Invariance of Boolean Functions**. In: *Phys. Rev. A* 84.6 (Dec. 2011), p. 062329. DOI: 10.1103/PhysRevA.84.062329. 12

[HSX$^+$12] Cheng Huang, Huseyin Simitci, Yikang Xu, Aaron Ogus, Brad Calder, Parikshit Gopalan, Jin Li, and Sergey Yekhanin. **Erasure Coding in Windows Azure Storage**. In: *Presented as Part of the 2012 USENIX Annual Technical Conference.* 2012, pp. 15–26. 5

[HZN$^+$20] Cupjin Huang et al. **Classical Simulation of Quantum Supremacy Circuits**. In: *CoRR* abs/2005.06787 (2020). DOI: 10.48550/arxiv.2005.06787. 3

[Hua19] Hao Huang. **Induced Subgraphs of Hypercubes and a Proof of the Sensitivity Conjecture**. In: *Annals of Mathematics* 190.3 (Nov. 1, 2019). ISSN: 0003-486X. DOI: 10.4007/annals.2019.190.3.6. 2

[IL89] R. Impagliazzo and M. Luby. **One-Way Functions Are Essential for Complexity Based Cryptography**. In: *30th Annual Symposium on Foundations of Computer Science.* 30th Annual Symposium on Foundations of Computer Science. Research Triangle Park, NC, USA:

IEEE, 1989, pp. 230–235. ISBN: 978-0-8186-1982-3. DOI: 10.1109/SFCS.1989.63483. 22

[IV12] Tsuyoshi Ito and Thomas Vidick. **A Multi-prover Interactive Proof for NEXP Sound against Entangled Provers**. In: *2012 IEEE 53rd Annual Symposium on Foundations of Computer Science*. 2012 IEEE 53rd Annual Symposium on Foundations of Computer Science (FOCS). New Brunswick, NJ, USA: IEEE, Oct. 2012, pp. 243–252. ISBN: 978-0-7695-4874-6 978-1-4673-4383-1. DOI: 10.1109/FOCS.2012.11. 12

[JNV$^+$21] Zhengfeng Ji, Anand Natarajan, Thomas Vidick, John Wright, and Henry Yuen. **MIP* = RE**. In: *Commun. ACM* 64.11 (2021), pp. 131–138. DOI: 10.1145/3485628. 12

[KRS23] Gil Kalai, Yosef Rinott, and Tomer Shoham. **Questions and Concerns about Google's Quantum Supremacy Claim**. In: *CoRR* abs/2305.01064 (2023). DOI: 10.48550/arXiv.2305.01064. 3

[KR15] Yael Tauman Kalai and Ron D. Rothblum. **Arguments of Proximity - [Extended Abstract]**. In: *Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part II*. Ed. by Rosario Gennaro and Matthew Robshaw. Vol. 9216. Lecture Notes in Computer Science. Springer, 2015, pp. 422–442. DOI: 10.1007/978-3-662-48000-7_21. 9, 18

[KT00] Jonathan Katz and Luca Trevisan. **On the Efficiency of Local Decoding Procedures for Error-Correcting Codes**. In: The 32nd Annual ACM Symposium on Theory of Computing. Portland Oregon USA: ACM, May 2000, pp. 80–86. ISBN: 978-1-58113-184-0. DOI: 10.1145/335305.335315. 1, 7, 76

[KMS18] Subhash Khot, Dor Minzer, and Muli Safra. **On Monotonicity Testing and Boolean Isoperimetric-Type Theorems**. In: *SIAM J. Comput.* 47.6 (2018), pp. 2238–2276. DOI: 10.1137/16M1065872. 138

[KW00] Alexei Kitaev and John Watrous. **Parallelization, Amplification, and Exponential Time Simulation of Quantum Interactive Proof Systems**. In: The 32nd Annual ACM Symposium on Theory of Computing. Portland Oregon USA: ACM, May 2000, pp. 608–617. ISBN: 978-1-58113-184-0. DOI: 10.1145/335305.335387. 11

[KS17] Swastik Kopparty and Shubhangi Saraf. **Local Testing and Decoding of High-Rate Error-Correcting Codes**. In: *Electronic Colloquium on Computational Complexity (ECCC)* 24 (2017), p. 126. 5

[Lev87] Leonid A. Levin. **One-Way Functions and Pseudorandom Generators**. In: *Comb.* 7.4 (1987), pp. 357–363. DOI: 10.1007/BF02579323. 57, 133

[MLA+22] Lars S. Madsen et al. **Quantum Computational Advantage with a Programmable Photonic Processor**. In: *Nature* 606.7912 (June 2, 2022), pp. 75–81. ISSN: 0028-0836, 1476-4687. DOI: 10.1038/s41586-022-04725-x. 3

[MW05] Chris Marriott and John Watrous. **Quantum Arthur-Merlin Games**. In: *Comput. Complex.* 14.2 (2005), pp. 122–152. DOI: 10.1007/s00037-005-0194-x. 54, 123, 127

[MU05] Michael Mitzenmacher and Eli Upfal. **Probability and Computing: An Introduction to Randomized Algorithms and Probabilistic Analysis**. New York: Cambridge University Press, 2005. 352 pp. ISBN: 978-0-521-83540-4. DOI: 10.1017/CBO9780511813603. 28

[MW16] Ashley Montanaro and Ronald de Wolf. **A Survey of Quantum Property Testing**. In: *Theory Comput.* 7 (2016), pp. 1–81. DOI: 10.4086/toc.gs.2016.007. 12, 51, 52

[MR10] Dana Moshkovitz and Ran Raz. **Two-Query PCP with Subconstant Error**. In: *Journal of the ACM* 57.5 (June 2010), pp. 1–29. ISSN: 0004-5411, 1557-735X. DOI: 10.1145/1754399.1754402. 7

[MP80] J.I. Munro and M.S. Paterson. **Selection and Sorting with Limited Storage**. In: *Theoretical Computer Science* 12.3 (Nov. 1980), pp. 315–323. ISSN: 03043975. DOI: 10.1016/0304-3975(80)90061-4. 2

[Mut05] S. Muthukrishnan. **Data Streams: Algorithms and Applications**. In: *Foundations and Trends in Theoretical Computer Science* 1.2 (2005), pp. 117–236. ISSN: 1551-305X, 1551-3068. DOI: 10.1561/0400000002. 2

[Nao91] Moni Naor. **Bit Commitment Using Pseudorandomness**. In: *Journal of Cryptology* 4.2 (Jan. 1991), pp. 151–158. ISSN: 0933-2790, 1432-1378. DOI: 10.1007/BF00196774. 22

[NV17] Anand Natarajan and Thomas Vidick. **A Quantum Linearity Test for Robustly Verifying Entanglement**. In: *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017*. Ed. by Hamed Hatami, Pierre McKenzie, and Valerie King. ACM, 2017, pp. 1003–1015. DOI: 10.1145/3055399.3055468. 12

[NW19] Anand Natarajan and John Wright. **NEEXP Is Contained in MIP**. In: 2019 IEEE 60th Annual Symposium on Foundations of Computer Science (FOCS). Baltimore, MD, USA: IEEE, Nov. 2019, pp. 510–518. ISBN: 978-1-72814-952-3. DOI: 10.1109/FOCS.2019.00039. 12

[New91] Ilan Newman. **Private vs. Common Random Bits in Communication Complexity**. In: *Information Processing Letters* 39.2 (July 1991), pp. 67–71. ISSN: 00200190. DOI: 10.1016/0020-0190(91)90157-D. 86

[NC16] Michael A. Nielsen and Isaac L. Chuang. **Quantum Computation and Quantum Information (10th Anniversary Edition)**. Cambridge University Press, 2016. ISBN: 978-1-107-00217-3. DOI: 10.1017/CBO9780511976667. 29, 58, 145

[OS10] Ryan O'Donnell and Rocco A. Servedio. **New Degree Bounds for Polynomial Threshold Functions**. In: *Comb.* 30.3 (2010), pp. 327–358. DOI: 10.1007/s00493-010-2173-3. 126

[OW15] Ryan O'Donnell and John Wright. **Quantum Spectrum Testing**. In: *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015, Portland, OR, USA, June 14-17, 2015*. Ed. by Rocco A. Servedio and Ronitt Rubinfeld. ACM, 2015, pp. 529–538. DOI: 10.1145/2746539.2746582. 12

[Rab81] Michael O Rabin. **Fingerprinting by Random Polynomials**. Center for Research in Computing Techn., Aiken Computation Laboratory, Univ., 1981. 157

[RY20] Anup Rao and Amir Yehudayoff. **Communication Complexity: And Applications**. Cambridge University Press, 2020. ISBN: 978-1-108-67164-4. DOI: 10.1017/9781108671644. 23, 62, 159

[RS04] Ran Raz and Amir Shpilka. **On the Power of Quantum Proofs**. In: *19th Annual IEEE Conference on Computational Complexity (CCC 2004), 21-24 June 2004, Amherst, MA, USA*. IEEE Computer Society, 2004, pp. 260–274. DOI: 10.1109/CCC.2004.1313849. 18

[RT22] Ran Raz and Avishay Tal. **Oracle Separation of BQP and PH**. In: *Journal of the ACM* 69.4 (Aug. 2022), pp. 1–21. ISSN: 0004-5411, 1557-735X. DOI: 10.1145/3530258. 3

[Raz03] A A Razborov. **Quantum Communication Complexity of Symmetric Predicates**. In: *Izvestiya: Mathematics* 67.1 (Feb. 2003), pp. 145–159. ISSN: 1468-4810. DOI: 10.1070/im2003v067n01abeh000422. 54, 116, 119

[RRR21] Omer Reingold, Guy N. Rothblum, and Ron D. Rothblum. **Constant-Round Interactive Proofs for Delegating Computation**. In: *SIAM J. Comput.* 50.3 (2021). DOI: 10.1137/16M1096773. 9, 196

[RR20a] Noga Ron-Zewi and Ron D. Rothblum. **Local Proofs Approaching the Witness Length [Extended Abstract]**. In: *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020, Durham, NC, USA, November 16-19, 2020.* Ed. by Sandy Irani. IEEE, 2020, pp. 846–857. DOI: 10.1109/FOCS46700.2020.00083. 7

[RR20b] Guy N. Rothblum and Ron D. Rothblum. **Batch Verification and Proofs of Proximity with Polylog Overhead**. In: *Theory of Cryptography - 18th International Conference, TCC 2020, Durham, NC, USA, November 16-19, 2020, Proceedings, Part II.* Ed. by Rafael Pass and Krzysztof Pietrzak. Vol. 12551. Lecture Notes in Computer Science. Springer, 2020, pp. 108–138. DOI: 10.1007/978-3-030-64378-2_5. 18, 196

[RVW13] Guy N. Rothblum, Salil P. Vadhan, and Avi Wigderson. **Interactive Proofs of Proximity: Delegating Computation in Sublinear Time**. In: *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013.* Ed. by Dan Boneh, Tim Roughgarden, and Joan Feigenbaum. ACM, 2013, pp. 793–802. DOI: 10.1145/2488608.2488709. 9, 18, 196

[RS96] Ronitt Rubinfeld and Madhu Sudan. **Robust Characterizations of Polynomials with Applications to Program Testing**. In: *SIAM J. Comput.* 25.2 (1996), pp. 252–271. DOI: 10.1137/S0097539793255151. 1

[Sch80] J. T. Schwartz. **Fast Probabilistic Algorithms for Verification of Polynomial Identities**. In: *Journal of The ACM* 27.4 (Oct. 1980), pp. 701–717. ISSN: 0004-5411. DOI: 10.1145/322217.322225. 157

[Sha48] Claude E. Shannon. **A Mathematical Theory of Communication**. In: *Bell System Technical Journal* 27.3 (1948), pp. 379–423. DOI: 10.1002/j.1538-7305.1948.tb01338.x. 3

[ST23] Alexander A. Sherstov and Justin Thaler. **Vanishing-Error Approximate Degree and QMA Complexity**. In: *Chicago Journal of Theoretical Computer Science* 2023.3 (2023). DOI: 10.4086/cjtcs.2023.002. 13, 15, 16, 148

[Sho99] Peter W. Shor. **Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer**. In: *SIAM Review* 41.2 (Jan. 1999), pp. 303–332. ISSN: 0036-1445, 1095-7200. DOI: 10.1137/S0036144598347011. 3

[Sim97] Daniel R. Simon. **On the Power of Quantum Computation**. In: *SIAM Journal on Computing* 26.5 (Oct. 1997), pp. 1474–1483. ISSN: 0097-5397, 1095-7111. DOI: 10.1137/S0097539796298637. 3

[SYZ$^+$21] Xiaoqiang Sun, F. Richard Yu, Peng Zhang, Zhiwei Sun, Weixin Xie, and Xiang Peng. **A Survey on Zero-Knowledge Proof in Blockchain**. In: *IEEE Network* 35.4 (2021), pp. 198–205. DOI: 10.1109/MNET.011.2000473. 19

[Tha13] Justin Thaler. **Time-Optimal Interactive Proofs for Circuit Evaluation**. In: *Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part II*. Ed. by Ran Canetti and Juan A. Garay. Vol. 8043. Lecture Notes in Computer Science. Springer, 2013, pp. 71–89. DOI: 10.1007/978-3-642-40084-1_5. 19

[Tha16] Justin Thaler. **Semi-Streaming Algorithms for Annotated Graph Streams**. In: *43rd International Colloquium on Automata, Languages, and Programming, ICALP 2016, July 11-15, 2016, Rome, Italy*. Ed. by Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi. Vol. 55. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016, 59:1–59:14. DOI: 10.4230/LIPIcs.ICALP.2016.59. 19

[Tre04] Luca Trevisan. **Some Applications of Coding Theory in Computational Complexity**. In: *Electronic Colloquium on Computational Complexity (ECCC)* (2004). 5

[Vad07] Salil Vadhan. **The Complexity of Zero Knowledge**. In: *FSTTCS 2007: Foundations of Software Technology and Theoretical Computer Science*. Ed. by V. Arvind and Sanjiva Prasad. Red. by David Hutchison et al. Vol. 4855. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 52–70. ISBN: 978-3-540-77049-7 978-3-540-77050-3. DOI: `10.1007/978-3-540-77050-3_5`. 19

[Vad99] Salil Pravin Vadhan. **A Study of Statistical Zero-Knowledge Proofs**. PhD thesis. Massachusetts Institute of Technology, 1999. 19

[Val11] Paul Valiant. **Testing Symmetric Properties of Distributions**. In: *SIAM Journal on Computing* 40.6 (Jan. 2011), pp. 1927–1968. ISSN: 0097-5397, 1095-7111. DOI: `10.1137/080734066`. 13

[VW16] Thomas Vidick and John Watrous. **Quantum Proofs**. In: *Foundations and Trends in Theoretical Computer Science* 11.1-2 (2016), pp. 1–215. ISSN: 1551-305X. DOI: `10.1561/0400000068`. 11

[Wat00] J. Watrous. **Succinct Quantum Proofs for Properties of Finite Groups**. In: 41st Annual Symposium on Foundations of Computer Science. Redondo Beach, CA, USA: IEEE Comput. Soc, 2000, pp. 537–546. ISBN: 978-0-7695-0850-4. DOI: `10.1109/SFCS.2000.892141`. 11

[Wat03] John Watrous. **PSPACE Has Constant-Round Quantum Interactive Proof Systems**. In: *Theoretical Computer Science* 292.3 (Jan. 2003), pp. 575–588. ISSN: 03043975. DOI: `10.1016/S0304-3975(01)00375-9`. 11

[Wat09] John Watrous. **Quantum Computational Complexity**. In: *Encyclopedia of Complexity and Systems Science*. Ed. by Robert A. Meyers. Springer, 2009, pp. 7174–7201. DOI: `10.1007/978-0-387-30440-3_428`. 127

[WBC$^+$21] Yulin Wu et al. **Strong Quantum Computational Advantage Using a Superconducting Quantum Processor**. In: *Physical Review Letters* 127.18 (Oct. 2021), p. 180501. ISSN: 0031-9007, 1079-7114. DOI: `10.1103/PhysRevLett.127.180501`. 3

[Yek08] Sergey Yekhanin. **Towards 3-Query Locally Decodable Codes of Subexponential Length**. In: *J. ACM* 55.1 (2008), 1:1–1:16. DOI: `10.1145/1326554.1326555`. 7

[Yek12] Sergey Yekhanin. **Locally Decodable Codes**. In: *Foundations and Trends in Theoretical Computer Science* 6.3 (2012), pp. 139–255. DOI: 10.1561/0400000030. 5

[ZWD$^+$20] Han-Sen Zhong et al. **Quantum Computational Advantage Using Photons**. In: *Science* 370.6523 (Dec. 2020), pp. 1460–1463. ISSN: 0036-8075, 1095-9203. DOI: 10.1126/science.abe8770. 3