Streaming Zero-Knowledge Proofs

Graham Cormode
University of Warwick

Marcel Dall'Agnol
University of Warwick

Tom Gur*
University of Warwick

Chris Hickey
University of Manchester

Abstract

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 *sum-check* 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, frequency moments, and inner product. Our protocols are efficient in terms of time and space, as well as communication: the space complexity is 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.

*Tom Gur is supported by UKRI Future Leaders Fellowship MR/S031545/1 and EPSRC New Horizons Grant EP/X018180/1.

Contents

1 Introduction

Zero-knowledge interactive proofs are fundamental protocols in theoretical computer science, which have provided a highly fertile ground for investigations in complexity theory and cryptography (see, e.g., [Gol02, Vad07, Gol08] and references therein), and led to substantial practical applications [BSCG+13, BSCG+14, BSBHR18, BSCG+19, BSCR+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, Tha14, CCM+15, 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 that has sequential, one-pass access to a massive input as well as the prover's messages. We note that prover and verifier observe the same stream of bits, 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 this 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.

1.1 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 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 a *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 $\mathrm{poly}(s)$ amount of memory. (See Section 4 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. (And this restriction does not forbid streaming algorithms from taking superpolynomial time.) Therefore, the desired notion of indistinguishability in the data stream model requires small but constant bias. Our protocols achieve further robustness with a stronger $o(1)$ bound, ensuring that the probability of information leakage tends to zero.

The second crucial distinction is that the notion of zero-knowledge for SIPs is *information-theoretic*, faithfully to the nature of the data stream model. We prove our protocols secure *unconditionally*, without any computational assumptions. 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., [GKR08, CMT12]), whose zero-knowledge property is still with respect to the standard setting: the honest verifier is a streaming algorithm, but the protocols are only secure against polynomial-time adversaries. In this work, *adversaries are also streaming algorithms.*

This paper 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?

1.2 Main results

Our main contribution 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. In doing so, we obtain *exponentially* smaller space complexity for the fundamental INDEX and frequency moment problems 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 2.3), 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 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 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 7.2 and 7.3 for the formal version) in generality, but note that standard parameter settings imply space complexity $s = \text{polylog}(n)$ as well as $O(n^{1+\delta})$ communication (for any constant $\delta > 0$) and $n^{o(1)}$ 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 1.1 (Sumcheck zkSIP)**.** *For every m-variate low-degree polynomial over $\mathbb{F}$, there exists a sumcheck zkSIP where the verifier uses $s = O(m^2 \log |\mathbb{F}|)$ bits of space. The protocol communicates $\tilde{O}(|\mathbb{F}|^m)$ bits in its setup stage and $|\mathbb{F}|^{\log\log|\mathbb{F}|} \cdot \text{poly}(|\mathbb{F}|)$ in the interactive stage.*

The round complexity is $m + O(1)$, a small constant larger than that of the standard sumcheck protocol.

We stress that while the sumcheck protocol is traditionally used (in the polynomial-time setting) to verify exponentially large sums in polynomial time, this is *not* the streaming variant's goal, 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 6.2 and 6.3 for the formal statements.) It allows a streaming algorithm to access data that was already streamed 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 SIPs, solving key problems such as INDEX.

Theorem 1.2 (Polynomial evaluation zkSIP)**.** *For every m-variate low-degree polynomial over $\mathbb{F}$, there exists a polynomial evaluation zkSIP where the verifier uses $O(m \log |\mathbb{F}|)$ bits of space. The communication complexity is $\tilde{O}(q^m)$ in the setup and $\mathrm{poly}(q)$ in the interactive stage.*

As in Theorem 1.1, standard parameter settings imply zkSIPs with polylogarithmic space, $n^{o(1)}$ time and communication complexity (in the interactive stage)[1] as well as near-linear communication in the setup. The round complexity is $O(1)$.

1.2.1 Streaming commitment protocols

En route to proving Theorems 1.1 and 1.2, 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.[2]

Streaming algebraic commitment protocol. 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.

Theorem 1.3 (Theorem 5.8, 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 scheme communicates $\tilde{O}\left(|\mathbb{F}|^{3\ell}\right)$ bits.*

[1] A nontrivial security guarantee still holds with $\mathrm{polylog}(n)$ communication, but with $n^{o(1)}$ the protocol becomes secure against arbitrary $\mathrm{polylog}(n)$-space adversaries; see Remark 6.6.

[2] We refer to commitment *protocols* rather than schemes in the streaming model to avoid ambiguity, as they are not in exact correspondence; see Definition 5.1.

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

Theorem 1.4 (Theorem 5.17, informally stated)**.** *Let Γ be an alphabet and A a space-s streaming algorithm with $s = \text{polylog}\,|\Gamma|$. 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 $\alpha = z_i$.*

1.3 Applications

Recall that Theorems 1.1 and 1.2 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 of our tools by applying them to construct explicit zkSIPs for three streaming problems of fundamental importance: INDEX, FREQUENCY-MOMENTS and INNER-PRODUCT.

The following statements highlight space complexities, but (as mentioned in the previous section) all other parameters, including time and 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 1.5 (Corollary 6.5, informally stated)**.** *There exists a zkSIP for* INDEX *with logarithmic verifier space complexity.*

Note that this matches the space 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 $F_k(x) = \sum_{i \in [\ell]} f_i^k$, the k^{th} moment of the frequency vector $(f_1, \ldots, f_\ell)$, where f_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 1.6 (Corollary 7.5, informally stated)**.** *For every $\ell \in [n]$ and k, there exists a zkSIP that computes F_k with* $\text{polylog}(n)$ *verifier space complexity.*

Lastly, we illustrate the flexibility of our protocols by constructing an additional zkSIP for the INNER-PRODUCT problem; in it, an algorithm streams $x, y \in [\ell]^n$ and must compute the inner product between the frequency vectors of x and y.

Corollary 1.7 (Corollary 7.7, informally stated)**.** *For every $\ell \in [n]$, there exists a zkSIP for* INNER-PRODUCT *with* $\text{polylog}(n)$ *verifier space complexity.*

1.4 Related work

This work builds on the line of research on streaming interactive proofs, initiated by [CCM09] and actively investigated over the last decade [CMT13, CTY11, CMT12, CCGT14, Tha14, GR15,

CCM+15, DTV15, CH18, CGT20]. These sublinear interactive proofs are also closely related to proofs of proximity [RVW13, Gur17, GR17, GR18, GGR18, RR20, CG18, GG21, GLR21, DGMT22].

Indeed, our two main results can be seen as zero-knowledge versions of the main techniques in [CCM+15] and [CMT12]: respectively, a polynomial evaluation and a sumcheck protocol. (We note that while [CFGS22] construct a zero-knowledge sumcheck protocol via an algebraic commitment scheme, their model and techniques are completely different.)

Although past work has studied zero-knowledge protocols where the verifier is able to process incoming messages in a streaming fashion (e.g., [GKR08, CMT12]) their zero-knowledge property is still with respect to the standard (polynomial-time) setting; that is, while the honest verifier is a streaming algorithm, the security of the protocol holds against polynomial-time adversaries, whereas we consider adversaries that are also streaming algorithms.

We also note that while zero-knowledge proofs within sublinear models of computation have been actively explored in the last decade (e.g., [BRV18, IW14]), our work is the first to do so in the streaming model.

1.5 Open problems

This work opens several avenues for future research; in this section, we highlight two 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 polylog(n) space complexity, we raise the following problem.

Open problem 1. *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 polylog n) communication and time complexity. Making this parameter sublinear would be a major step towards practical applicability.

Open problem 2. *Can zero-knowledge SIPs achieve sublinear communication complexity?*

Organisation

The rest of the paper is organised as follows. In Section 2 we give a high-level overview of the challenges and the techniques we use to endow SIPs with zero-knowledge. We briefly discuss the preliminaries for the technical sections in Section 3, and, in Section 4, formally define the notion of streaming zero-knowledge and discuss key conceptual points. In Section 5 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 and 7, respectively.

2 Technical overview

We provide a high-level overview of the techniques we use and build upon in this paper. For concreteness, we illustrate our methodology by focusing 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 2.1, 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 2.2 and 2.3 via the construction of two types of *streaming commitment protocols*.

In Section 2.4, we apply the foregoing protocols to obtain a streaming interactive proof for INDEX and provide an overview of the proof of its zero-knowledge property, which requires an involved simulator argument. Finally, Section 2.5 sketches another application of this framework that obtains an additional powerful and flexible tool: a *zero-knowledge streaming sumcheck* protocol.

2.1 A starting point: the polynomial evaluation protocol

Recall that in the INDEX problem, a streaming algorithm with s bits of memory receives a length-n string x over an alphabet Γ, 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 save x_j) fails: both parties observe *the same* stream of information, 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.

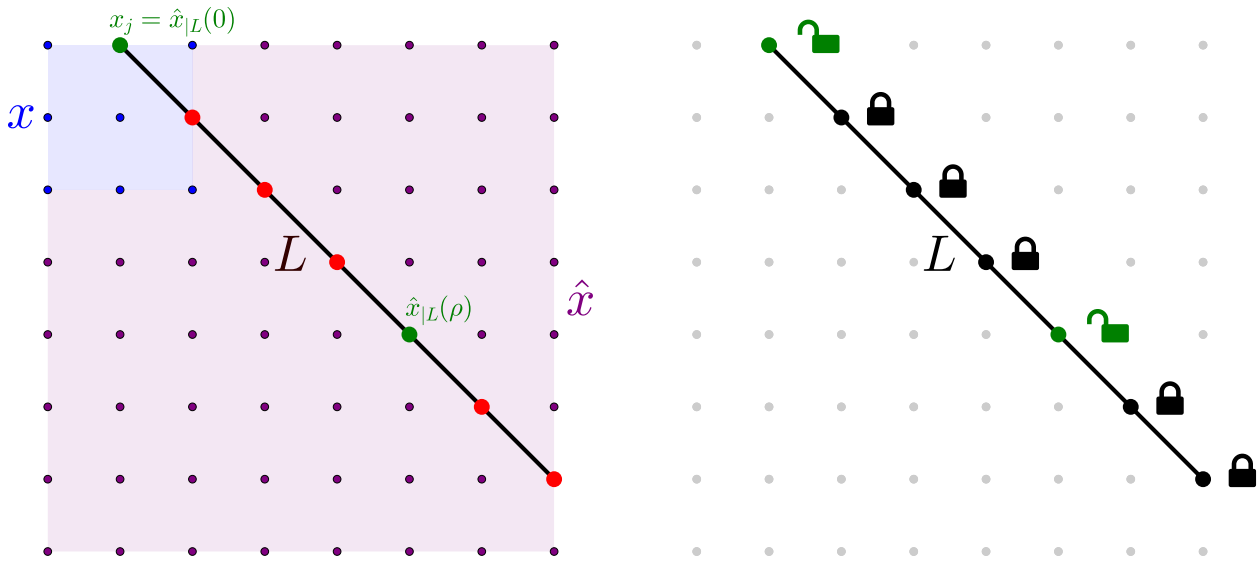
Remarkably, an exponential reduction in space complexity is possible despite both prover and verifier not knowing the index j before it appears in the stream. We recall the SIP in [CCM+15], upon which we build, and argue why it is *not* zero-knowledge to begin with. Their SIP is an application of pep, the *polynomial evaluation protocol*, 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; 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 $|\mathbb{F}| = \text{polylog}(n)$.)

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.

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 Fig. 1a). Note that learning the restriction



(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 1: 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.

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?

2.2 Curtailing leakage with commitments

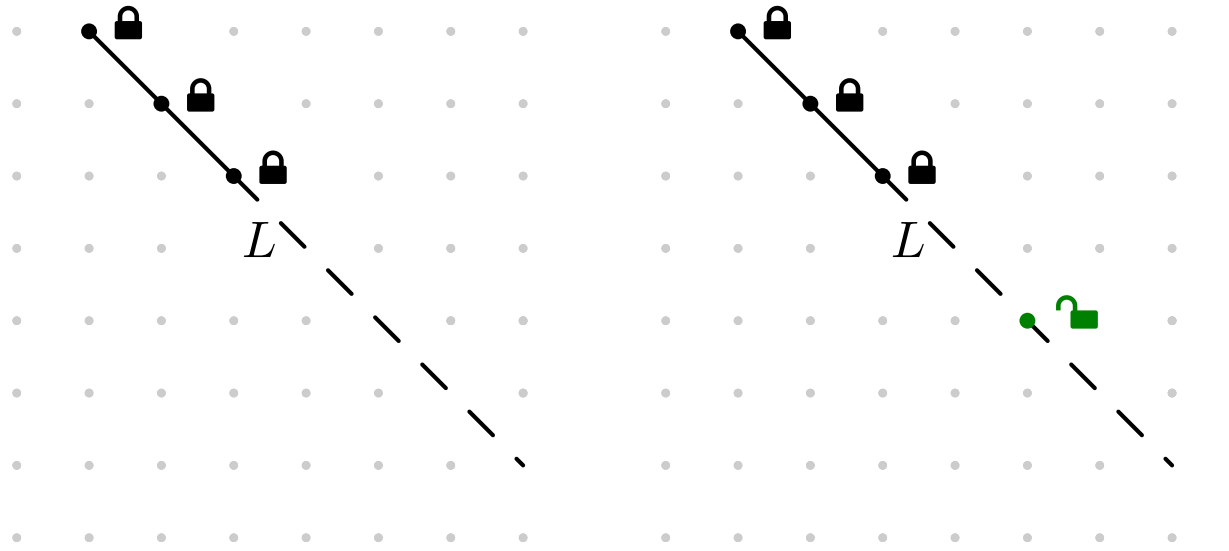
To make the foregoing approach more precise, let us first assume the existence of a *commitment protocol* that allows P to transmit any field element α to V in two steps: sending a string $\mathsf{commit}(\alpha)$, from which V is unable to extract any information about α; and later, upon the verifier's 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 Fig. 1b). 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 2.3. 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 Fig. 2). We next present the basic commitment protocol, and then extend it to be algebraic.



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

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

Figure 2: 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.

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 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 5.5 and Lemma 5.9).

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" $\gamma = \alpha - y_k$.[3] Of course, the discussion thus 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 5.2 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 $\boldsymbol{\alpha} = (\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_\ell)$ 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 5.1.) 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 2.1) 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 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 5.3 for details.

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

2.3 From honest to malicious verifiers: temporal commitments

Recall that a source of leakage in the INDEX protocol of Section 2.1 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 counterintuitive, 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 $\hat{x}$.[4] 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 5.4). 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.

2.4 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

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

Section 2.2 and a verifier-to-prover *temporal* commitment in Section 2.3. 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.

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 samples (another) evaluation point $\boldsymbol{\sigma}$ and computes the fingerprint $y(\boldsymbol{\sigma}, \boldsymbol{\beta}) = \sum_i \boldsymbol{\beta}_i \hat{y}_i(\boldsymbol{\sigma})$, where the tuple $\boldsymbol{\beta}$ satisfies $\sum_i \boldsymbol{\beta}_i \hat{x}_{|L}(i) = \hat{x}(\boldsymbol{\rho})$;[5] 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 $\boldsymbol{\beta}_i$).[6] V outputs the (alleged) x_j if the decommitment is consistent with $\hat{x}(\boldsymbol{\rho})$, and rejects otherwise.

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

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

[5] 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.

[6] 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.

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* $\boldsymbol{\rho} \in C$.[7] 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 4.

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 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 6.3 for details.

Remark 2.1 (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 5.9, 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*.

[7] 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 6.3.

2.5 A general-purpose zero-knowledge SIP: sumcheck

Lastly, we briefly mention how the commitment protocols developed in Sections 2.2 and 2.3 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 7 for details.

3 Preliminaries

General notation. For an integer $k \geq 1$, we denote by $[k]$ the set $\{1, 2, \ldots, k\}$. Vectors are denoted with notation analogous to that of sets, i.e., $(\alpha_i : i \in [k])$ denotes the vector $(\alpha_1, \ldots, \alpha_k)$. We use n to denote the length of a string that is the input to an algorithm, and $\text{poly}(n)$ (respectively, $\text{polylog}(n)$) to denote an arbitrary polynomial (respectively, polylogarithmic) function in n.

We use lowercase latin letters to denote positive integers (e.g., $d, i, j, k, \ell, m, p, v$) or strings (e.g., x, y, z); r and t often (but not always) denote random strings. Lowercase greek letters denote elements of a finite alphabet or field (e.g., α, β, γ), and we reserve ρ, σ for random elements. Uppercase letters denote either algorithms (e.g., A, B, P, S, V) or sets (e.g., C, K), with T used as the indeterminate of a polynomial.

When f and g are functions, we sometimes use $\alpha \in f$ as a shorthand for $\alpha \in \text{Im}\, f$ and $f_{|g}$ for $f \circ g$; if f is a low-degree polynomial that is communicated in an interactive protocol, we assume it is sent in a canonical form (e.g., a line is communicated by a pair of points $f(0), f(1)$). We use $\mathbb{1}[\cdot = x_0]$ to denote the delta function at x_0 (i.e., $\mathbb{1}[x_0 = x_0] = 1$ and $\mathbb{1}[x = x_0] = 0$ for $x \neq x_0$) and log to denote $\log_2$.

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

Vectors and matrices. The notation we use 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.

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., $\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma}$) and random tuples are (similarly to strings) 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.)

Probability. We use $X \sim \mu$ to denote a random variable with distribution μ, and, for the uniform distribution over a set S, we write $X \sim S$. 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 $\mathbb{P}_{X\sim\mu, Y\sim\lambda}[E]$. The internal randomness of an algorithm is generally omitted; 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 $\mathbb{P}_{\substack{X\sim\mu \\ r\sim\{0,1\}^m}}[A(X; r) = 0]$, where r is A's internal randomness.

We will also make use of the following versions of the Chernoff and Hoeffding bounds.

Lemma 3.1 (Additive Chernoff-Hoeffding bound)**.** *Let $X_1, \ldots, X_k$ be independent Bernoulli random variables distributed as X. Then, for every $\delta \in [0, 1]$,*

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

Lemma 3.2 (Hoeffding's inequality)**.** *Let $X_1, \ldots, X_k$ be independent random variables distributed as $X \in [a, b]$. Then, for every $\delta \in [0, 1]$,*

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

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 sumcheck protocols solve the PEP and SUMCHECK problems, respectively).

We generally use A, D, S and V to denote streaming algorithms, while P denotes an algorithm with unbounded computational resources (including space). $A(x)$ is the output of an algorithm that receives x as input; 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 $n \log |\Gamma|$.

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.

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).

The *snapshot* of an algorithm is synonymous to 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.

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 $i \in [k]$. 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., $\hat{y}(\boldsymbol{\alpha}, \boldsymbol{\beta}) = \sum_i \boldsymbol{\beta}_i \hat{y}_i(\boldsymbol{\alpha})$.

3.1 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

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

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

$$H(t) = H(1-t) = -t \log t - (1-t)\log(1-t). \tag{2}$$

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

$$-t(1+t) \leq \ln(1-t) \leq -t. \tag{3}$$

The (relative) *Hamming distance* between two strings $a, b \in \Gamma^k$ over a finite alphabet is the fraction of coordinates where they differ, i.e., $d(a, b) = \frac{1}{k}|\{i \in [k] : a_i \neq b_i\}| \in [0, 1]$. With $\gamma = |\Gamma|$,

the volume of a *Hamming ball* $\mathcal{B}(b,\delta) \coloneqq \{a \in \Gamma^k : d(a,b) \le \delta\}$ of radius $\delta = 1-\varepsilon$, when k is large enough and $\varepsilon = k^{-1}\operatorname{polylog}(k)$, satisfies

$$\gamma^{H_\gamma(\delta)k} \ge |\mathcal{B}(b,\delta)| = \Omega\left(\frac{\gamma^{H_\gamma(\delta)k}}{\sqrt{\varepsilon k}}\right) = \frac{\gamma^{H_\gamma(\delta)k}}{\operatorname{polylog}(k)}.^{8} \tag{4}$$

The entropy of a discrete random variable X taking values in Γ is $H(X) = \sum_{\alpha\in\Gamma} H\big(\mathbb{P}[X=\alpha]\big)$. Every such random variable satisfies

$$H(X) \in [0, \log|\Gamma|]. \tag{5}$$

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

$$H(X|Y) \le H(X). \tag{6}$$

If X, Y are independent, then

$$H(X,Y) = H(X) + H(Y). \tag{7}$$

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

$$H(X_1,\dots,X_n) = \sum_{i=1}^{n} H(X_i|X_1,\dots,X_{i-1}). \tag{8}$$

For ease of notation, when $X \sim \mu$, $Y \sim \lambda$ and both distributions are over the space Γ, we denote the distribution of B conditioned on $A = a$ as λ_a. The *KL divergence* is

$$\mathrm{KL}(\mu \mid\mid \lambda) = \sum_{x\in\Gamma} \mu(x)\log\frac{\mu(x)}{\lambda(x)}, \tag{9}$$

which upper bounds the Euclidean distance between probability vectors via *Pinsker's inequality* (see, e.g., [BLM13]):

$$\|\mu-\lambda\|^2 \le \frac{\mathrm{KL}\big(\mu \mid\mid \lambda\big)}{2\ln 2}. \tag{10}$$

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

$$\begin{aligned} I(\mu:\lambda) &\coloneqq I(A:B) \\ &= I(B:A) \\ &= H(B) - H(B|A) \\ &= \mathbb{E}_{A\sim\mu}[\mathrm{KL}(\lambda_A \mid\mid \lambda)]. \end{aligned} \tag{11}$$

[8] The lower bound is a simplification of

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

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].)

4 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 4.1.

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.[9] To this end, we first recall the formalisation of *streaming interactive proofs* (SIPs) [CTY11] without any zero-knowledge requirement.

Definition 4.1. 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.[10] 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 paper) allow for improving soundness by a desired factor with a logarithmic increase to their space complexity (see Section 5.1). We stress

[9] We stress that this is markedly different from zero-knowledge interactive proofs that reduce to evaluating low-degree polynomials defined by the input and thus allow for it to be processed in a streaming fashion, such as [GKR08]. Such protocols are secure in the standard sense, i.e., against polynomial-time adversaries, whereas our protocols are secure against *streaming* adversaries (and do not rely on computational assumptions).

[10] 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.

that Definition 4.1 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 4.2. 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 $\text{View}_{P,\widetilde{V}}(x, r)$, is the random variable defined as

$$\text{View}_{P,\widetilde{V}}(x, r; t) = (r, y_1, \ldots, y_{k_1}, x, y_{k_1+1}, \ldots, y_{k_1+k_2}).^{11}$$

While Definition 4.2 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 $\text{View}_{P,\widetilde{V}}(x, r)$ 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).

4.1 Definition

We now ready to give a formal definition of zero-knowledge streaming interactive proofs.

Definition 4.3 (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 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 $\text{View}_{P,\widetilde{V}}(x, r)$ and $S(\widetilde{V}, x, r)$ are indistinguishable by any streaming space-s' algorithm. That is, for every space-s' streaming algorithm D,

$$\left| \mathbb{P}\left[D\big(\text{View}_{P,\widetilde{V}}(x, r)\big) \text{ accepts} \right] - \mathbb{P}\left[D\big(S(\widetilde{V}, x, r)\big) \text{ accepts} \right] \right| = o(1).$$

We note that all our applications have $s = \text{polylog}(n)$, and the protocols are secure against adversaries with any space $s' = \text{poly}(s)$ (see Remark 6.6).

[11] 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.

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 4.4. 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,

$$\mathcal{W}(b,z) = \max_{y \in \{0,1\}^n} \{\mathbb{P}[A(y) \text{ outputs } z \text{ when its initial snapshot is } b]\}\,.$$

Remark 4.5. 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.

5 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 as they are not in direct correspondence; this is the case in particular with respect to the hiding property in the definition that follows.)

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 5.1).

We first formally define streaming commitment protocols. We note that while the definition that follows can be generalised,[12] it suffices to capture our constructions.

Definition 5.1. 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 $K \subseteq \{0,1\}^p$ 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 $\alpha, \beta \in \Gamma$. Sample $k, k' \sim K$ then 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:* For every $\alpha \in \Gamma$ and $k \in K$,

$$\mathbb{P}\big[\langle P, V\rangle\big(\mathsf{commit}(\alpha, k), \alpha\big) = 1\big] = 1,$$

  whereas when $\beta \neq \alpha$ and $k \sim K$,

$$\mathbb{P}\big[\langle P, V\rangle\big(\mathsf{commit}(\alpha, k), \beta\big) = 1\big] = o(1).$$

Note that, with some abuse of notation, the binding condition corresponds to (P, V) being an SIP for the language $L = \{(\mathsf{commit}(\alpha, k), \alpha) : \alpha \in \Gamma, k \in K\}$ (where the input is generated by the prover and soundness is "average-case", with respect to a random key).

The next sections introduce the commitment protocols we will use to build our protocols. Section 5.1 begins by defining the concepts and tools we build upon: low-degree extensions and the polynomial evaluation protocol (pep). In Section 5.2, 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 5.3 then extends the construction of Section 5.2 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 5.4 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.

[12] A natural generalisation is to parametrise 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 $\varepsilon_c = 0$.

5.1 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.

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 $n \le (d+1)^m$, 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 $(i_1, \dots, i_m)$ as the image of i by $[n] \hookrightarrow \mathbb{F}^m$,

$$\hat{x} = \sum_{i=1}^{n} x_i \chi_i = \sum_{i_1,\dots,i_m \in [d+1]} x_{i_1,\dots,i_m} \chi_{i_1,\dots,i_m}$$

where the χ_i are the Lagrange basis polynomials, given by

$$\chi_i(\alpha_1, \dots, \alpha_m) \coloneqq \prod_{j=1}^{m} \prod_{\substack{k=1 \\ k \neq i_j}}^{d+1} \frac{\alpha_j - k}{i_j - k}$$

(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 $0 \le i, j \le d$. We will use the charactersiation 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 $\big(\chi_1(\boldsymbol{\alpha}), \dots, \chi_n(\boldsymbol{\alpha})\big)$ 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 5.2 (Schwartz-Zippel)**.** *If $x, y \in \mathbb{F}_q^n$ are distinct, then $\mathbb{P}_{\boldsymbol{\rho} \sim \mathbb{F}^m}\big[\hat{x}(\boldsymbol{\rho}) = \hat{y}(\boldsymbol{\rho})\big] \le dm/q$.*

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 5.1, 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+15]). 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 $\textsc{pep}(f, \alpha)$ 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 5.1: $\mathsf{pep}(f, \alpha)$

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 $f^x_{|L}$.[13]

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 $g(0) = \alpha$.[14] 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 5.1 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 $\mathsf{pep}(f, \alpha)$ 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 5.1 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 5.1 yields a simple and efficient streaming interactive proof for 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 $q = |\mathbb{F}| = \text{polylog}(n)$ (and $\boldsymbol{\beta} \in \mathbb{F}^m$ is the identification of j); it is simply an instantiation of pep where $d = 2$, $m = \log n$ and the function $f^x = \hat{x}$ is the m-variate (multilinear) LDE of x,[15] 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 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

[13] 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 $(f^x \circ L(i) : i \in [dm])$. (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.)

[14] Note that the Lagrange polynomials in this case satisfy $\chi_i(j) = \mathbb{1}[i = j]$ for all $0 \leq i, j \leq dm$.

[15] The space complexity can be reduced to $O(\log n)$ with the choice of parameters for q, d and m in Corollary 6.5.

points, without sacrificing soundness.

5.2 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 INDEX problem. By sending a message hidden at a random coordinate, we exploit the fact that any 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 5.3. 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 5.4. 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 5.5. *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 provide a proof of this fact in Appendix A.1 for completeness).

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 5.2), while the decommit stage simply instantiates pep appropriately (see Protocol 5.3). We note that the inputs listed in the description of the protocols are those available to the verifier.

Protocol 5.2: commit(α)

Input: explicit access to $p, d, m, q \in \mathbb{N}$ with $p \le d^m$, $q > d$ 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 5.3: 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 5.2 and 5.3 form a streaming commitment protocol, i.e., they satisfy the hiding and binding properties of Definition 5.1 if p is large enough; these follow from the hardness of SEARCH-INDEX and the soundness of pep, respectively.

Theorem 5.6. *Protocols 5.2 and 5.3 form a streaming commitment protocol with space complexity $s = O(m \log q)$ when $p = q^3$ and $dm = \text{polylog}(q)$. The protocol is secure against $\text{poly}(s)$-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 5.2) implies $\hat{y}(\boldsymbol{\rho}) = \hat{y}_{|L}(\rho) \neq g(\rho)$ except with probability $dm/q = o(1)$, in which case V rejects.[16] 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 $\text{poly}(s) = \text{polylog}(q)$ that distinguishes commitments between some $\alpha \in \mathbb{F}$ and $\alpha' \in \mathbb{F} \setminus \{\alpha\}$ with constant bias:[17] that is,

$$\mathbb{P}_{\substack{y \sim \mathbb{F}^p \\ k \sim [p]}} \left[D(y, k, \alpha - y_k) \text{ accepts} \right] - \mathbb{P}_{\substack{y \sim \mathbb{F}^p \\ k \sim [p]}} \left[D(y, k, \alpha' - y_k) \text{ accepts} \right] \geq \varepsilon$$

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

[16] 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.

[17] Note that allowing $\text{poly}(s)$ 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.

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 5.5.

$$\begin{aligned}
\mathbb{P}_{\substack{x\sim\mathbb{F}^p \\ j\sim[p]}}\left[A(x,j) = x_j\right] &= \frac{1}{q} \cdot \mathbb{P}_{\substack{x\sim\mathbb{F}^p \\ j\sim[p]}}\left[D(x,j,\alpha - x_j) \text{ accepts}\right] + \frac{1}{q} \cdot \mathbb{P}_{\substack{x\sim\mathbb{F}^p \\ j\sim[p]}}\left[D(x,j,\alpha' - x_j) \text{ rejects}\right] \\
&= \frac{1}{q}\left(1 + \mathbb{P}_{\substack{x\sim\mathbb{F}^p \\ j\sim[p]}}\left[D(x,j,\alpha - x_j) \text{ accepts}\right] - \mathbb{P}_{\substack{x\sim\mathbb{F}^p \\ j\sim[p]}}\left[D(x,j,\alpha' - x_j) \text{ accepts}\right]\right) \\
&\geq \frac{1+\varepsilon}{q} \\
&= \frac{1}{q} + \Omega\left(\frac{1}{q}\right).
\end{aligned}$$

Since $1/q = \sqrt{q/p} = \omega\left(\sqrt{\mathrm{poly}(s)/p}\right)$, owing to $s = \mathrm{polylog}(q)$, the result follows. □

Remark 5.7. 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 4.3. (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 $\hat{y}_{|L} = \left(\hat{y}_{|L}(i) : i \in \{0\} \cup [dm]\right)$ 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 2.2, 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*.

5.3 Making the commitment algebraic

In this section, we will show how to modify the commitment protocol laid out in Section 5.2 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 $(y_1, \ldots, y_\ell, \boldsymbol{\gamma}, k)$ 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.[18] 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 5.4, which enables a prover to commit to multiple messages and decommit (via Protocol 5.3, 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 5.8. *Protocol 5.4 (algebraic-commit) and Protocol 5.3 (decommit) form a streaming commitment protocol with space complexity $s = O\big((\ell + m) \log q\big)$ if $p = q^{3\ell}$ and $dm = \text{polylog}(q)$. The scheme is secure against $\text{poly}(s)$-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 $s = O(m \log q)$.

Since the proof is a straightforward extension of Theorem 5.6, we defer it to Appendix A.2.

Protocol 5.4: algebraic-commit($\boldsymbol{\alpha}$)

Input: explicit access to $p, m, d, q \in \mathbb{N}$ with $p \leq d^m$, $q > d$ 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 $\gamma = \sum_{i=1}^{\ell} \boldsymbol{\beta}_i \boldsymbol{\gamma}_i$.

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 $\boldsymbol{\alpha}' - \boldsymbol{\alpha}$.

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

[18] 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.

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 6.2).

However, an important issue still remains: the exponential dependency of Theorem 5.8 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 $\{0,1\}^\ell$ from a commitment to a random one.[19]

Lemma 5.9. *Let (A, B) be a one-way protocol with s-bit messages that distinguishes between a length-p algebraic commitment to a fixed $\boldsymbol{\alpha} \in \{0,1\}^\ell$ and a random commitment with advantage ε; that is, such that*

$$\left| \mathbb{P}_{\substack{y\sim\{0,1\}^{\ell\times p} \\ k\sim[p]}} \left[B\big(A(y), (\boldsymbol{\alpha}_i \oplus y_{ik} : i \in [\ell]), k\big) \text{ accepts} \right] - \mathbb{P}_{\substack{y\sim\{0,1\}^{\ell\times p} \\ k\sim[p] \\ \boldsymbol{\tau}\sim\{0,1\}^\ell}} \left[B(A(y), \boldsymbol{\tau}, k) \text{ accepts} \right] \right| = \varepsilon.$$

Then there exists an average-case one-way communication protocol for (binary) INDEX *over p-bit strings that communicates $64\ell^2 s/\varepsilon^2$ bits and succeeds with probability $1 - \frac{1}{e} = \frac{1}{2} + \Omega(1)$.*

Proof. Define, for ease of notation, $y^{(k)} \coloneqq (y_{ik} : i \in [\ell])$ (i.e., the k^{th} column of y) and

$$a_{\boldsymbol{\tau}} \coloneqq \mathbb{P}\left[B\left(A(y), \boldsymbol{\tau} \oplus y^{(k)}, k\right) \text{ accepts} \right] = \mathbb{E}\left[B\left(A(y), \boldsymbol{\tau} \oplus y^{(k)}, k\right) \right],$$

where we interpret Bob's output as 1 (respectively 0) when it accepts (respectively rejects). Define, also, $\varepsilon_{\boldsymbol{\tau}} \coloneqq a_{\boldsymbol{\alpha}} - a_{\boldsymbol{\tau}}$ and note that[20]

$$\varepsilon = a_{\boldsymbol{\alpha}} - \frac{1}{2^\ell} \sum_{\boldsymbol{\tau}\in\{0,1\}^\ell} a_{\boldsymbol{\tau}} = \frac{1}{2^\ell} \sum_{\boldsymbol{\tau}\in\{0,1\}^\ell} \varepsilon_{\boldsymbol{\tau}}.$$

Finally, define, for each $0 \le i < \ell$,

$$\varepsilon_i \coloneqq \frac{1}{2^{\ell-i}} \sum_{\substack{\boldsymbol{\tau}\in\{0,1\}^\ell \\ \forall i' \le i,\ \boldsymbol{\tau}_{i'} = \boldsymbol{\alpha}_{i'}}} \varepsilon_{\boldsymbol{\tau}}.$$

We divide the analysis into two cases: suppose, first, that $\varepsilon_i \ge \varepsilon_{i-1} \cdot \left(1 - \frac{1}{2\ell}\right)$ for all $i \in [\ell - 1]$. Then, by Bernoulli's inequality ($(1+t)^\ell \ge 1 + t\ell$ when $t \le -1$), we have

$$\varepsilon_{\ell-1} = \frac{1}{2}\left(a_{\boldsymbol{\alpha}} - a_{\boldsymbol{\alpha}^{\oplus\ell}}\right) \ge \left(1 - \frac{1}{2\ell}\right)^\ell \varepsilon \ge \frac{\varepsilon}{2},$$

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 $2/\varepsilon^2$

[19] Note that while this parameter setting suggests $\mathbb{F} = \mathbb{F}_2$, we may also take tuples over $\mathbb{F}^{\ell'}$ for a field extension of the binary field, and apply the reduction with $\ell = \ell' \cdot \log|\mathbb{F}|$.

[20] This assumes the acceptance probability of a commitment to a fixed message is larger than that of a random commitment, which is without loss of generality (otherwise Bob can simply flip his output bit).

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 $\sigma(x)_k \coloneqq x_{\sigma(k)}$), 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,

$$\mathbb{P}\left[\mu \leq a_{\boldsymbol{\alpha}} - \frac{\varepsilon}{2}\right] \leq \frac{1}{e}$$

by the Chernoff-Hoeffding bound (Lemma 3.1, 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. (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).

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 of

$$\frac{1}{2^{\ell-i}} \left(\sum_{\substack{\boldsymbol{\tau} \in \{0,1\}^\ell \\ \forall i' \leq i,\ \boldsymbol{\tau}_{i'} = \boldsymbol{\alpha}_{i'}}} \left(B\left(A(y), \boldsymbol{\tau}^{\oplus i} \oplus y'^{(k)}, k\right) - B\left(A(y), \boldsymbol{\tau} \oplus y'^{(k)}, k\right) \right) \right),$$

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 $x_j = 0$,

$$\mathbb{E}[\mu] = \frac{1}{2^{\ell-i}} \left(\sum_{\substack{\boldsymbol{\tau} \in \{0,1\}^\ell \\ \forall i' \leq i,\ \boldsymbol{\tau}_{i'} = \boldsymbol{\alpha}_{i'}}} (a_{\boldsymbol{\tau}^{\oplus i}} - a_{\boldsymbol{\tau}}) \right) > \frac{\varepsilon}{2\ell},$$

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 3.2, with $a = -1$, $b = 1$, $\delta = 1/2$ and $64\ell^2/\varepsilon^2$ samples): in the $x_j = 0$ case,

$$\mathbb{P}\left[\mu \leq \frac{\varepsilon}{4\ell}\right] \leq \frac{1}{e};$$

and, likewise, in the $x_j = 1$ case we have $\mathbb{P}\left[\mu \geq -\frac{\varepsilon}{4\ell}\right] \leq \frac{1}{e}$. □

5.4 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 5.1), 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 5.10 and 5.11). 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 5.12, that if Alice's message has s bits, Bob cannot reconstruct significantly more than s coordinates of the input. Then, in Lemma 5.13, 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 5.13 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 5.13.

To this end we define, for each fixed string $x \in \Gamma^n$, a variant of PAIR that we call PAIR(x) (Definition 5.15). 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 5.17, 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 5.10. 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 5.11. 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 5.12. *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 $b \in B \coloneqq \{b(a) : a \in A\}$, 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 $M_b \coloneqq \{i \in [v] : z_i = b_i\}$. For simplicity of notation, denote also

$\gamma := |\Gamma|$. Note that the expected score of this one-way protocol is

$$\begin{aligned}\mathbb{E}_{z\sim\Gamma^v}\left[\sum_{b\in B}\mathbb{1}[z\in P_b]\cdot|M_b|\right] &= \sum_{b\in B}\mathbb{P}[z\in P_b]\cdot\mathbb{E}_{z\sim P_b}\left[|M_b|\right]\\ &= \sum_{\substack{b\in B\\ |P_b|\geq\frac{s}{v}\cdot\frac{\gamma^v}{2^s}}}\mathbb{P}[z\in P_b]\cdot\mathbb{E}_{z\sim P_b}\left[|M_b|\right] + \sum_{\substack{b\in B\\ |P_b|<\frac{s}{v}\cdot\frac{\gamma^v}{2^s}}}\mathbb{P}[z\in P_b]\cdot\mathbb{E}_{z\sim P_b}\left[|M_b|\right].\end{aligned}$$

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:

$$\begin{aligned}\mathbb{E}_{z\sim\Gamma^v}\left[\sum_{b\in B}\mathbb{1}[z\in P_b]\cdot|M_b|\right] &\leq \max_{\substack{b\in B\\ |P_b|\geq\frac{s\gamma^v}{v\cdot 2^s}}}\mathbb{E}_{z\sim P_b}\left[|M_b|\right] + \sum_{\substack{b\in B\\ |P_b|<\frac{s\gamma^v}{v\cdot 2^s}}}\mathbb{P}[z\in P_b]\cdot v\\ &\leq \max_{\substack{b\in B\\ |P_b|\geq\frac{s\gamma^v}{v\cdot 2^s}}}\mathbb{E}_{z\sim P_b}\left[|M_b|\right] + s.\end{aligned}$$

Let $\delta\in(0,1)$ be such that the volume of Hamming balls of radius δ is $\mathcal{V} := \frac{s\gamma^v}{v\cdot 2^s}\leq\frac{s\gamma^v}{v^2}$. (Recall that $s\geq\log v$.) 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 $P_b = \mathcal{B}(b,\delta')$, the ball centered at b (whose radius δ' is determined by the equality $|\mathcal{B}(b,\delta')| = |P_b|$). Since $|P_b|\geq\mathcal{V}$ implies $\delta'\geq\delta$, we have

$$\frac{1}{|P_b|}\cdot\sum_{z\in\mathcal{B}(b,\delta')}|\{i\in[v] : z_i = b_i\}| \leq \frac{1}{\mathcal{V}}\cdot\sum_{z\in\mathcal{B}(b,\delta)}|\{i\in[v] : z_i = b_i\}|,$$

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 $z\sim\mathcal{B}(b,\delta')\setminus\mathcal{B}(b,\delta)$, 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:

$$H(\sigma) = H(1-\sigma)\in\left[\sigma\log\frac{1}{\sigma}, \sigma\left(\log\frac{1}{\sigma}+\frac{2}{\ln 2}\right)\right] \tag{12}$$

We begin with the lower bound on ε, which uses the lower bound of Eq. 12 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

$$
\begin{aligned}
&H_\gamma\left(1-\frac{\log\gamma}{v\log\log\gamma}\right)\\
&=\frac{\left(1-\frac{\log\gamma}{v\log\log\gamma}\right)\log(\gamma-1)+H\left(1-\frac{\log\gamma}{v\log\log\gamma}\right)}{\log\gamma} && \text{(by Eq. 1)}\\
&=\frac{\left(1-\frac{\log\gamma}{v\log\log\gamma}\right)\log(\gamma-1)+H\left(\frac{\log\gamma}{v\log\log\gamma}\right)}{\log\gamma} && \text{(by Eq. 2)}\\
&\geq\frac{\left(1-\frac{\log\gamma}{v\log\log\gamma}\right)\left(\log\gamma+\log\left(1-\frac{1}{\gamma}\right)\right)}{\log\gamma}+\frac{\log\left(\frac{v\log\log\gamma}{\log\gamma}\right)}{v\log\log\gamma} && \text{(by Eq. 12)}\\
&=1+\left(\frac{1}{\log\gamma}-\frac{1}{v\log\log\gamma}\right)\log\left(1-\frac{1}{\gamma}\right)+\frac{\log\frac{v}{\gamma}+\log\log\log\gamma}{v\log\log\gamma}-\frac{1}{v}\\
&\geq 1-\frac{1}{\gamma\ln 2}\left(1+\frac{1}{\gamma}\right)\left(\frac{1}{\log\gamma}-\frac{1}{v\log\log\gamma}\right)+\frac{\log\frac{v}{\gamma}+\log\log\log\gamma}{v\log\log\gamma}-\frac{1}{v} && \text{(by Eq. 3)}\\
&\geq 1-\frac{1}{\gamma\ln 2}\left(1+\frac{1}{\gamma}\right)\left(\frac{1}{\log\gamma}-\frac{1}{v\log\log\gamma}\right)-\frac{1}{v}\\
&\geq 1-\frac{3}{2v},
\end{aligned}
$$

where the second-to-last inequality uses $v\geq\gamma$; and the last inequality 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. 4, the volume of a ball (centered at any point b) of radius $1-\frac{\log\gamma}{v\log\log\gamma}=1-\frac{\text{polylog}(v)}{v}$ satisfies

$$
\begin{aligned}
\left|\mathcal{B}\left(b,1-\frac{\log\gamma}{v\log\log\gamma}\right)\right|&\geq\frac{\gamma^{H_\gamma\left(1-\frac{\log\gamma}{v\log\log\gamma}\right)v}}{\sqrt{\log v}}\\
&\geq\frac{\gamma^v}{2^{\frac{3}{2}\log\gamma+\frac{1}{2}\log\log v}}\\
&\geq\frac{\gamma^v}{2^{\frac{7}{4}\log\gamma}}.
\end{aligned}
$$

Then

$$
\begin{aligned}
|\mathcal{B}(b,\delta)|=\mathcal{V}&=\frac{s\gamma^v}{v\cdot 2^s}\\
&\leq\frac{\gamma^v\,\text{polylog}(v)}{v^2}\\
&\leq\frac{\gamma^v}{2^{\frac{15}{8}\log v}}\\
&\leq\left|\mathcal{B}\left(b,1-\frac{\log\gamma}{v\log\log\gamma}\right)\right|,
\end{aligned}
$$

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. 12. Since $\gamma^{H_\gamma(\delta)v} \geq \mathcal{V} = \frac{s\gamma^v}{v \cdot 2^s}$ (Eq. 4), taking the logarithm of both sides and using Eq. 1 yields

$$\frac{(1-\varepsilon)\log(\gamma-1) + H(1-\varepsilon)}{\log \gamma} = H_\gamma(1-\varepsilon) = H_\gamma(\delta) \geq 1 - \frac{s + \log \frac{v}{s}}{v \log \gamma}. \tag{13}$$

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. 12, which, combined with Eq. 13 (multiplied by $\log \gamma$), implies

$$(1-\varepsilon)\log(\gamma-1) + \varepsilon \log \frac{1}{\varepsilon} + \frac{2\varepsilon}{\ln 2} \geq \log \gamma - \frac{s + \log v - \log s}{v}.$$

Rearranging yields

$$\varepsilon \left(\log \varepsilon + \log \gamma + \log\left(1 - \frac{1}{\gamma}\right) - \frac{2}{\ln 2} \right) \leq \frac{s + \log v - \log s}{v} + \log\left(1 - \frac{1}{\gamma}\right).$$

The bounds $-\log(1 - 1/\gamma) = O(1/\gamma) = O(\log\log v / v)$ (Eq. 3) and $s \geq \log v$ show that the right-hand side is $O(s/v)$; and Eq. 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 $\mathrm{polylog}(v)/v$).

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 $\varepsilon = v^{-1}\,\mathrm{polylog}(v)$, Eq. 4 applies, giving $\mathcal{V}' \leq \gamma^{H_\gamma(1-2\varepsilon)}$ 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:

$$\begin{aligned} H_\gamma(1-\varepsilon) - H_\gamma(1-2\varepsilon) &= \frac{\varepsilon\log(\gamma-1) + H(\varepsilon) - H(2\varepsilon)}{\log\gamma} \\ &\geq \frac{\varepsilon}{\log\gamma}\left(\log(\gamma-1) + \log\frac{1}{\varepsilon} - 2\log\frac{1}{2\varepsilon} - \frac{4}{\ln 2}\right) && \text{(by Eq. 12)} \\ &= \frac{\varepsilon}{\log\gamma}\left(\log(\varepsilon\gamma) + \log\left(1 - \frac{1}{\gamma}\right) + 2 - \frac{4}{\ln 2}\right) \\ &\geq \frac{\varepsilon \log\log\gamma}{\log\gamma}, \end{aligned}$$

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 ($\gamma \geq 32v/\log\log v$ suffices, as $\log(1-1/\gamma)+2-4/\ln 2 > -5$). Therefore,

$$\begin{aligned}
\sqrt{s}\cdot\gamma^{-(H_\gamma(1-\varepsilon)-H_\gamma(1-2\varepsilon))v} &\leq \sqrt{s}\cdot\gamma^{-\frac{\varepsilon v\log\log\gamma}{\log\gamma}} \\
&= \sqrt{s}\cdot 2^{-\varepsilon v\log\log\gamma} \\
&< \sqrt{s}\cdot 2^{-\log\gamma} \\
&= \frac{\sqrt{s}}{\gamma} \\
&= \Theta\left(\frac{\sqrt{s}\log\log v}{v}\right) \\
&= o(s/v),
\end{aligned}$$

where the last line is due to $\sqrt{s} \geq \sqrt{\log v} = \omega(\log\log v)$ 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

$$\begin{aligned}
\mathbb{E}_{z\sim\Gamma^v}\left[\sum_{b\in B}\mathbb{1}[z\in P_b]\cdot|M_b|\right] &\leq \max_{\substack{b\in B\\ |P_b|\geq\frac{s\gamma^v}{v\cdot 2^s}}}\mathbb{E}_{z\sim P_b}\left[|M_b|\right]+s \\
&\leq s+\frac{1}{\mathcal{V}}\cdot\sum_{z\in\mathcal{B}(b,\delta)}|\{i\in[v]:z_i=b_i\}| \\
&\leq s+\frac{\mathcal{V}'}{\mathcal{V}}\cdot v+\left(1-\frac{\mathcal{V}'}{\mathcal{V}}\right)\cdot 2\varepsilon v \\
&= s+o(s),
\end{aligned}$$

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 5.13. *Any one-way protocol for* PAIR *with alphabet size* $\frac{32v}{\log\log v} \leq |\Gamma| = O\left(\frac{v}{\log\log v}\right)$ *and message length* s*, where* $\log v \leq s = \mathrm{polylog}(v)$*, 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*

$$\mathbb{P}\big[\textit{Bob outputs } (z_i,i) \textit{ with } i\notin C\big|E\big] = o(1).$$

Proof. We will first show how to construct a protocol for RECONSTRUCT given one for PAIR, and then use Lemma 5.12 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 $b(a) \in \Gamma \times [v]$;[21] 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

$$\begin{aligned}\sum_{i=1}^{v} \mathbb{P}_{z\sim P_a}[b_1 = z_i \mid b_2 = i] &= \sum_{i=1}^{v} \mathbb{P}_{\substack{z\sim P_a \\ b_1\sim\mu_i}}[b_1 = z_i] \\ &= \sum_{i=1}^{v} \sum_{\alpha\in\Gamma} \mathbb{P}_{b_1\sim\mu_i}[b_1 = \alpha] \cdot \mathbb{P}_{z\sim P_a}[\alpha = z_i] \\ &\le \sum_{i=1}^{v} \mathbb{P}_{z\sim P_a}[y_i = z_i] \\ &= \mathbb{E}_{z\sim P_a}\left[|M_a|\right],\end{aligned}$$

where, as before, $M_a = \{i \in [v] : y_i = z_i\}$.

Recall that in Lemma 5.12 we showed that, as long as $|P_a| \ge \frac{s|\Gamma|^v}{v2^s}$, the above expectation is $o(s)$. To conclude, we will use the following claim, whose proof is deferred to Appendix A.3:

Claim 5.14. *Let $p, q \in [0,1]^v$ be probability vectors and $t \le v$ be an integer. There exists a set $C \subseteq [v]$ of size t such that $\sum_{i\in[v]\setminus C} p_i q_i \le 1/t$.*

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 5.14 to $p = \left(\mathbb{P}[b_2 = i] : i \in [v]\right)$, $q = r/\|r\|_1$ and $t = s$,

[21] 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.

we obtain a set $C_a \subset [v]$ of size s such that

$$\begin{aligned}\mathbb{P}_{\substack{z\sim P_a\\ b\sim\mu(a)}}\big[b = (z_i, i) \text{ with } i \notin C_a\big] &= \sum_{i\notin C_a}^{v} p_i r_i \\ &= \|r\|_1 \sum_{i\notin C_a}^{v} p_i q_i \\ &\le \frac{\|r\|_1}{s} \\ &= \frac{\sum_{i=1}^{v} \mathbb{P}[b_1 = z_i \mid b_2 = i]}{s} \\ &= o(1)\end{aligned}$$

whenever $|P_a| \ge \frac{s|\Gamma|^v}{v2^s}$. 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

$$\begin{aligned}&\mathbb{P}\left[b(a) = (z_i, i) \text{ with } i \notin C_a \;\middle|\; |P_a| \ge \frac{s|\Gamma|^v}{v2^s}\right] \\ &= \frac{1}{\mathbb{P}_{z\sim\Gamma^v}\left[|P_a| \ge \frac{s|\Gamma|^v}{v2^s}\right]} \sum_{\substack{a\in A\\ |P_a|\ge\frac{s|\Gamma|^v}{v2^s}}} \mathbb{P}_{z\sim\Gamma^v}[z \in P_a] \cdot \mathbb{P}_{\substack{z\sim P_a\\ b\sim\mu(a)}}[b = (z_i, i) \text{ with } i \notin C_a] \\ &= \frac{1}{1 - o(1)} \cdot o(1) = o(1),\end{aligned}$$

which concludes the proof. □

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 5.13 carries over to it.

Definition 5.15. 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 5.16. *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 5.13 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 5.13, $\{P_a\}$ 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

$$r_i \coloneqq \max_{x\in\Gamma^n} \left\{ \mathbb{P}_{\substack{z\sim P_a \\ b_1\sim\mu_i(a,x)}}[b_1 = z_i] \right\} \le \mathbb{P}_{z\sim P_a}[y_i = z_i]$$

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 5.14, we obtain a set $C_a \subset [v]$ of size s such that *for every* $x \in \Gamma^n$,

$$\begin{aligned}\mathbb{P}_{\substack{z\sim P_a \\ b\sim\mu(a,x)}}\left[b = (z_i, i) \text{ and } i \notin C_a\right] &= \sum_{i=1}^{v} \mathbb{P}_{b\sim\mu(a,x)}[b_2 = i] \cdot \mathbb{P}_{\substack{z\sim P_a \\ b_1\sim\mu_i(a,x)}}[b_1 = z_i] \\ &\le \|r\|_1 \sum_{i\notin C_a}^{v} p_i q_i = o(1),\end{aligned}$$

and we conclude with same calculation of Lemma 5.13. □

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 5.17. *Let* Γ *be an alphabet of size* $32v/\log\log v \le |\Gamma| = \Theta(v/\log\log v)$ *and fix* $x \in \Gamma^n$. *Let* $\widetilde{V}$ *be a streaming space-*s *algorithm with* $\log v \le s = \text{polylog}(v)$ *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*

$$\mathbb{P}\left[\widetilde{V}(z) \text{ outputs } (z_i, i) \notin C\right] = o(1).$$

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 A zero-knowledge SIP for polynomial evaluation

Our goal in this section will be to combine the components constructed in Sections 5.3 and 5.4 – *homomoprhic* 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 6.1. 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. $\text{PEP}(f, \alpha)$ 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 $q = |\mathbb{F}| = \omega(1)$.

6.1 The protocol

For any mapping f and field element α, Protocol 6.1 lays out $\mathsf{zk\text{-}pep}(f, \alpha)$, our zero-knowledge SIP for $\text{PEP}(f, \alpha)$. Theorems 6.2 and 6.3 prove, respectively, the correctness (i.e., completeness and soundness) and the zero-knowledge properties of 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 3), and that $\boldsymbol{\chi}(\rho)$ denotes a vector of Lagrange polynomials (see Section 5.1); 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 6.1: $\mathsf{zk\text{-}pep}(f, \alpha)$

Input: Explicit access to $\mathbb{F} \supset \mathbb{F}_2$, $\alpha \in \mathbb{F}$, 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

P: Send a string $z \sim \big(\mathbb{F}^m\big)^v$.

V: Sample $\boldsymbol{\rho} \sim \mathbb{F}^m$ and stream z. For each i, 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.

6.2 Analysis of the protocol

We now show that zk-pep is a valid (i.e., complete and sound) streaming interactive proof, as well as compute its space and communication complexities.

Theorem 6.2. *Let f such that an evaluation of f^x can be computed by streaming x in space $O(m \log q)$. Then, for any $\alpha \in \mathbb{F}$, Protocol 6.1 is an SIP for $\textsc{pep}(f, \alpha)$ 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 5.3), 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 5.2 (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 5.2 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

$$O\big(q^m(\log m + \log\log q)m\log q\big) = O\left(q^m m\log^2 q\right) \qquad \text{and}$$
$$O\left(pdm\log q\right) = O\left(d^4m^5q^3\log q\right)$$

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 Zero-knowledge

Having shown that zk-pep is a valid streaming interactive proof, we now show it is also zero-knowledge.

Theorem 6.3. *Protocol 6.1 is zero-knowledge, secure against distinguishers with space $dm^2\,\mathrm{polylog}(q)$. 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 an SIP with a space-s verifier is zero-knowledge against $dm^2\,\mathrm{polylog}(q)$-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 $dm^2\,\mathrm{polylog}(q)$-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 6.1 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 probabilties. More precisely, S calls the whitebox oracle $\mathcal{W}$ (see Definition 4.4) 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(n empty) 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 $\mu_\ell \coloneqq \mathcal{W}\big(b, (z_\ell, \ell)\big)$. 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 $s = \operatorname{polylog}(p)$, Theorem 5.17 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, where it simulates $\widetilde{V}(x)$ and, with $F \coloneqq \{z_i : (z_i, i) \in C\}$, computes $f^x(\boldsymbol{\rho})$ *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 $\beta \in L^{-1}(F)$.[22] 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 $\rho \coloneqq L^{-1}(\boldsymbol{\rho}) \in \mathbb{F}^\times \setminus [dm]$; 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 $\mathbb{F}^m \times [v]$. 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 6.1

Input: Whitebox access to $\widetilde{V}$; oracle access to a length-q^{m+8} random bit string interpreted as $(z, y) \in \big(\mathbb{F}^m\big)^v \times \mathbb{F}^{dm\times p}$; streaming access to $(x, \boldsymbol{\beta}) \in \Gamma^n \times \mathbb{F}^m$.

[22] 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.

Output: View $\big(z, x, \boldsymbol{\beta}, y, \boldsymbol{\gamma}, k, (h(i) : i \in [dm])\big)$ 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

- $\widetilde{\boldsymbol{V}}$: 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

- $\widetilde{\boldsymbol{V}}$: 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 $\Big(\sum_{i=1}^{dm} \chi_i(\rho) \cdot \hat{y}_i \circ L'(j) : j \in [dm]\Big)$.

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 6.4. *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 6.1 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 5.9. 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 $\ell \leq dm$. (Then Lemma 5.9 applies to the bit representation of elements in $\mathbb{F}^\ell$.)

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 5.17.

Viewing L as the set of pairs $\{(L(\sigma), \sigma) : \sigma \in \mathbb{F}\}$, define $\ell \coloneqq dm - |L \cap C|$ and assume,[23] 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 $\big(f^x(i) : i \in [\ell]\big)$ 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: $y_i \coloneqq w_i$ for $i \in [\ell]$, $y_i \sim \mathbb{F}^p$ for $\ell < i < dm$ and

$$y_{dm} \coloneqq \chi_{dm}(\rho')^{-1} \cdot \left(t - \sum_{i=1}^{dm-1} \chi_i(\rho') y_i \right),$$

where $y_{\ell+1}, \ldots, y_{dm-1}$ 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 Fig. 3 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.[24] (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

[23] 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 $dm > |L \cap C|$.

[24] 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.

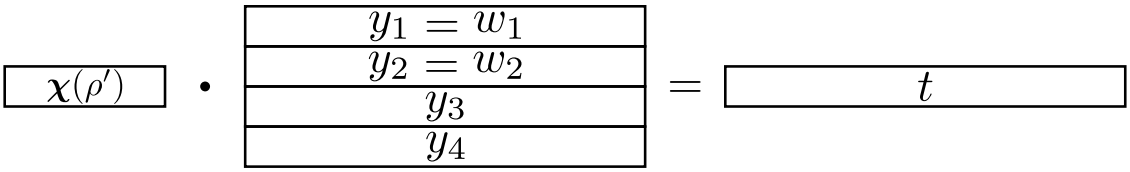


Figure 3: Reduction from INDEX to distinguishability of views when $\ell = 3$ and $dm = 4$. The instance w is inserted into the first 2 rows of y, while y_3 is filled in with joint randomness and y_4 is the solution of the linear system shown in the diagram.

tuple $\boldsymbol{\gamma}$ defined as follows:[25]

$$\boldsymbol{\gamma}_i = \begin{cases} \boldsymbol{\eta}_i, & \text{if } i \le \ell \\ \hat{x}(i) - y_{ik}, & \text{if } \ell < i < dm \end{cases}$$

and

$$\boldsymbol{\gamma}_{dm} \coloneqq \chi_{dm}(\rho')^{-1} \left(f^x_{|L}(\rho') - \chi_0(\rho')\alpha - t_k - \sum_{i=1}^{dm-1} \chi_i(\rho')\boldsymbol{\gamma}_i \right).$$

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 $i \in [dm] \setminus [\ell] = L \cap C$.[26] 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

[25] Recall that all y_i for all $\ell < i < dm$ are contained in Alice and Bob's shared randomness.

[26] 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.

matches f^x in (0 and) $L \cap C$. Since D distinguishes between the two cases with bias $\varepsilon/2$, then so does the one-way protocol. Therefore,

$$\begin{aligned}
&\mathbb{P}_{\substack{w\sim\mathbb{F}^{\ell\times p}\\ k\sim[p]}}\left[B\left(A(w),\left(f^x(i)-w_{ik} : i\in[\ell]\right),k\right)\ \text{accepts}\right] \\
&\qquad - \mathbb{P}_{\substack{w\sim\mathbb{F}^{\ell\times p}\\ k\sim[p]\\ \boldsymbol{\eta}\sim\mathbb{F}^{\ell}}}\left[B\big(A(w),\boldsymbol{\eta},k\big)\ \text{accepts}\right] \\
&\qquad = o(1)\cdot\left(\frac{1}{2}-\frac{1}{2}\right)+\big(1-o(1)\big)\cdot\left(1-\frac{1}{q}\right)\cdot\left(\frac{1}{2}-\frac{1}{2}\right)+\big(1-o(1)\big)\cdot\frac{1}{q}\cdot\frac{\varepsilon}{2} \\
&\qquad \geq \frac{\varepsilon}{3q}\ .
\end{aligned}$$

Finally, note that as $\mathbb{F}_2 \subset \mathbb{F}$, the tuples $\mathbb{F}^\ell$ are in bijection with $\{0,1\}^{\ell \log q}$. Applying Lemma 5.9, we conclude that there exists a one-way binary INDEX protocol for strings of length $p = m(dmq)^3$ with messages of length $\frac{dm^2q^2\ell^2}{\varepsilon^2}\operatorname{polylog}(q) \leq d^3m^4q^{2.01}$ and constant bias, a contradiction with $\sqrt{\frac{d^3m^4q^{2.01}}{p}} = o(1)$. □

6.4 Application: INDEX

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 5.4) as a corollary:

Corollary 6.5. *Fix $\delta \in (0,1]$. For any $\alpha \in \mathbb{F}_q \supset \mathbb{F}_2$ where $q = \Theta\left(\log^{1+\frac{2}{\delta}} n\right)$, DECISION-INDEX($\alpha$) admits a zkSIP with space complexity $O(\log n)$ and communication complexity $O(n^{1+\delta})$. The protocol is secure against $\tilde{O}\left(\log^{1+\frac{2}{\delta}} n\right)$-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 6.1 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 $O(d^4m^5q^3) = \operatorname{polylog}(n)$ in the interactive phase; moreover, it is secure against distinguishers with space $dm^2\operatorname{polylog}(q) = \tilde{O}\left(\log^{2+\frac{2}{\delta}} n\right)$. □

Remark 6.6. Inspecting the proof of Claim 6.4, 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 $s' = \text{polylog}(n)$. 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 $\text{polylog}(n)$-space distinguishers.

7 A zero-knowledge sumcheck SIP

In the previous section we showed how Protocol 5.1, 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 7.2 leys out zk-sumcheck, a zkSIP for the SUMCHECK problem (Definition 7.1) 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 [GKR08] and subsequent delegation-of-computation protocols (e.g., [RVW13, RRR19, RR20]). 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 [CCM09], 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 7.1) 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.[27]

Protocol 7.1: sumcheck(f, α)

Input: Explicit access to $\mathbb{F} = \mathbb{F}_q \supset \mathbb{F}_2$, 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 the polynomial $f_i(T) = \sum_{\beta_{i+1},\ldots,\beta_m \in H} f(\boldsymbol{\rho}_1, \ldots, \boldsymbol{\rho}_{i-1}, T, \beta_{i+1}, \ldots, \beta_m)$.

V: Send $\boldsymbol{\rho}_i$.

[27] Protocol 7.1 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.

$\boldsymbol{V}$: 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 $2 \leq i < m$, 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 sums of polynomials can be performed in a streaming fashion, the verifier only needs $O(m \log q)$ bits of space.

7.1 The protocol

We now show that the techniques of Section 5 enable us to construct a streaming zero-knowledge variant of sumcheck(f, α), which solves the problem defined next.

Definition 7.1. 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. SUMCHECK(f, α) 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 zk-sumcheck and zk-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) $\boldsymbol{\chi}(\rho)$ 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 7.2: zk-sumcheck(f, α)

Input: Explicit access to $\mathbb{F} \supset \mathbb{F}_2$, $\alpha \in \mathbb{F}$, 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

$\boldsymbol{P}$: Send a string $z \sim \big((\mathbb{F} \setminus [d+1])^m\big)^v$.

$\boldsymbol{V}$: 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 $f_1(T) = \sum_{\beta_2,\ldots,\beta_m \in H} f^x(T, \beta_2 \ldots, \beta_m)$ and sample $k \sim [p]$.

$\boldsymbol{V}$: Sample $\boldsymbol{\sigma}^{(1)}, \ldots, \boldsymbol{\sigma}^{(m+1)} \sim \mathbb{F}^m$. Compute $\boldsymbol{\chi}(\boldsymbol{\rho}_1) \ldots, \boldsymbol{\chi}(\boldsymbol{\rho}_m)$ 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 $\boldsymbol{\gamma}^{(i)} = \left(f_i(j) - y^{(i)}_{jk} : j \in [d+1]\right)$.

$\boldsymbol{V}$: 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 $\boldsymbol{\rho}_i$.

$\boldsymbol{P}$: If $i < m$, compute $f_{i+1}(T) = \sum_{\beta_{i+2},\ldots,\beta_m \in H} f^x(\boldsymbol{\rho}_1, \ldots, \boldsymbol{\rho}_i, T, \beta_{i+2}, \ldots, \beta_m)$.

$\boldsymbol{P}$: Send k.

Step 3: Temporal decommitment

$\boldsymbol{V}$: Send ℓ.

$\boldsymbol{P}$: Check that $z_\ell = \boldsymbol{\rho} \in \left(\mathbb{F} \setminus [d+1]\right)^m$, aborting otherwise.

Step 4: Algebraic decommitments

$\boldsymbol{V}$: For all $1 < i \leq m$, 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 $\mathsf{decommit}\left(f^x(\boldsymbol{\rho}), \boldsymbol{\chi}(\boldsymbol{\rho}_m) \cdot y^{(m)}, k\right)$ with fingerprint $\hat{y}^{(m)}\left(\boldsymbol{\sigma}^{(m+1)}, \boldsymbol{\chi}(\boldsymbol{\rho}_m)\right)$ and correction $\boldsymbol{\chi}(\boldsymbol{\rho}_m) \cdot \boldsymbol{\gamma}^{(m)}$.
Accept if all decommitments accept, and reject otherwise.

7.2 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 7.2. *Let f such that an evaluation of f^x can be computed by streaming x in space $O(m^2 \log q)$. For any $\alpha \in \mathbb{F}$, Protocol 7.2 is an SIP for* SUMCHECK(f, α) *with $s = O(m^2 \log q)$ space complexity and communication complexities $O(q^m m \log^2 q)$ and $O(q^{\log\log q} dm \log q) = q^{\log\log q}\,\mathrm{poly}(q)$ in the setup and interactive phases, respectively.*

Proof. As in Theorem 6.2, we first show completeness and soundness, then compute the complexities.

Completeness. Recall that $\mathsf{decommit}(\beta, w, k)$ 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

$$\begin{aligned}
\boldsymbol{\theta}\cdot\boldsymbol{\gamma}^{(i)} - \boldsymbol{\chi}(\boldsymbol{\rho}_{i-1})\cdot\boldsymbol{\gamma}^{(i-1)} + \left(\boldsymbol{\theta}\cdot y^{(i)} - \boldsymbol{\chi}(\boldsymbol{\rho}_{i-1})\cdot y^{(i-1)}\right)_k \\
&= \sum_{j=1}^{d+1}\left(\boldsymbol{\theta}_j\left(\boldsymbol{\gamma}_j^{(i)} + y_{jk}^{(i)}\right) - \chi_j(\boldsymbol{\rho}_{i-1})\left(\boldsymbol{\gamma}_j^{(i-1)} + y_{jk}^{(i-1)}\right)\right) \\
&= \sum_{j=1}^{d+1}\boldsymbol{\theta}_j f_i(j) - \sum_{j=1}^{d+1}\chi_j(\boldsymbol{\rho}_{i-1})f_{i-1}(j) \\
&= \left(\sum_{\beta\in H} f_i(\beta)\right) - f_{i-1}(\boldsymbol{\rho}_{i-1}) \\
&= 0.
\end{aligned}$$

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 5.2 (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 $O(m^2 \log q)$. □

7.3 Zero-knowledge

Having shown that zk-sumcheck is a valid streaming interactive proof, we now show it is also zero-knowledge.

Theorem 7.3. *Protocol 7.2 is zero-knowledge against* $\mathrm{poly}(q)$*-space streaming distinguishers. The simulator has space complexity* $\mathrm{poly}(q)$*.*

Proof. We shall prove indistinguishability as we have done earlier: with the simulator S shown in Algorithm 7.1, 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 $\mathrm{poly}(q)$-space distinguisher D that accepts $\mathrm{View}_{P,\widetilde{V}}(x, r)$ with probability $\varepsilon = \Omega(1)$ above that with which D accepts $S\left(\widetilde{V}, x, r\right)$. Then, via Lemma 5.9, 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 $\mathrm{poly}(q)$ time, and thus space.[28] Note, moreover, that the alphabet over which z is taken has size

[28] 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 5.17 applies. (The conditions $(q-d-1)^m = \Theta\left(\frac{v}{\log\log v}\right)$ and $\log q \leq s = \text{polylog}(q)$ are also clearly satisfied.)

Algorithm 7.1: Simulator for Protocol 7.2

Input: Whitebox access to $\widetilde{V}$; oracle access to random bit string of length $q^{m+\log\log q}\,\text{poly}(q)$ 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 $\left(z, x, \left(y^{(i)}, \boldsymbol{\gamma}^{(i)} : i \in [m]\right), k, \left(h_i : i \in [m+1]\right)\right)$ with $z \in \left((\mathbb{F} \setminus [d+1])^d\right)^v$, $y^{(i)} \in \mathbb{F}^{(d+1)\times p}$, $\boldsymbol{\gamma}^{(i)} \in \mathbb{F}^{d+1}$, $k \in [p]$ and $h_i : \mathbb{F} \to \mathbb{F}$ of 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 given by taking g randomly 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 $y_j^{(i)} = y_j'^{(i)} + (\boldsymbol{\xi} \cdot w)_{(i-1)d+j}$. She also sets the remaining rows (i.e., $y_{d+1}^{(i)}$ for all i as well as $y_d^{(m)}$) to satisfy

$$\begin{aligned} \boldsymbol{\theta} \cdot y^{(1)} &= t^{(1)}, \\ \boldsymbol{\theta} \cdot y^{(i)} - \boldsymbol{\chi}(\boldsymbol{\rho}'_{i-1}) \cdot y^{(i-1)} &= t^{(i)} \quad \text{for } 1 < i \leq m \text{ and} \\ \boldsymbol{\chi}(\boldsymbol{\rho}'_m) \cdot y^{(m)} &= t^{(m+1)}. \end{aligned}$$

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.[29] (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 $y_{d+1}^{(i)}$ 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 $\boldsymbol{\gamma}_j^{(i)} = (\boldsymbol{\xi} \cdot \boldsymbol{\eta})_{(i-1)d+j} - y_{jk}'^{(i)}$ 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

$$\begin{aligned} \boldsymbol{\theta} \cdot \boldsymbol{\gamma}^{(1)} &= \alpha - t_k^{(1)}, \\ \boldsymbol{\theta} \cdot \boldsymbol{\gamma}^{(i)} - \boldsymbol{\chi}(\boldsymbol{\rho}'_{i-1}) \cdot \boldsymbol{\gamma}^{(i-1)} &= -t_k^{(i)} \quad \text{for } 1 < i \leq m \text{ and} \\ \boldsymbol{\chi}(\boldsymbol{\rho}'_m) \cdot \boldsymbol{\gamma}^{(m)} &= f^x(\boldsymbol{\rho}') - t_k^{(m+1)}. \end{aligned}$$

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 $\boldsymbol{\gamma}_j^{(i)}$ satisfies

$$\begin{aligned} \boldsymbol{\gamma}_j^{(i)} &= (\boldsymbol{\xi} \cdot \boldsymbol{\eta})_{(i-1)d+j} - y_j'^{(i)} \\ &= g_i(j) - (\boldsymbol{\xi} \cdot w)_{(i-1)d+j,k} - y_{jk}'^{(i)} \\ &= g_i(j) - y_{jk}^{(i)} \end{aligned}$$

for all i, j in $[m-1] \times [d]$ and $\{m\} \times [d-1]$ (equivalently, for all i, j such that $y_j^{(i)}$ 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

[29] 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 $\alpha = 0$.

$$\begin{aligned}
\boldsymbol{\theta} \cdot \boldsymbol{\gamma}^{(1)} &= \alpha - t_k^{(1)} \\
&= \sum_{j=1}^{d+1} \boldsymbol{\theta}_j g(j) - t_k^{(1)} \\
&= \sum_{j=1}^{d+1} \boldsymbol{\theta}_j \left(g(j) - y_{jk}^{(1)} \right), \\
\boldsymbol{\chi}(\boldsymbol{\rho}_m) \cdot \boldsymbol{\gamma}^{(m)} &= f^x(\boldsymbol{\rho}) - t_k^{(m+1)} \\
&= g_m(\boldsymbol{\rho}_m) - t_k^{(m+1)} \\
&= \sum_{j=1}^{d+1} \chi_j(\boldsymbol{\rho}_m) \left(g_m(j) - y_{jk}^{(m)} \right),
\end{aligned}$$

and, for $1 < i \le m$,

$$\begin{aligned}
\boldsymbol{\theta} \cdot \boldsymbol{\gamma}^{(i)} - \boldsymbol{\chi}(\boldsymbol{\rho}_{i-1}) \cdot \boldsymbol{\gamma}^{(i-1)} &= -t_k^{(i)} \\
&= \sum_{j=1}^{d+1} \left(\chi_j(\boldsymbol{\rho}_{i-1}) y_{jk}^{(i-1)} - \boldsymbol{\theta}_j y_{jk}^{(i)} \right) \\
&= \sum_{j=1}^{d+1} \boldsymbol{\theta}_j \left(g_i(j) - y_{jk}^{(i)} \right) + \sum_{j=1}^{d+1} \chi_j(\boldsymbol{\rho}_{i-1}) \left(g_{i-1}(j) - y_{jk}^{(i-1)} \right).
\end{aligned}$$

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 $\mathrm{View}_{P,\widetilde{V}}(x, r)$ 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

$$\begin{aligned}
&\mathbb{P}_{\substack{w \sim \mathbb{F}^{\ell \times p} \\ k \sim [p]}} \left[B\left(A(w), \left(f^x(i) - w_{ik} : i \in [\ell]\right), k\right) \text{ accepts} \right] \\
&\qquad - \mathbb{P}_{\substack{w \sim \mathbb{F}^{\ell \times p} \\ k \sim [p] \\ \boldsymbol{\eta} \sim \mathbb{F}^\ell}} \left[B\left(A(w), \boldsymbol{\eta}, k\right) \text{ accepts} \right] \\
&\qquad = o(1) \cdot 0 + \left(1 - o(1)\right) \cdot \left(1 - \frac{1}{s}\right) \cdot 0 + \left(1 - o(1)\right) \cdot \frac{1}{s} \cdot \frac{\varepsilon}{2} \\
&\qquad \ge \frac{\varepsilon}{3s}.
\end{aligned}$$

Applying Lemma 5.9 (to the bit representation of elements in $\mathbb{F}^\ell$), there exists a one-way binary INDEX protocol for strings of length $p = q^{\log \log q}$ with messages of length $\frac{s^2 \ell^2}{\varepsilon^2} \operatorname{poly}(q) = \operatorname{poly}(q)$ and constant bias. But this contradicts Proposition 5.5's upper bound of $O\left(\sqrt{\operatorname{poly}(q)/p}\right) = o(1)$. □

7.4 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].

Definition 7.4. Fix $k \in \mathbb{N}$. For every $\ell \in [n]$ and $t \in [n^k]$, the language $\text{FREQUENCY-MOMENT}_k(t)$ is $\left\{x \in [\ell]^n : \sum_{i\in[\ell]} \phi_i(x)^k = t\right\}$, where $\phi_i(x) \coloneqq |\{j \in [n] : x_j = i\}|$.

Corollary 7.5. *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 $\text{FREQUENCY-MOMENT}_k(t)$ 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 $\text{polylog}(n)$-space distinguishers.*

Proof. We set the same parameters of Corollary 6.5: degree $d = \log^{\frac{2}{\delta}} n$, dimension $m = \frac{\delta \log n}{2 \log\log n}$, and take a field $\mathbb{F} \supset \mathbb{F}_2$ 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 $\varphi = \phi(x) \coloneqq \big(\phi_i(x) : i \in [\ell]\big)$, set $f^x(\boldsymbol{\alpha}) \coloneqq \sum_{i\in[d+1]} \hat{\varphi}(i, \boldsymbol{\alpha})^k$ 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 sum to the k^{th} power and adds the results to obtain $f^x(\boldsymbol{\rho})$.

Applying Protocol 7.2, 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

$$q^m m \log^2 q = n^{1+\frac{\delta}{2}} \,\text{polylog}(n) = O\left(n^{1+\delta}\right),$$

and $q^{\log\log q} \,\text{poly}(q) = n^{o(1)}$ in the interactive phase. Lastly, the protocol is secure against distinguishers with space $\text{poly}(q) = \text{polylog}(n)$. □

Our last application is a small modification of the F_2 protocol that allows us to compute inner products.

Definition 7.6. For every $\ell \in [n]$, $t \in [n^2\ell]$ and field $\mathbb{F}$, the language $\text{INNER-PRODUCT}(t)$ is defined as $\left\{(x, y) \in \mathbb{F}^n \times \mathbb{F}^n : \phi(x) \cdot \phi(y) = \sum_{i\in[\ell]} \phi_i(x)\phi_i(y) = t\right\}$.

Corollary 7.7. *For every $\delta \in (0, 1]$, $\ell \in [n]$, $t \in [n^2\ell]$ and field $\mathbb{F}_q \supset \mathbb{F}_2$ with $q = \Theta\left(\log^{1+\frac{2}{\delta}} n\right)$, there exists a zkSIP for $\text{INNER-PRODUCT}(t)$ 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 7.5 and define

$$f^{x,y}(\boldsymbol{\alpha}) = \sum_{i\in[d+1]} \widehat{\phi(x)}(i, \boldsymbol{\alpha})\widehat{\phi(y)}(i, \boldsymbol{\alpha}),$$

a polynomial of degree $2d = 2\log^{\frac{2}{\delta}} n$ whose evaluation the verifier computes by saving $\widehat{\phi(x)}(i, \boldsymbol{\rho})$ and $\widehat{\phi(y)}(i, \boldsymbol{\rho})$ for $i \in [d+1]$. Protocol 7.1 enables the verifier to check that $\sum_{i \in [\ell]} \phi_i(x)\phi_i(y)$ equals t, as desired. □

We remark that while one might reduce inner product to F_2, by taking the difference between the second moment of $\phi(x) + \phi(y)$ and the second moments of $\phi(x)$ and $\phi(y)$, the resulting protocol leaks these values, and is therefore not zero-knowledge.

Acknowledgements

We thank Aditya Prakash for the proof of Claim 5.14, as well as Justin Thaler and Nick Spooner for fruitful discussions and careful reading of an earlier version of this manuscript.

References

[AB09] Sanjeev Arora and Boaz Barak. *Computational Complexity - A Modern Approach*. Cambridge University Press, 2009.

[AMS99] Noga Alon, Yossi Matias, and Mario Szegedy. The space complexity of approximating the frequency moments. *Journal of Computer and system sciences*, 58(1):137–147, 1999.

[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 Yael Kalai and Leonid Reyzin, editors, *Theory of Cryptography - 15th International Conference, TCC 2017, Baltimore, MD, USA, November 12-15, 2017, Proceedings, Part II*, volume 10678 of *Lecture Notes in Computer Science*, pages 172–206. Springer, 2017.

[BLM13] Stéphane Boucheron, Gábor Lugosi, and Pascal Massart. *Concentration inequalities: A nonasymptotic theory of independence*. Oxford university press, 2013.

[Blu83] Manuel Blum. Coin flipping by telephone a protocol for solving impossible problems. *ACM SIGACT News*, 15(1):23–27, 1983.

[BRV18] Itay Berman, Ron D Rothblum, and Vinod Vaikuntanathan. Zero-knowledge proofs of proximity. In *9th Innovations in Theoretical Computer Science Conference (ITCS 2018)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2018.

[BSBHR18] Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, and Michael Riabzev. Scalable, transparent, and post-quantum secure computational integrity. *Cryptology ePrint Archive*, 2018.

[BSCG+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 *Annual cryptology conference*, pages 90–108. Springer, 2013.

[BSCG+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*, pages 459–474. IEEE, 2014.

[BSCG+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 Conference*, pages 494–521. Springer, 2019.

[BSCR+19] Eli Ben-Sasson, Alessandro Chiesa, Michael Riabzev, Nicholas Spooner, Madars Virza, and Nicholas P Ward. Aurora: Transparent succinct arguments for r1cs. In *Annual international conference on the theory and applications of cryptographic techniques*, pages 103–128. Springer, 2019.

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

[CCM09] Amit Chakrabarti, Graham Cormode, and Andrew Mcgregor. Annotations in data streams. *Automata, Languages and Programming*, pages 222–234, 2009.

[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)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2015.

[CFGS22] Alessandro Chiesa, Michael A. Forbes, Tom Gur, and Nicholas Spooner. Spatial isolation implies zero knowledge even in a quantum world. *J. ACM*, 69(2):15:1–15:44, 2022.

[CG18] Alessandro Chiesa and Tom Gur. Proofs of proximity for distribution testing. In *9th Innovations in Theoretical Computer Science Conference (ITCS 2018)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2018.

[CG19] Amit Chakrabarti and Prantar Ghosh. Streaming verification of graph computations via graph structure. *APPROX/RANDOM 2019, September 20-22, 2019*, 2019.

[CGT20] Amit Chakrabarti, Prantar Ghosh, and Justin Thaler. Streaming verification for graph problems: Optimal tradeoffs and nonlinear sketches. *arXiv preprint arXiv:2007.03039*, 2020.

[CH18] Graham Cormode and Chris Hickey. Cheap checking for cloud computing: Statistical analysis via annotated data streams. In *AISTATS*, 2018.

[CMT12] Graham Cormode, Michael Mitzenmacher, and Justin Thaler. Practical verified computation with streaming interactive proofs. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, pages 90–112. ACM, 2012.

[CMT13] Graham Cormode, Michael Mitzenmacher, and Justin Thaler. Streaming graph computations with a helpful advisor. *Algorithmica*, 65(2):409–442, 2013.

[CTY11] Graham Cormode, Justin Thaler, and Ke Yi. Verifying computations with streaming interactive proofs. *Proceedings of the VLDB Endowment*, 5(1):25–36, 2011.

[DGMT22] Marcel Dall'Agnol, Tom Gur, Subhayan Roy Moulik, and Justin Thaler. Quantum proofs of proximity. *Quantum*, 6:834, 2022.

[DTV15] Samira Daruki, Justin Thaler, and Suresh Venkatasubramanian. Streaming verification in data analysis. In *International Symposium on Algorithms and Computation*, pages 715–726. Springer, 2015.

[GG21] Oded Goldreich and Tom Gur. Universal locally verifiable codes and 3-round interactive proofs of proximity for CSP. *Theor. Comput. Sci.*, 878-879:83–101, 2021.

[GGR18] Oded Goldreich, Tom Gur, and Ron D Rothblum. Proofs of proximity for context-free languages and read-once branching programs. *Information and Computation*, 261:175–201, 2018.

[GKR08] Shafi Goldwasser, Yael Tauman Kalai, and Guy N Rothblum. Delegating computation: interactive proofs for muggles. In *Proceedings of the fortieth annual ACM symposium on Theory of computing*, pages 113–122. ACM, 2008.

[GLR21] Tom Gur, Yang P. Liu, and Ron D. Rothblum. An exponential separation between MA and AM proofs of proximity. *Comput. Complex.*, 30(2):12, 2021.

[Gol02] Oded Goldreich. Zero-knowledge twenty years after its invention. *IACR Cryptol. ePrint Arch.*, 2002:186, 2002.

[Gol08] Oded Goldreich. Computational complexity: a conceptual perspective. *ACM Sigact News*, 39(3):35–39, 2008.

[GR15] Tom Gur and Ran Raz. Arthur–merlin streaming complexity. *Information and Computation*, 243:145–165, 2015.

[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)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2017.

[GR18] Tom Gur and Ron D Rothblum. Non-interactive proofs of proximity. *computational complexity*, 27(1):99–207, 2018.

[GRS12] Venkatesan Guruswami, Atri Rudra, and Madhu Sudan. *Essential coding theory*. Draft available at https://cse.buffalo.edu/faculty/atri/courses/coding-theory/book/, 2012.

[Gur17] Tom Gur. *On locally verifiable proofs of proximity*. PhD thesis, The Weizmann Institute of Science (Israel), 2017.

[HILL99] Johan Håstad, Russell Impagliazzo, Leonid A Levin, and Michael Luby. A pseudorandom generator from any one-way function. *SIAM Journal on Computing*, 28(4):1364–1396, 1999.

[IL89] Russell Impagliazzo and Michael Luby. One-way functions are essential for complexity based cryptography. In *30th Annual Symposium on Foundations of Computer Science*, pages 230–235. IEEE Computer Society, 1989.

[IW14] Yuval Ishai and Mor Weiss. Probabilistically checkable proofs of proximity with zero-knowledge. In Yehuda Lindell, editor, *Theory of Cryptography - 11th Theory of Cryptography Conference, TCC 2014, San Diego, CA, USA, February 24-26, 2014. Proceedings*, volume 8349 of *Lecture Notes in Computer Science*, pages 121–145. Springer, 2014.

[Nao91] Moni Naor. Bit commitment using pseudorandomness. *Journal of cryptology*, 4(2):151–158, 1991.

[Rab81] Michael O Rabin. *Fingerprinting by random polynomials*. Center for Research in Computing Techn., Aiken Computation Laboratory, Univ., 1981.

[RR20] Guy N Rothblum and Ron D Rothblum. Batch verification and proofs of proximity with polylog overhead. In *Theory of Cryptography Conference*, pages 108–138. Springer, 2020.

[RRR19] Omer Reingold, Guy N Rothblum, and Ron D Rothblum. Constant-round interactive proofs for delegating computation. *SIAM Journal on Computing*, 50(3):STOC16–255, 2019.

[RVW13] Guy N Rothblum, Salil Vadhan, and Avi Wigderson. Interactive proofs of proximity: delegating computation in sublinear time. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*, pages 793–802, 2013.

[RY20] Anup Rao and Amir Yehudayoff. *Communication Complexity: and Applications*. Cambridge University Press, 2020.

[Sch80] J. T. Schwartz. Fast probabilistic algorithms for verification of polynomial identities. *J. ACM*, 27(4):701–717, October 1980.

[SYZ+21] Xiaoqiang Sun, F. Richard Yu, Peng Zhang, Zhiwei Sun, Weixin Xie, and Xiang Peng. A survey on zero-knowledge proof in blockchain. *IEEE Network*, 35(4):198–205, 2021.

[Tha13] Justin Thaler. Time-optimal interactive proofs for circuit evaluation. In *Annual Cryptology Conference*, pages 71–89. Springer, 2013.

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

[Vad07] Salil Vadhan. The complexity of zero knowledge. In *International Conference on Foundations of Software Technology and Theoretical Computer Science*, pages 52–70. Springer, 2007.

A Deferred proofs

A.1 Proof of Proposition 5.5

Proposition A.1 (Proposition 5.5, restated)**.** *Any one-way communication protocol for* SEARCH-INDEX *with input* $(x, j) \sim \Gamma^p \times [p]$ *that sends an* s*-bit message succeeds with probability at most* $\frac{1}{|\Gamma|} + O\left(\sqrt{s/p}\right)$.

Proof. 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,

$$\begin{aligned}
\mathbb{P}_{\substack{x\sim\Gamma^p \\ j\sim[p]}}[\text{Bob outputs } x_j] &= \sum_{a\in\{0,1\}^s} \mathbb{P}_{x\sim\Gamma^p}[A(x) = a] \cdot \mathbb{P}_{\substack{x\sim\Gamma^p \\ j\sim[p]}}\big[b(a, j) = x_j \;\big|\; A(x) = a\big] \\
&= \sum_{a\in\{0,1\}^s} \mathbb{P}_{A\sim\lambda}[A = a] \cdot \mathbb{P}_{\substack{x\sim P_a \\ j\sim[p]}}\big[b(a, j) = x_j\big] \\
&= \mathbb{E}_{\substack{A\sim\lambda \\ j\sim[p]}}\left[\mathbb{P}_{x\sim P_A}\big[b(A, j) = x_j\big]\right] \\
&\le \mathbb{E}_{\substack{A\sim\lambda \\ j\sim[p]}}\left[\max_{\alpha\in\Gamma}\{\mathbb{P}_{x\sim P_A}[x_j = \alpha]\}\right], \qquad (14)
\end{aligned}$$

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. 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. 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

$$\begin{aligned}\mathbb{E}_{\substack{A\sim\lambda\\ j\sim[p]}}\left[\max_{\alpha\in\Gamma}\left\{\mathbb{P}_{x\sim P_A}[x_j=\alpha]\right\}\right] - \frac{1}{\gamma} &= \mathbb{E}_{\substack{A\sim\lambda\\ j\sim[p]}}\left[\max_{\alpha\in\Gamma}\left\{\mu_{j,A}(\alpha)-\frac{1}{\gamma}\right\}\right]\\ &\le \mathbb{E}_{\substack{A\sim\lambda\\ j\sim[p]}}\left[\max_{\alpha\in\Gamma}\left\{\left|\mu_{j,A}(\alpha)-\frac{1}{\gamma}\right|\right\}\right]\\ &\le \mathbb{E}_{\substack{A\sim\lambda\\ j\sim[p]}}\left[\|\mu_{j,A}-\mu\|\right]\\ &\le \sqrt{\frac{\sum_{i=1}^{p} I(A:x_i)}{2p\ln 2}},\end{aligned}$$

so that using Eq. 14 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,

$$\begin{aligned}\sum_{i=1}^{p} I(A:x_i) &= \sum_{i=1}^{p}\big(H(x_i)-H(x_i|A)\big) && \text{(by Eq. 11)}\\ &= H(x)-\sum_{i=1}^{p} H(x_j|A) && \text{(by Eq. 7)}\\ &\le H(x)-\sum_{i=1}^{n} H(x_i|x_1,\ldots,x_{i-1},A) && \text{(by Eq. 6)}\\ &= H(x)-H(x|A) && \text{(by Eq. 8)}\\ &= I(A:x) \le H(A) && \text{(by Eq. 11)}\\ &\le s && \text{(by Eq. 5)}\end{aligned}$$

and the result follows. □

A.2 Proof of Theorem 5.8

Theorem A.2 (Theorem 5.8, restated)**.** *Protocol 5.4 (algebraic-commit) and Protocol 5.3 (decommit) form a streaming commitment protocol with space complexity $s = O\big((\ell+m)\log q\big)$ if $p = q^{3\ell}$ and $dm = \mathrm{polylog}(q)$. The scheme is secure against $\mathrm{poly}(s)$-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 $s = O(m\log q)$.

Proof. We follow the same steps of Theorem 5.6, beginning with the binding property: using $y^{(i)}$ to denote the i^{th} column of y, when P is honest, i.e., sends the correction tuple $\boldsymbol{\gamma} = \boldsymbol{\alpha} - y^{(k)}$ in the commit stage and the polynomial $\hat{z}_{|L}$ where $z = \boldsymbol{\beta} \cdot y$ in the decommit stage, then V accepts as $\hat{z}_{|L}(\rho) = \hat{z}(\boldsymbol{\rho}) = \hat{y}(\boldsymbol{\rho}, \boldsymbol{\beta})$ and $\hat{z}_{|L}(0) + \gamma = z_k + \boldsymbol{\beta} \cdot \boldsymbol{\gamma} = \boldsymbol{\alpha} \cdot \boldsymbol{\beta}$. (Recall that the line L is such that $L(0) = k$ and $L(\rho) = \boldsymbol{\rho}$.)

Now, suppose P replies with a polynomial g such that $g(0) \neq \sum_{i \in [\ell]} \boldsymbol{\beta}_i y_{ik} = z_k = \hat{z}_{|L}(0)$; then the Schwartz-Zippel lemma implies $g(\rho) \neq \hat{z}_{|L}(\rho)$ except with probability $dm/q = o(1)$, in which case V rejects. As the verifier only needs to store the evaluation point $\boldsymbol{\rho} \in \mathbb{F}^m$, the coordinate $k \in [p]$ and a constant number of additional field elements, its space complexity is $O(m \log q)$ as long as each $\boldsymbol{\beta}_i$ can be computed in this space (e.g., when $\boldsymbol{\beta}_i = \boldsymbol{\beta}_i(\rho)$ is the evaluation of an m-variate polynomial over $\mathbb{F}$); if $\boldsymbol{\beta}$ must be stored in its entirety, the complexity becomes $O\big((\ell + m) \log q\big)$.

To show the hiding property, assume towards contradiction that there exists a streaming algorithm D with space $\text{poly}(s) = \text{poly}(\ell, \log q)$ that distinguishes commitments between some $\boldsymbol{\alpha} \in \mathbb{F}^\ell$ and $\boldsymbol{\alpha}' \in \mathbb{F} \setminus \{\boldsymbol{\alpha}\}$ with constant bias: that is,

$$\mathbb{P}_{\substack{y \sim \mathbb{F}^{\ell \times p} \\ k \sim [p]}} \Big[D(y, \boldsymbol{\alpha} - y^{(k)}, k) \text{ accepts} \Big] - \mathbb{P}_{\substack{y \sim \mathbb{F}^{p} \\ k \sim [p]}} \Big[D(y, \boldsymbol{\alpha}' - y^{(k)}, k) \text{ accepts} \Big] \geq \varepsilon$$

for some $\varepsilon = \Omega(1)$. Now consider the following one-way communication protocol for SEARCH-INDEX over the alphabet $\mathbb{F}^\ell$ with input $(x, j) \in (\mathbb{F}^\ell)^p \times [p]$: Alice, viewing x as an element of $\mathbb{F}^{\ell \times p}$, simulates D on the stream $(x, \boldsymbol{\gamma})$, where $\boldsymbol{\gamma} \sim \mathbb{F}^\ell$, and sends the polylog$(p)$-bit snapshot of D to Bob, who finishes the simulation with j; if D accepts output $\boldsymbol{\alpha} - \boldsymbol{\gamma}$, and otherwise output $\boldsymbol{\alpha}' - \boldsymbol{\gamma}$. Note that Bob outputs correctly exactly when $\boldsymbol{\gamma} = \boldsymbol{\alpha} - y^{(k)}$ and D accepts, or $\boldsymbol{\gamma} = \boldsymbol{\alpha}' - y^{(k)}$ and D rejects. We will now show that the protocol solves SEARCH-INDEX with a bias that is too large, contradicting Proposition 5.5.

$$\begin{aligned}
&\mathbb{P}_{\substack{x \sim (\mathbb{F}^\ell)^p \\ j \sim [p]}} [\text{Bob outputs } x_j] \\
&\quad = \frac{1}{q^\ell} \cdot \mathbb{P}_{\substack{x \sim (\mathbb{F}^\ell)^p \\ j \sim [p]}} \left[D(x, \boldsymbol{\alpha} - x_j, j) \text{ accepts} \right] + \frac{1}{q^\ell} \cdot \mathbb{P}_{\substack{x \sim (\mathbb{F}^\ell)^p \\ j \sim [p]}} \Big[D(x, \boldsymbol{\alpha}' - x_j, j) \text{ rejects} \Big] \\
&\quad = \frac{1}{q^\ell} \left(1 + \mathbb{P}_{\substack{x \sim (\mathbb{F}^\ell)^p \\ j \sim [p]}} \left[D(x, \boldsymbol{\alpha} - x_j, j) \text{ accepts} \right] - \mathbb{P}_{\substack{x \sim (\mathbb{F}^\ell)^p \\ j \sim [p]}} \Big[D(x, \boldsymbol{\alpha}' - x_j, j) \text{ accepts} \Big] \right) \\
&\quad \geq \frac{1 + \varepsilon}{q^\ell} \\
&\quad = \frac{1}{q^\ell} + \Omega\left(\frac{1}{q^\ell} \right).
\end{aligned}$$

Since $q^{-\ell} = \Omega\left(\sqrt{q^\ell / p} \right) = \omega\left(\sqrt{\text{poly}(s)/p} \right)$, owing to $s = \text{poly}(\ell, \log q)$, the result follows. The communication complexity of the protocols is dominated by the prover sending ℓp field elements, for a total of $O(\ell q^{3\ell} \log q)$ bits. □

A.3 Proof of Claim 5.14

Claim A.3 (Claim 5.14, restated)**.** *Let $p, q \in [0,1]^v$ be probability vectors and $t \in [v]$ a positive integer. There exists a set $C \subseteq [v]$ of size t such that $\sum_{i \in [v] \setminus C} p_i q_i \leq 1/t$.*

Proof. 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 $f : \Delta' \times \Delta \to \mathbb{R}_+$ 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

$$i p_i^* (q_1^* + q_i^*) \leq p_1^* q_1^* + i p_i^* q_i^*,$$

and thus $i p_i^* \leq p_1^*$ (since $q_1^* \neq 0$). But then

$$f(p^*, q^*) = \sum_{i=1}^{\ell} i p_i^* q_i^* \leq p_1^* \sum_{i=1}^{\ell} q_i^* = p_1^* < 1,$$

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. □