Electronic Colloquium on Computational Complexity, Report No. 181 (2024)

Zero-Knowledge in Streaming Interactive Proofs

Tomer Gewirtzman[*] Ron D. Rothblum[†]

Abstract

In a recent work, Cormode, Dall'Agnol, Gur and Hickey (CCC, 2024) introduced the model of *Zero-Knowledge Streaming Interactive Proofs* (zkSIPs). Loosely speaking, such proof-systems enable a prover to convince a *streaming verifier* that the input x, to which it has read-once streaming access, satisfies some property, in such a way that nothing beyond the correctness of the claim is revealed. Cormode *et al.* also gave constructions of zkSIPs to some specific and notable problems of interest.

In this work, we advance the study of zero-knowledge proofs in the streaming model, by presenting protocols that are significantly more *general* and more *secure*. We use a definition of zero-knowledge that is a variation of that used by Cormode *et al.*, which we find more appealing but is technically incomparable.

Our main result is a zkSIP for any NP relation, that can be decided by low-depth polynomial-size circuits. We emphasize that this is the first *general purpose* protocol in this model, which captures, as a special case, the problems considered by the prior work. We also construct a specialized protocol for the "polynomial evaluation" problem considered in that work, with improved parameters.

The protocols constructed by Cormode *et al.* have an inverse polylogarithmic simulation error (i.e., a gap with which a bounded-space distingiusher can distinguish the simulation from a real execution). This means that their protocols are entirely insecure if run multiple times (say on different inputs). In contrast, our protocols achieve a *negligible* zero-knowledge error, a stronger and far more robust security guarantee.

[*] Technion. Email: gewirtzman@cs.technion.ac.il

[†] Succinct and Technion. Email: rothblum@gmail.com

ISSN 1433-8092

Contents

1 Introduction

Interactive proofs [GMR89] are a fundamental computational paradigm involving two players: a computationally powerful but untrusted prover, and a verifier, who lacks the resources to independently perform a desired computation. Interactive proofs enable the verifier, by interacting with the prover, to ascertain the truth of complex claims that it cannot directly compute. Crucially, the protocol must also guard against malicious provers attempting to trick the verifier to accept false statements.

The conceptual evolution of interactive proofs paved the way for the emergence of the astonishing concept of *zero-knowledge proofs* [GMR89]. Such proofs enable a prover to prove the correctness of its claim to the verifier, while disclosing no extraneous information beyond the mere fact of the claim's truth. This ingenious concept has had far-reaching implications and applications, both in theory and practice.

Another branch of study in the field of interactive proofs that has drawn significant attention in the last decade is *streaming interactive proofs* ([CTY11, CMT12, CCM+15, GR15, CGT20]). Here, similarly to a classical interactive proof, a computationally limited verifier uses the help of a powerful untrusted prover to verify some claim. However, in this model the computational constraints of the verifier are in the form of limited memory that prevent it from storing the entirety of the input, which is read by the verifier in a sequential one-pass manner.

In a recent exciting work, Cormode *et al.* [CDGH23] introduced the model of *zero-knowledge streaming interactive proofs* (zkSIP). Their model extends the streaming interactive proof model by requiring that the proof be *zero-knowledge*. Loosely speaking, this means that a bounded space (potentially malicious) verifier, does not learn anything in the interaction beyond the correctness of the statement. Cormode *et al.* also gave constructions of such zkSIPs for several interesting problems, which are discussed in further detail below.

We emphasize that the zkSIP model is quite challenging. In the classical setting, both participants have full access to the input. In contrast, in the streaming scenario, the verifier reads the input bits sequentially, and has no space to store it entirely. In particular, the verifier does not have the luxury of full random access to the input at will.

We continue the line of work that was initiated in [CDGH23]. Jumping ahead, we present *general purpose* protocols in the zkSIP model, which, moreover, achieve what we find to be a substantially stronger notion of zero-knowledge (although formally the definitions are incomparable) and with better parameters. This shows that the expressive power of such protocols is much broader than shown in the original work of Cormode *et al.* (who gave constructions for specific problems of interest). We elaborate on these results in Section 1.2 but first discuss the model of zero-knowledge proofs in the streaming setting.

1.1 Streaming Zero-Knowledge Proofs

A *streaming interactive proof* involves a computationally unbounded prover $\mathcal{P}$ and a bounded-space, probabilistic verifier $\mathcal{V}$. As in the classic interactive proof model, both parties exchange messages based on a shared input. However, the key distinction in this model is that the input is streamed. While this poses no issue for the prover, who can store and access the entire stream, it creates a challenge for the verifier, whose memory is limited. The prover and verifier may interact amongst themselves while simultaneously streaming the input, and messages exchanged during the interaction are also sent as a stream of bits.[1] At the conclusion of the interaction, the verifier decides whether to accept or reject.[2]

As in the standard interactive proof model, we aim to preserve both completeness and soundness. This means that the prover should be able to use the protocol to convince the verifier of true statements, while any malicious prover should be unable to convince the verifier of a false claim. The protocols constructed in our work will be *doubly-efficient* [GKR15, Gol18], meaning the the honest prover runs in polynomial time. We emphasize however that we focus on *statistical soundness* – even an unbounded prover (both in terms of time and space) should be unable to convince the verifier to accept a false statement (other than with small probability).

Following Cormode *et al.*, in some (but not all) of our protocols we will allow the communication between the prover and verifier to be long – e.g., of length polynomial in the input. (For example, this gives the prover the ability to resend the input to the verifier.) The verifier can access the communication tape in a similar read-once access to the way it accesses the input - i.e., it can read data but can only store as much as its memory allows. This should be contrasted with the standard interactive streaming model in which communication is typically poly-logarithmic.

Following the foundational work of Goldwasser, Micali, and Rackoff [GMR89], our goal for zero-knowledge in the streaming model is to ensure that everything observed by the verifier during its interaction with the prover can be simulated by a machine with similar computational resources. In other words, the verifier should be able to simulate its own view of the interaction.

Thus, loosely speaking, we say that a streaming interactive proof is zero-knowledge for space bound s, if for any space-s (malicious) verifier there exists a simulator, with comparable space to that used by the verifier, that generates a view that is indistinguishable to an external space-bounded distinguisher. Morally this implies that a bounded space verifier learns no more than what it could have generated by itself.

1.1.1 Differences from the [CDGH23] Definition

Next, we highlight some key ways in which our definition differs from that of [CDGH23].

On "White-box" Simulators. In the standard model, a very common approach for demonstrating zero-knowledge is via *rewinding*. However, in the streaming model, where the input is streamed only once, rewinding is infeasable, as the simulator has no opportunity to revisit the input and make additional attempts.

To overcome this difficulty, Cormode *et al.* [CDGH23, p. 21], allowed their simulator to have "white-box access" to the verifier. White-box access gives the simulator additional knowledge, *which may be hard to compute*, of the verifier's strategy. More specifically, they define white-box access to the verifier V^* as an oracle access to a function that takes as input a snapshot of V^*'s current state $b \in \{0,1\}^s$ and a candidate output $z \in \{0,1\}^m$, and returns the maximum probability over all inputs $y \in \{0,1\}^n$ with which V^* starting with state b outputs z.

In this work we handle the inability to rewind in a conceptually different, but technically similar, manner. Specifically, we model our simulators as *read-once branching programs* (ROBP). This is

[1] For messages that are short enough to be stored entirely in memory, it makes no practical difference whether they are streamed or sent all at once. However, the model allows for the exchange of messages that exceed the verifier's memory capacity, as long as it is accessed in a streaming manner.

[2] Although the streaming model can naturally be extended to handle search problems, where the verifier's output may be any string, in this work we focus exclusively on decision problems.

described in detail in Section 3.1. Branching programs are a well-studied computational model in the literature, that models non-uniform bounded space algorithms. In a nutshell, the power of branching programs over their uniform counterparts is that they can perform any desired local computation in order to transition from state to state, as long as the compuation does not involve the input. For example, if the algorithm is currently at some state $st \in \{0,1\}^s$, it can immediately transition to whatever state $f(st) \in \{0,1\}^s$ it wishes to, even if the function f is hard to compute, see [Gol08, Section 8.4] for additional details and motivation.

From the simulator's perspective the verifier is fixed and so any computation relative to the "code" of the verifier can be done for free when modeling the simulator as a branching program.[3] Moreover, modeling the simulator as a branching program means that we can demonstrate zero-knowledge against stronger malicious verifiers that are similarly modeled as ROBPs.

We mention however, that the technical way in which our simulators use the branching program abstraction is to achieve a similar goal as in the white-box model of [CDGH23]. However, we find the modeling via branching program to be more natural, see Section 2 for additional details.

Simulation Error and Space Overhead. The zero-knowledge definition of [CDGH23] only requires the distinguishing gap (i.e., the ability to disinguish a real interaction from a simulated one) to be any sub-constant function (i.e., $o(1)$). The gap actually achieved by their constructions is not explicitly specificed, but seems to be $1/\mathrm{polylog}(n)$ (and their techniques are inherently limited to a gap of at most $1/\mathrm{poly}(n)$). Thus, a malicious verifier could, with inverse-polylogarithmic probability, distinguish a real interaction from a simulated one.

This relatively large error can be extremely problematic. Not only is the error itself non-negligible but if one is to compose the protocol in a natural way (e.g., running it several times on different inputs), the simulation error would quickly grow and give a meaningless guarantee. Cormode *et al.* [CDGH23, Remark 4.4] point out that in the read-once setting the protocol will not be run again on the *same* input, but neither their definition nor their constructions protect provers that are involved in multiple distinct interactions.

In contrast, our zero-knowledge definition and constructions achieve a *negligible* simulation error. In addition, while Cormode *et al.* only consider malicious verifiers that use exactly the same space as the honest verifier, our constructions allow malicious verifier to use a larger amount of space (jumping ahead, our constructions guarantee zero-knowledge against malicious verifiers that use quadratically more space than the honest one).

These two improvements (i.e., negligible zero-knowledge error and handling quadratic malicious verifiers) come at a certain cost which makes our definition technically incomparable (and still, in our opinion, much preferable). Specifically, while Cormode *et al.* consider malicious verifiers that run in space that is the same as that of the honest verifier, they allow the *distinguisher* to use a polynomially larger space.

In contrast, we require the distinguisher to use roughly the same amount of space as the malicious verifier (i.e., quadratically more than the honest verifier). Since the distinguisher and verifier represent different aspects of the malicious verifier (i.e., during and after the interaction), we find this choice to be quite natural.

[3] Interestingly, this type of non blackbox use of the verifier is quite different from that in Barak's celebrated work [Bar01].

1.2 Our Results

We construct new zero-knowledge streaming protocols as described next.

zkSIP for Low-Depth NP Relations. Our main contribution is a general purpose zero-knowledge streaming interactive proof (zkSIP) for every NP relation that can be decided by a polynomial-size and polylogarithmic-depth (i.e., NC) circuit.

Furthermore, as alluded to above, our protocols achieve negligible zero-knowledge and soundness errors.

Theorem 1 (Informally stated, see Theorem 7.1)**.** *Let $\mathcal{R}$ be an* NP *relation decidable by a polynomial-size polylogarithmic-depth circuits. Then, there exists a streaming interactive proof for $\mathcal{R}$ with perfect completeness, negligible soundness error, honest verifier space $s = \mathrm{polylog}(n)$, and polynomial communication complexity. The proof is secure against adversaries with space at most $s^{2-\delta}$ (for any constant $\delta > 0$) with a negligible zero-knowledge error. Furthermore, the prover runs in polynomial-time given the* NP *witness as an auxiliary input.*

We find Theorem 1 striking, as previous work only provided constructions tailored to specific problems of interest. In contrast, our work presents a general-purpose construction of zero-knowledge protocols in the streaming setting for a *very* broad class of problems. This result is highly non-trivial even when restricted to low-depth computations, and becomes even more so when extended to non-deterministic computations.

In particular, as immediate corollaries one can derive alternate streaming zero-knowledge proofs for all problems considered by [CDGH23]. Moreover, as discussed above, the resulting protocol also achieves notions of security (albeit uses polynomially more communication).

As another concrete example of interest, captured by Theorem 1, consider the INTERSECTION problem (the complement of DISJOINTNESS). The input is $(x, y) \in \{0,1\}^n \times \{0,1\}^n$ and the goal is to accept if there exists some $i \in [n]$ such that $x_i = y_i = 1$. Theorem 7.1 yields a streaming zero-knowledge proof for this problem in which the verifier learns only that the two vectors intersect and nothing beyond that (in particular what is the intersection point).

zkSIP for the Polynomial Evaluation Problem (PEP). We also consider a specific problem that was studied by [CDGH23] called the "polynomial evaluation problem". The input in this problem is a multilinear polynomial $P : \mathbb{F}^m \to \mathbb{F}$ followed by an evaluation point $x \in \mathbb{F}^m$ and value $\alpha \in \mathbb{F}$. The goal is for the verifier to check that $P(x) = \alpha$. The core challenge arises from the fact that the point of evaluation is provided only *after* the polynomial has been streamed. The interest in this problem stems from the fact that it generalizes some key problems from the streaming literature, including the central "Index" Problem (in which the goal is, given x and then i, to determine x_i).

We give a direct protocol for PEP, which builds on the protocol of [CDGH23], but improves several key aspects.

Theorem 2 (Informally stated, see Corollary 6.3)**.** *There exists a streaming interactive proof for the polynomial evaluation problem (*PEP*) where the honest verifier's space s as well as the total communication complexity is polylogarithmic in the input length. The proof has perfect completeness, negligible soundness error, and negligible zero-knowledge error against adversaries with space $s^{2-\delta}$ (for any $\delta > 0$).*

We highlight several aspects in which our construction improves upon that of [CDGH23].

1. **Improved Communication Complexity:** The protocol for the PEP by Cormode *et al.* requires superlinear communication, whereas our protocol achieves *polylogarithmic* communication complexity. This resolves [CDGH23, Open Problem 2].[4]

2. **Stronger Security Guarantees:** The protocols of [CDGH23] have an $o(1)$ completeness, soundness and zero-knowledge errors. The $o(1)$ terms are not specified explicitly, but in the current constructions are $1/\mathrm{polylog}(n)$ and it seems that their approach is fundametally limited at $1/\mathrm{poly}(n)$ error.

   In contrast, our protocols have no completeness error and negligible soundness and zero-knowledge error. This resolves [CDGH23, Open Problem 4].

3. **Simpler and Modular Constructions:** Our protocols are designed to be modular, using our commitment schemes as black boxes. This modularity simplifies both the construction and analysis of the protocols, enhancing their understandability. The high-level structure is borrowed from [CDGH23], but the actual implementation of their strategy is tailored to the specific commitment schemes that they use.

1.2.1 New Streaming Commitment Schemes

The security of our zkSIPs relies on two novel streaming commitment constructions that we construct, and may be of independent interest. Our commitment schemes are quite different from analogous schemes presented in [CDGH23], and the improvemed parameters that we achieve in Theorem 2 stem from the improved commitment schemes.

Both of our commitment protocols achieve hiding and binding with *information-theoretic* security, without relying on any complexity-theoretic assumptions (such as the existence of one-way functions). Instead, they depend only on the space limitations imposed on one of the parties by the underlying streaming model. We elaborate on these commitments next.

Statistically Hiding Streaming Commitment. This interactive protocol allows a streaming sender to commit to a value with a streaming receiver. The scheme is secure against unbounded malicious receivers and against malicious senders whose space is at most quadratic relative to that of an honest sender.

Theorem 3 (Informally stated, see Theorem 4.2)**.** *Let n be the security parameter and let $\epsilon > 0$. There exists an interactive, computationally binding, statistically hiding, streaming string commitment, where the sender and receiver require space $O(n)$ and $O(n^{1+\epsilon})$, respectively. The scheme has negligible binding error against adversarial senders with space at most $O(n^2)$, and a negligible hiding error against unbounded receivers. The communication complexity is $O(n^2)$.*

Statistically Binding Linearly Homomorphic Streaming Commitment. This *non-interactive* protocol allows a sender to commit to a vector v and later de-commit to a *linear function of v*, as

[4] Here and below, we remind the reader that our model of zero-knowledge streaming algorithm is technically incomparable to that of [CDGH23]. Still we believe that our model captures the underlying questions and resolves them in a satisfactory way.

specified by the receiver. The scheme guarantees negligible hiding and binding errors, even against receivers with quadratic space and unbounded senders.

Theorem 4 (Informally stated, see Theorem 5.8)**.** *Let n be the security parameter, and let $\epsilon > 0$. Then, there exists a linearly homomorphic, statistically binding, streaming commitment, where the sender and receiver require space $O(n^{1+\epsilon})$. The scheme has negligible hiding error against adversarial receivers with space at most $O(n^2)$, and a negligible binding error against unbounded senders. The communication complexity is $O(n^2)$.*

1.3 Organization

We begin in Section 2 with the technical overview. In Section 3, we formally introduce the computational models used throughout the paper, define streaming interactive proofs and zero-knowledge in the streaming setting. Sections 4 and 5 introduce the definitions and constructions of our streaming commitment schemes, which form the core building blocks for the zkSIPs for PEP and low-depth NP relations. Finally, Sections 6 and 7 present these protocols and prove their correctness.

2 Technical Overview

To present our techniques, it is instructive to first present our improved protocol for the Polynomial Evaluation Problem, also known as PEP (see Theorem 2). Recall that in PEP, the input is a multivariate low degree polynomial $P : \mathbb{F}^m \to \mathbb{F}$ (over a sufficiently large finite field $\mathbb{F}$), a point $x \in \mathbb{F}^m$ and a value $\alpha \in \mathbb{F}$ and the goal is to verify that $P(x) = \alpha$. In the streaming setting this is challenging, since the streaming algorithm first sees P and only then sees the evaluation point x.

Motivated by an easy reduction from the central *Index* Problem, PEP was first introduced in [CCM+19], who also showed a streaming interactive protocol for it (which is not zero-knowledge). Cormode *et al.* [CDGH23] developed a zero-knowledge variant by integrating commitment schemes into the original protocol. In this work, we improve upon their zero-knowledge construction by constructing new commitment schemes tailored to the streaming model, which we incorporate into the protocol of [CDGH23]. Later, we show how we then further extend the techniques to construct the general purpose protocol of Theorem 1.

We start by reviewing the known protocols for PEP and then proceed to describe our improvements. In Sections 2.1 and 2.2, we introduce our new commitment constructions, and in Section 2.4, we detail how these techniques are used to construct a zkSIP for every low-depth NP relation.

A Streaming Interactive Proof for PEP. In this protocol, the prover and verifier stream (i.e., read symbol-by-symbol) the input simultaneously. While the input is being streamed, the verifier computes the value of the polynomial $P : \mathbb{F}^m \to \mathbb{F}$ at a hidden random point $r \in \mathbb{F}^m$. Once the input stream is complete the verifier gets the second part of the input, which is the point $x \in \mathbb{F}^m$ and value α (recall that it needs to verify that $P(x) = \alpha$). The verifier sends to the prover a line $L : \mathbb{F} \to \mathbb{F}^m$ that passes through x at 0 and through r at some random point $\mu \in \mathbb{F}$ (i.e., $L(0) = x$ and $L(\mu) = r$). The prover responds with the restriction of P to L, which results in a univariate polynomial $g(\cdot) = P(L(\cdot))$. The verifier then checks that $g(0) = \alpha$ and that $g(\mu) = P(r)$.

Since the prover is unaware of r and μ, any attempt to deceive the verifier – by sending a polynomial $g' \neq g$ when $P(x) \neq \alpha$ – can be easily detected by the verifier. To see this, recall that if P has total degree d then both g and g' are distinct degree-d univariate polynomials and so can agree on no more than d points. Thus, if $g' \neq g$, the probability that $g'(\mu) = g(\mu) = P(L(\mu)) = P(r)$ is very small. As a result, the verifier will detect the inconsistency with high probability and reject the prover's response – thus preserving the soundness of the protocol. This means that the prover must compute g truthfully, and the verifier can check that $g(0) = \alpha$.

However, this protocol is not zero-knowledge. For example, Consider a verifier who, given a PEP instance (P, x, α), wants to determine the value of the polynomial P at $x + 1$. Evaluating P at $x + 1$ is at least as hard as evaluating it at x, which is not feasible for a streaming verifier. Now, consider the following attack on the protocol described above. While streaming the input, the verifier ignores P and only records the value of x at the end of the stream. Then, it sends the prover the line L that connects x and $x + 1$. The prover responds with the restriction of P to L, thereby revealing the value of P at $x + 1$ to the verifier. Thus, the verifier learns information it could not have obtained on its own, demonstrating that the protocol is not zero-knowledge. Note that while this demonstrates that the protocol is not zero-knowledge against adversarial attacks, it is not clear that even the *honest* verifier does not learn anything from the interaction, albeit inadvertently.

A Zero-Knowledge Protocol for PEP. We outline the method used in [CDGH23] to transform the above protocol into a zkSIP. Their approach follows a well-established protocol structure using commitments, originally introduced in [GK96] for interactive proofs in the standard model, and proceeds as follows.

- Verifier's Commitment. The verifier begins by committing to all the randomness it will use throughout the protocol.
- Prover's Commitment. The prover then sends a commitment to a message.
- Verifier's De-commitment. The verifier de-commits its randomness.
- Prover's De-commitment. After verifying the verifier's de-commitment, the prover de-commits the previously committed message.
- Verifier's Output: The verifier uses the de-committed information to compute its output.

In the PEP protocol, before the input stream begins, the verifier commits to the point r where it will evaluate the input polynomial. After the stream is completed, the verifier sends to the prover the line L connecting x and r, and the prover responds with a *commitment* to the restricted polynomial $g = P|_L$. The verifier then de-commits to r, and if the de-committed value is consistent with L, the prover de-commits to g, revealing the values the verifier needs to perform its checks. The verifier's commitment to r ensures two things: first, the verifier cannot retroactively choose a line L that aligns with the input as part of an attack, and second, the value of r remains hidden from the prover until it has already committed to g, which guarantees the soundness of the protocol.

Unfortunately, a somewhat subtle detail in the protocol described above prevents us from being able to prove that it adheres to the definition of zero-knowledge in the streaming model despite the limitations imposed on the verifier. Note that at the end of the protocol, the prover reveals the polynomial g, which includes evaluations of many other points aside from x and r. While these points are on a line that cannot be predicted by the verifier (as it is unaware of x at the time of choosing r) and it may seem that learning the points on a fairly random line is something that the verifier can do by itself, yet the fact that the line goes through the point x makes the task of showing the verifier can indeed learn the points on this line much more challenging.

To circumvent this, we have the prover commit to g using a special commitment scheme, where it can de-commit to specific points in g and not reveal the entire polynomial. This way, the only points revealed to the verifier are the points from the input whose values it already knows, i.e. $P(x)$ and $P(r)$.

To conclude the discussion so far, the strength of the PEP protocol is fundamentally tied to the robustness of its commitment schemes. Cormode *et al.* instantiate the above framework using commitment schemes that they develop. In this work we substantially improve these commitments schemes, which will lead us to the improved PEP protocol (and will also be used in our protocol for general purpose computation). Our main contribution lies in constructing more secure and efficient commitment schemes, which form the backbone of our improved protocol. These stronger commitments not only enhance security but also reduce communication complexity.

We next describe our new commitment constructions and then return back to establishing security of the PEP protocol.

2.1 Statistically Hiding Streaming Commitment

We present our construction for a statistically hiding streaming commitment. Recall that in such a commitment the hiding property should hold unconditionally, whereas the binding condition need only hold for a sender with bounded memory. The commitment constructions presented in this and the subsequent section draw inspiration from the work of [GZ19]; however, our analysis takes a markedly different approach.

We proceed to the implementation of our commitment scheme, focusing for simplicity on the case of bit commitment (i.e., committing to just a single bit). Let n denote the security parameter. We shall aim to construct a scheme in which the honest parties use roughly n space, but security holds against adversaries that use roughly n^2 space.

The sender draws a random vector $s \in \{0,1\}^{4n}$, which serves as the commitment key. The receiver sends a matrix $A \in \{0,1\}^{4n \times n}$ as a stream to the sender row by row. The receiver computes $\alpha = s^T A \in \{0,1\}^n$. Notice this can be done in small space by computing $s^T A = \sum_{i \in [4n]} s_i \cdot a_i$ as an accumulative sum during the streaming of A, where s_i is the i-th bit of s and a_i is the i-th row of A. Now observe that:

1. A is too large for the sender to hold in memory in its entirety, and
2. The vector $\alpha = s^T A \in \{0,1\}^n$ is shorter than s.

To commit to a bit $b \in \{0,1\}$, the sender sends (r, α, β), where $r \in \{0,1\}^{4n}$ is a random vector of the same length as the key s, and $\beta = \langle s, r \rangle \oplus b$ (where $\langle x, y \rangle = \sum_i x_i \cdot y_i \pmod 2$ denotes the inner-product operator).

To de-commit, the sender sends s, and the receiver verifies that $\alpha = s^T A$ and computes the value of b as $\beta \oplus \langle s, r \rangle$.

The statistical hiding property of the commitment results from the fact that α is shorter than s, and so the entropy of s is high from the receiver's perspective. As such, by the Leftover Hash Lemma [HILL99], the statistical distance between (r, α, β) and (r, α, ρ), where ρ is a random bit, is small, and the latter contains no information about b.

Observe that to break the binding property, the sender would need to find $s \neq s'$ such that $\alpha = s^T A = (s')^T A$, which is equivalent to identifying a vector s'' such that $(s'')^T A = 0$. The binding property of the commitment, therefore, stems from the challenge the sender faces in locating a vector within the kernel of A, given streaming access to it.

We prove that a space-bounded machine cannot find such a vector (with non-negligible probability) via a series of reductions from the problem of solving systems of linear equations. Eventually, we utilize Raz's [Raz18] celebrated result that bounded width branching programs cannot solve linear equations (on the average).

Observe that a straightforward implementation of the commitment, as described above, would require the receiver to store the entire matrix A in order to verify that $\alpha = s^T A$ during the de-commitment phase. However, since we intend to use this commitment scheme in our zero-knowledge constructions, it is important to simulate both the sender and the receiver within limited space, which means storing the entire matrix A is not feasible.

To circumvent this issue, we have the receiver draw a small matrix $T \in \{0,1\}^{n \times \lambda}$ (where λ is a secondary security parameter, significantly smaller than n), and as it streams A to the sender, it computes $AT \in \{0,1\}^{n \times \lambda}$, which is significantly smaller than A. When the sender sends $\alpha = s^T A$, the receiver computes αT, and when the sender de-commits by sending the key s, the receiver checks that $\alpha T = s^T AT$, which can be done in small space.

As noted earlier, breaking the binding of the commitment requires the sender to locate a vector in the kernel (null-space) of a matrix. With our recent modification, this matrix has an expanded null-space because, although the number of rows in AT and A is similar, AT has fewer columns than A. This enlargement of the null-space makes it easier for a malicious sender to find such a vector, thereby weakening the binding property of the commitment.

This situation highlights the trade-off between the receiver's space requirements and the binding security of the scheme, which is adjustable through our construction's parameters. By carefully selecting these parameters, we can achieve a negligible binding error while maintaining minimal space requirements for the receiver.

To extend from a single-bit commitment to an ℓ-bit string commitment, one approach is to commit bit-by-bit. However, to avoid streaming a new matrix A for each commitment, we adopt a more efficient method. We use a matrix $S \in \{0,1\}^{\ell \times 4n}$ as the key, with each row of S acting as a distinct bit key s. This allows us to commit to an entire ℓ-bit vector using a single matrix A. However, because the sender needs to store the entire matrix S for de-commitment, it is important to keep S at a manageable size.

2.2 Statistically Binding Streaming Commitment

Our second commitment construction is a linearly homomorphic vector commitment, allowing the sender to commit to a vector $v \in \{0,1\}^\ell$ and then selectively de-commit to a linear function $u = L(v) \in \{0,1\}^j$ of the vector, revealing no information beyond the fact that the input belongs to the subspace $L^{-1}(u)$. Binding should be statistical and hold even with respect to an unbounded sender.

We proceed to describe our construction of the statistically binding linearly homomorphic commitment in more detail.

Denote the security parameter by n and let $\ell \ll n$ denote the length of messages to which the sender wants to commit. The sender starts by generating a random matrix $K \in \{0,1\}^{n \times \ell}$ whose number of columns is equal to the length of the vector $v \in \{0,1\}^\ell$ to which the sender commits. The sender streams a large, random matrix $A \in \{0,1\}^{n\times n}$ to the receiver, computing $AK \in \{0,1\}^{n\times \ell}$ simultaneously during the streaming process. The space-bounded receiver, who cannot store the entirety of A, chooses a small secret matrix $T \in \{0,1\}^{\lambda\times n}$ (where once again, λ is a secondary security parameter, much smaller than n), and while streaming A it computes $TA \in \{0,1\}^{\lambda\times n}$. After the streaming of A has completed, the sender sends as a stream to the receiver the tuple (r, K, β), where $r \in \{0,1\}^n$ is a random vector and $\beta = r^T AK + v^T \in \{0,1\}^\ell$. The receiver stores r, β and computes $TAK \in \{0,1\}^{\lambda\times\ell}$.

To de-commit to $v^T L$, where $L \in \{0,1\}^{\ell\times j}$ is the matrix representing the linear function applied to the committed vector, the sender transmits $AKL \in \{0,1\}^{n\times j}$. The receiver checks that $T(AKL) = (TAK)L$ and recovers $v^T L$ by computing $\beta L - r^T AKL \in \{0,1\}^j$. Note that L can be known to both parties in advance or sent by the receiver prior to the de-commitment stage.

Notice that the de-commitment key is given by AK, with both A and K sent to the receiver during the commit stage. Despite this, the commitment remains hiding because the receiver is space-bounded and cannot store all of A. Thus, intuitively, after streaming K, the entropy of the key AK, from the receiver's perspective, remains high.

In our analysis, we show that given any bounded-space, malicious receiver strategy, the entropy of AK is high in expectation over A, and so, using an analysis inspired by the proof of the Leftover Hash Lemma [HILL99], we show that the receiver's view of the commitment is statistically close to a random string.

Recall that the hiding requirement for this commitment is stronger than the standard hiding property, as the scheme must ensure that after the de-commitment, no information regarding the committed vector's projection on the subspace orthogonal to L has leaked. To prove that our construction satisfies this property, we show a reduction from the conventional hiding property. Specifically, if there exists an adversary that can break the hiding property of the commitment to a vector of length ℓ, after receiving a de-commitment to its projection on a subspace of rank j, then we can construct an adversary that would break the hiding property of the commitment *prior* to the de-commitment on vectors of length $\ell - j$.

The binding property of the commitment is ensured by the fact that the matrix T, selected by the receiver, remains concealed from the sender throughout the protocol. Compromising this binding would require finding a matrix with the dimensions of AKL that lies within the null-space of T, which is infeasible from an information-theoretic perspective.

Similar to the statistically hiding commitment, there is a trade-off between the receiver's storage capacity and the binding error of the commitment. By carefully choosing the parameters, we can minimize the binding error while maintaining minimal space requirements for the receiver.

Polynomial Commitment. Building on the foundation of the linearly homomorphic commitment, we construct a polynomial commitment, where the sender commits to a polynomial and can later de-commit to evaluations of the polynomial at specific points based on the receiver's queries. This polynomial commitment is essentially a reinterpretation of the linearly homomorphic commitment, with the committed vector representing the polynomial (by either evaluations or coefficients) and polynomial evaluations corresponding to linear functions of this vector.

Note that the linearly homomorphic commitment is defined over the binary field. To use it as a polynomial commitment for polynomials over larger fields, we focus on extension fields of the binary field. This approach works because a linear transformation on a vector over an extension field can be represented as a linear transformation on the binary representation of the vector over the binary field. See Section 5.2 for details.

2.3 Using Commitments for Zero-Knowledge

Consider the PEP protocol described in Section 2 above, and observe that the final prover message includes the value $g(\mu) = P(r)$ where P is the input polynomial. A simulator sending an incorrect value would lead the verifier to reject, enabling a distinguisher to easily differentiate between the real execution and the simulated one. Thus, the simulator must be able to reproduce the value $P(r)$ during the simulation. The challenge is that the verifier only reveals r *after* the input was streamed.

The one power that the simulator has over the prover, is that it has the code (and randomness) of the verifier. So for example, for a rather "tame" verifier that just copies r from its random string, the simulator can select r by itself and place it in the correct position in the verifier's random string.

However, when dealing with an *arbitrary malicious* verifier, the selection of the value r might involve an extremely complex function that is obfuscated within its code. This is where we will take advantage of the fact that we model our simulator as a read-once branching program.

Recall that an ROBP can effectively compute any function that does not depend on the input. Thus, we need to show that there exists a well-defined function that maps a state in the simulation *prior* to the input stream, to a candidate value r', such that, with high probability, $r' = r$. If the commitment were statistically binding, we could just argue that the commitment fully determines the string r. Unfortunately, as the commitment is only computationally binding (wrt bounded space algorithms), the value committed to is *not* determined just by the commitment.

Thus, we would like to use the code of the verifier to *extract* the value of r, by continuing its execution. The challenge is that in order to continue the execution we need to provide the verifier with the input, but the function that we currently computing *cannot* access the input.

In order to construct such a function, we propose the following. Consider a table where each column represents a simulation state just after the commitment to r, and each row represents a possible interactive protocol transcript that follows the commitment up to the point of de-commitment. Note that, in particular, the rows encompass *every possible input to the protocol.* Each entry in the table will contain the value to which the verifier de-commits, based on the initial state represented by the column and the protocol transcript represented by the row. The objective is to assign each *column* of the table (which is our function's input) a single value r', such that for every given row, with high probability over selecting a column, r' is consistent with the value of the corresponding table entry.

We define the function using the following (inefficient) procedure:

1. Populate the table as outlined earlier by executing the protocol for every possible input and randomness.

2. Carefully select a small subset of rows I from the table, and determine the value of r' for each column based on one of the rows in I.

We use the binding property of the commitment to show that this construction meets its objective. Specifically, with high probability over the execution of the commitment to r, it holds that $r' = r$, independent of the actual input. For a detailed description and analysis, refer to Section 6.2.1.

2.4 Zero-Knowledge for Low-Depth NP relations

We proceed to describe our proof of Theorem 1 – a general purpose zero-knowledge streaming interactive proof for any bounded-depth NP relation. Our strategy for doing so is to start with an existing zero-knowledge proof in a different model, and show how to compile it, using the commitment schemes described above, into a zero-knowledge proof in the bounded streaming model.

More specifically, the model that we consider is that of an *interactive PCP* [KR08]. A proof-system in the IPCP model consists of a prover that first sends a PCP to which the verifier has oracle access. The prover and verifier then exchange messages (like in an interactive proof), and finally, based on its own randomness, the input, the prover's messages, and the answers to its PCP queries, the verifier decides whether to accept or reject. We will use an IPCP construction, due to Chiesa *et al.* [CFS17] (building on the doubly-efficient interactive proof of [GKR15]) which has a few properties that will be crucial for us:

1. The protocol has a verifier and simulator that use small space.
2. The interaction is of the public-coin form.
3. The verifier is "holographic" which, loosely speaking, means that it only needs to compute linear functions of the input (specifically evaluating the input's low-degree extension).
4. Lastly, the queries that verifier makes to both the input and PCP are non-adaptive – their locations depend only on the verifier's randomness (and not on responses to previous queries nor any other message sent by the prover).

This protocol structure adapts naturally to the streaming setting. As the input is streamed, the streaming verifier can compute the relevant linear functions that it needs to simulate. Instead of providing the PCP as an oracle-accessible string, the streaming prover streams the PCP directly to the verifier, who then records values at the query locations specified by the IPCP verifier. The interactive component and the verifier's decision process remain unchanged.

The zero-knowledge property in the IPCP constructions of [CFS17] hinges on a key aspect of algebraic complexity: as long as the number of verifier's queries to the PCP remains below a certain threshold, the values at unqueried points retain randomness. Thus, limiting the number of queries, even that of a *malicious verifier*, is crucial for preserving the protocol's zero-knowledge property.

Therefore, while it is clear that the above straightforward transformation of the IPCP protocol to a streaming setting preserves the original protocol's completeness and soundness, it also becomes evident that the resulting protocol loses its zero-knowledge property–even if the original IPCP was zero-knowledge. This loss occurs because, in the streaming model, the verifier has full access to the PCP, and the verifier's limited space and method of access is not enough to prevent it from extracting valuable information.

To address this issue, we incorporate our commitment schemes into the protocol similarly to our zero-knowledge streaming interactive proof for PEP. Specifically, rather than sending the PCP in the clear, the prover sends a *commitment* to the PCP, committing to each symbol individually. When the verifier issues a query, the prover responds by de-committing only to the specified queried locations. Due to the commitment's hiding property, this approach ensures that no information beyond the de-committed values is revealed.

This method naturally limits the number of queries the verifier can make, as the prover supplies the responses, eliminating the need to impose artificial restrictions on a malicious verifier, such as a query limit. This approach allows us to leverage the zero-knowledge guarantees of a query-bounded IPCP verifier without adding extra constraints. From a soundness perspective, the binding property of the commitment ensures that the prover cannot modify the PCP values after learning the verifier's query locations.

This concludes the high-level strategy. We proceed to a more detailed description below.

The IPCP-to-Streaming Compiler. Given a k-round IPCP $\langle \mathcal{P}_{\text{IPCP}}, \mathcal{V}_{\text{IPCP}} \rangle$, we construct a streaming protocol $\langle \mathcal{P}_{\text{S}}, \mathcal{V}_{\text{S}} \rangle$, where $\mathcal{P}_{\text{S}}$ and $\mathcal{V}_{\text{S}}$ will use $\mathcal{P}_{\text{IPCP}}$ and $\mathcal{V}_{\text{IPCP}}$, respectively, as black boxes.

The verifier $\mathcal{V}_{\text{S}}$ sets the randomness $r = \{r_i\}_{i=1}^k$ for $\mathcal{V}_{\text{IPCP}}$ and, as in our protocol for PEP, commits to r in the first stage using a statistically hiding streaming commitment. Since we consider a public-coin, non-adaptive verifier, the randomness r determines the query locations for both the input and the PCP of the IPCP verifier. Consequently, we assume that $\mathcal{V}_{\text{S}}$ knows the query points that $\mathcal{V}_{\text{IPCP}}$ will use.

The input is then streamed. During the streaming of the input, $\mathcal{V}_{\text{S}}$ will save the values from the input in the locations where $\mathcal{V}_{\text{IPCP}}$ will query during the protocol. Following the input stream, $\mathcal{P}_{\text{S}}$ sends to $\mathcal{V}_{\text{S}}$ a statistically binding streaming commitment to the PCP.

The streaming prover and verifier then simulate the interactive phase of the IPCP as follows. In each round $i \in [k]$ of the interaction, the verifier $\mathcal{V}_{\text{S}}$ sends a de-commitment to the randomness r_i to which it committed in the beginning. The streaming prover $\mathcal{P}_{\text{S}}$ forwards r_i to $\mathcal{P}_{\text{IPCP}}$, which responds with the next message β_i and sends it to the verifier.

Since the verifier's randomness determines the PCP query locations, the prover knows all query points by the end of the IPCP interaction and sends a de-commitment to $\mathcal{V}_{\text{S}}$ for each relevant PCP evaluation. Consequently, when $\mathcal{V}_{\text{IPCP}}$ issues its queries, it receives the answers directly from $\mathcal{V}_{\text{S}}$.

The Zero-Knowledge Simulator. Our goal is to demonstrate that the compiler preserves zero-knowledge; that is, if the original IPCP protocol is zero-knowledge, then the resulting streaming proof also retains zero-knowledge. Let $\mathcal{S}_{\text{IPCP}}$ denote the zero-knowledge simulator for the IPCP protocol. We construct a streaming simulator that interacts with a malicious streaming verifier $\mathcal{V}_{\text{S}}^*$, utilizing $\mathcal{S}_{\text{IPCP}}$ as a component.

The simulator first engages with $\mathcal{V}_{\text{S}}^*$ to obtain the commitments to $\{r_i\}_{i=1}^k$, which correspond to the messages from $\mathcal{V}_{\text{IPCP}}^*$ during the interactive phase of the IPCP protocol. By applying the same extraction technique used in the PEP simulator, it retrieves the values $\{r_i'\}_{i=1}^k$, which, with high probability, satisfy $r_i' = r_i$ for all $i \in [k]$. These extracted values determine two sets of query points: I_{pcp}, where the IPCP verifier queries the PCP, and I_{inp}, where it queries the low-degree extension of the input. During the input stream, the simulator computes and stores the values of the input's low-degree extension at the points in I_{inp}.

The simulator now emulates $\mathcal{S}_{\text{IPCP}}$ as follows. Whenever $\mathcal{S}_{\text{IPCP}}$ expects the i-th message from $\mathcal{V}_{\text{IPCP}}^*$, the simulator supplies the previously extracted value r_i'. Each time $\mathcal{S}_{\text{IPCP}}$ provides an answer to a PCP query or a simulated prover message β_i, the simulator records that value.

Using the stored values from the answers to the PCP queries, the simulator generates commitments to the values of a PCP π that match all responses given by $\mathcal{S}_{\text{IPCP}}$ to the queries from $\mathcal{V}_{\text{IPCP}}^*$. Note that all other values in the PCP can be arbitrary, and the simulator does not need to store commitment data for them, as it will not de-commit to those values later in the simulation.

After simulating the PCP commitment, the simulator proceeds to emulate the interactive phase of the streaming protocol as follows. For each $i \in [k]$, the simulator receives the de-commitment from $\mathcal{V}_{\mathsf{S}}^*$ for r_i. If $\mathcal{V}_{\mathsf{S}}^*$ fails to de-commit or if $r_i \neq r_i'$, the simulation is aborted. Otherwise, the simulator records β_i on the output tape. In the final step, the simulator writes to the output tape the de-commitments for all PCP points in I_{pcp}.

Extending to Non-deterministic Low-depth Circuits. The result by Chiesa *et al.* [CFS17, Theorem 11.1] provides a zero-knowledge IPCP for the GKR protocol, covering languages in NC. We observe that this result can be extended to an IPCP for NP with similar properties.

The main idea is straightforward: the extension to NP follows naturally once we recognize that the verifier in the protocol for NC only requires access to the low-degree extension of the input to the circuit. In the case of NP, the input to the circuit also includes the witness, which must remain hidden from the verifier in a zero-knowledge setting.

Therefore, it is natural for the prover to encode the witness and integrate it into the PCP. Special attention must be given to encoding the witness to ensure that the low-degree extension does not reveal any confidential information. To achieve this, we utilize techniques from [BSCF+17, CFS17].

Once the IPCP for an NP relation is established, we can apply our compiler to the protocol to obtain the desired streaming protocol.

3 Streaming Zero-Knowledge: Models and Definitions

3.1 Computational Models

In this work we consider two computational models that naturally capture a bounded verifier in the interactive streaming model, a *read-once interactive Turing machine* and a *read-once interactive branching program.*

Read-Once Interactive Turing Machine (ROTM). A read-once interactive Turing machine with space $s(n)$ consists of a constant size state automaton and has access to the following five tapes:

1. Input tape. Read-only tape, reading head starts at the left-most cell and may only move right. At the start of the protocol, the input $x \in \{0,1\}^n$ is written on this tape.

2. Randomness tape. Read-only tape, reading head starts at the left-most cell and may only move right. At the start of the protocol, a string of the appropriate length is drawn at random and written on this tape.

3. Message-receive tape. Read-only tape, reading head starts at the left-most cell and may only move right. Each received message is written to the right of the previous message.

4. Message-send tape. Write-only tape, reading head starts at the left-most cell and may only move right.

5. Work tape. Read and write tape. The tape consists of s cells, and the head may move left and right.

The input tape, the message-receive tape, and the message-send tape are mutual to both interacting parties (where the message-receive tape of one party is the message-send tape of the other and vise versa). Each next bit in those tapes will be written/read simultaneously by both parties when both parties reach the appropriate state.[5] The computation halts when the machine reaches one of two halting states: Accept or Reject.[6] Note that at each computation step, the machine accesses (reads/writes) only a single tape.

In this work, we analyze streaming algorithms and protocols, where the machine is constrained by limited memory and cannot store the entire input. Consequently, our primary focus is on the machine's space complexity, which, in the ROTM model, corresponds to the number of cells on the work tape. Another important resource in interactive protocols is communication complexity. In the context of the ROTM model, this is measured as the total length of the message-read and message-write tapes. These two factors – space and communication complexity – are central to our study of efficient protocols under constrained resources.

Other resources such as the time, rounds, and randomness complexities are of secondary importance in this model. Nevertheless, we aim for our constructions to be efficient in these measures as well.

Read-Once Interactive Branching Program (ROBP). A read-once branching program P is a layered directed graph associated with a binary read-only tape of length n. The graph consists of n edge layers. The edges in the first layer represent the outgoing edges from the start node s, while the edges in the n-th layer lead into one of two final nodes, t_a (accept) or t_r (reject). Each layer i of the graph is associated with the i-th cell of the read-only tape, and the edges are labeled either 0 or 1.[7]

For a binary string $\sigma = \sigma_1 \dots \sigma_n$, a corresponding path in the graph starts at the node s and, for each layer i, follows the edge labeled by σ_i. We say that P accepts σ if the path for σ terminates at t_a.

A read-once randomized branching program is a branching program where the read-only tape contains a string pair (x, r) where x is the input and r is a random string. The bits of x and r may be interwoven with one another in a predetermined manner, yet maintain their internal order. We say that P accepts x with probability p if P accepts (x, r) with probability p over a random selection of the string r.

A k-round read-once interactive branching program is a randomized branching program where the read-only tape contains the string tuple $(x, r, \{\beta_i\}_{i=1}^{k})$ where x is the input, r is the random string and $\{\beta_i\}_{i=1}^{k}$ is a set of k messages received from an external party E. The bits of $x, r, \{\beta_i\}_{i=1}^{k}$ may be interwoven into one another. In the beginning of the computation only x, r are written on the tape and blank spaces are left for the β's. There are $k + 1$ graph layers which are message-send layers, all of whose nodes include a message string. Upon arrival on the i-th message-send layer, the message in the node is sent to E, and the bits of β_i sent from E appear on the read-only tape. Note that the first and last messages of P may be empty.

[5] This prevents situations such as that one party reads the input and then sends a message, where the other party waits for the message and only then reads the input.

[6] This definition can easily be extended to a machine that outputs a string by adding a write-only output tape; however, we do not utilize this extension in this work.

[7] Read-Once Branching Programs (ROBPs) are typically defined in a more general manner, where they are not necessarily layered. However, the construction described here corresponds to an oblivious, layered ROBP, also known as an Ordered Binary Decision Diagram (OBDD). For simplicity, we will refer to this as an ROBP throughout.

In this work we also use a non-interactive read-once randomized branching program that outputs a string. This model also has an output tape and some graph nodes may include a string. When reaching a node with a string in the computation path, the string is concatenated to the current content of the output tape. The output of the program is the content of the output tape at the end of the program.

Note that the width of an ROBP, defined as the maximum number of nodes in any layer, represents the number of possible states the machine can be in after reading each input bit. Therefore, limiting the width of an ROBP effectively restricts its available memory, analogous to constraining the size of the work-tape of the ROTM. Since this work focuses on small-space algorithms, we consider narrow-width ROBPs as a fitting model to capture these space limitations. Intuitively, and this will be established in Lemma 3.3, one should think of the "space" of a ROBP computation as logarithmic in the ROBP's width.

Remark 3.1 (ROBP-ROTM Interaction in Protocols)**.** *Throughout this work, we sometimes consider cases where one party in the interaction is a ROTM and the other is a ROBP. In this case, each input bit will be read simultaneously by both parties, and each bit that the ROTM writes on the message-send tape is simultaneously read by the ROBP. When the ROBP reaches a message-send layer, the message in the node is written on the message-receive tape of the ROTM, and the ROTM may not access the input tape or the message send tape until reading the message from the ROBP in full.*

Remark 3.2 (Simulation of a ROBP by another ROBP)**.** *While simulating one Turing machine with another is a common and well-understood practice, the concept of a branching program simulating another branching program is less familiar. Therefore, we clarify the concept. We can conceptualize a branching program that simulates another branching program as a collection of programs, where each individual program in the collection corresponds to a specific instance of the simulated program. Let $w_{\mathcal{S}}$ represent the width required by the simulator to handle computations outside the scope of the simulation (e.g., processing the output of the simulated program), and let w_P denote the width of the simulated program itself. It follows straightforwardly that the total width of the resulting simulator instance is at most $w_{\mathcal{S}} \cdot w_P$.*

The Power of ROBP vs ROTM. An ROBP is a non-uniform model of computation, consisting of a family of programs designed for specific input lengths. Its power lies in its ability to compute functions of unbounded complexity after reading each bit of the input, provided that the the result of the computation has small size (logarithmic in the ROBP's width). For instance, the transition between two adjacent nodes in the ROBP's state graph can represent any two (short) states of a ROTM's work tape, even if the ROTM would require substantial computational effort to transition between those states, potentially passing through much longer intermediate states. This allows the ROBP to compute virtually any function in order to transition between states, as long as the computation depends only on the current state and the single next input symbol that is read (and the result remains compact).

The relationship between the space of a ROTM and the width of a ROBP is captured in the following lemma.

Lemma 3.3. *Let $\mathcal{L}$ be decidable by an ROTM M with space $s = s(n)$. Then, $\mathcal{L}$ is decidable by a ROBP with layer width $2^{s+\log(s)+\log(c)}$, where c is the number of states in the automaton of M.*

Proof. Construct a ROBP P where the nodes between in each edge layer are all the possible states of M. There are exactly $c \cdot s \cdot 2^s$ such states. Connect each node in a node layer to the two nodes in the following node layer that represent the state of M just before reading the next input (or randomness or message from other party) bit. □

All of our constructions feature bounded-space verifiers using the weaker model of ROTMs, which given Lemma 3.3 is preferable due to their simplicity and uniformity. However, for security purposes, we aim to defend against adversaries modeled as space-bounded ROBPs, which, as shown by Lemma 3.3, is a stronger adversarial model and thus offers a better security guarantee. In the context of zero-knowledge proofs we therefore model our malicious verifiers as ROBP. This stronger security does introduce the requirement that our simulators must also operate within the constraints of space-bounded ROBPs.

Still, one might wonder if for malicious verifiers that are modeled as Turing machines, it is possible to achieve simulation via a Turing machine. Both our work and the work of Cormode *et al.* do make use of the power of a non-uniform computational model and we leave the above as an interesting open question.

Unless explicitly mentioned, any construction of a "streaming algorithm with space at most s" refers to a construction in the ROTM model, which by the lemma above can be executed by a ROBP with width at most $2^{s+O(\log(s))}$. Similarly, by Lemma 3.3, any upper bound on ROBP's with width at most $2^{s+O(\log(s))}$, also applies on a ROTM with space s. To simplify notation, when we refer to an ROBP with space s, we mean an ROBP with width 2^s.

3.2 Streaming Interactive Proofs

A *streaming interactive protocol* is a protocol between a computationally unbounded prover $\mathcal{P}$, and a bounded space, probabilistic verifier $\mathcal{V}$. The input is read once simultaneously by both parties in a streaming fashion, and the prover and the verifier interact by exchanging messages. At the end of the interaction the verifier produces an output.

Definition 3.4. *A* streaming interactive proof *with soundness error $\epsilon = \epsilon(|x|)$ for a language $\mathcal{L}$ is a streaming interactive protocol where the prover and verifier get as input $x \in \{0,1\}^*$ and satisfies the following conditions:*

- ***Perfect Completeness**: if $x \in \mathcal{L}$, then $\mathcal{V}$, following the interaction with $\mathcal{P}$, must accept with probability* 1*.*
- *ϵ-**Soundness**: if $x \notin \mathcal{L}$, then after interacting with any prover $\mathcal{P}^*$, the verifier $\mathcal{V}$ must accept with probability at most $\epsilon(|x|)$.*

Remark 3.5. *In the classical interactive proof setting, soundness error is often bounded by a fixed constant, and this error can be reduced by repeating the protocol a polynomial number of times and deciding based on the majority outcome. However, in this work, we do not consider repetition and instead focus on analyzing the soundness error for a single execution of the protocol. Sequential repetition is not feasible in the streaming model, as the input is streamed only once, and we do not consider parallel repetition due to its impact on space complexity and its potential to compromise zero-knowledge.*

3.3 Zero-Knowledge in Streaming Interactive Proofs

We formally define zero-knowledge in the streaming model. In order to do so, we first define the view of a streaming algorithm. The view of a ROTM is every bit accessed on the read tapes of the machine, i.e. the input tape, the randomness tape, and the message-receive tape in the order in which it was accessed. The view of a ROBP is the content on the read tape of the machine (which includes the input, randomness and the received messages in the order they are read). Note that the outgoing messages and output of the machines are a deterministic function of the view as defined above, and thus need not be included in the view.

We define streaming zero-knowledge as follows.

Definition 3.6. *Let n denote the length of the input x, and let $\langle \mathcal{P}, \mathcal{V} \rangle$ be a streaming interactive proof for a language $\mathcal{L}$ where $\mathcal{V}$ has space $s_{\mathcal{V}}$. We say that $\langle \mathcal{P}, \mathcal{V} \rangle$ is a* zero-knowledge with error $\delta = \delta(n)$ against adversaries with space $s = s(n)$ *if for all streaming verifiers $\mathcal{V}^*$ with space at most s there exists an ROBP streaming simulator $\mathcal{S}$ with space at most $s_{\mathcal{S}} = s + O(s_{\mathcal{V}})$, such that for all streaming distinguishers $\mathcal{D}$ with space at most $s_{\mathcal{D}} = s_{\mathcal{S}} + O(s_{\mathcal{V}})$ and all $x \in \mathcal{L}$,*

$$\Big| \Pr[\mathcal{D}(\mathsf{view}(\langle \mathcal{P}, \mathcal{V}^* \rangle)(x)) = 1] - \Pr[\mathcal{D}(\mathcal{S}(x)) = 1] \Big| \leq \delta,$$

where $\mathsf{view}(\langle \mathcal{P}, \mathcal{V}^* \rangle)(x)$ is the verifier's view when interacting with the prover on input x, and $\mathcal{S}(x)$ is the output of the simulator.

The rationale for setting the space constraints for the participating agents is as follows. We ensure that the simulator has enough space to execute both the malicious and honest verifiers, and we also allocate sufficient space for the distinguisher to run the simulator.

4 Statistically Hiding Streaming Commitment

In this section we construct a commitment scheme that will be used later in our streaming zero-knowledge protocols. The commitment is statistically hiding and is binding with respect to any small space algorithm (modeled as a read-once branching program (ROBP), see Section 3.1). Our construction utilizes a celebrated result by Raz [Raz18], that solving a system of linear equations is hard for algorithms with limited space.

We first define statistically hiding streaming commitments:

Definition 4.1. *Let n denote the security parameter. Let ℓ be the length of the string to which we wish to commit. An* $\big(s_{\mathcal{S}}(n,\ell), s_{\mathcal{R}}(n,\ell), s_b(n), \delta_h(n), \delta_b(n)\big)$ interactive, computationally binding, statistically hiding, streaming string commitment *is a protocol executed in two stages called* commit *and* reveal*, between a probabilistic ROTM sender with space $s_{\mathcal{S}}(n,\ell)$ and a probabilistic ROTM receiver with space $s_{\mathcal{R}}(n,\ell)$. The commit stage is a streaming interactive protocol. The receiver, given input n, ℓ, streams a long string α to the sender, while saving a state $h \in \{0,1\}^{s_{\mathcal{R}}}$. In response the sender, given full access to an input string $m \in \{0,1\}^{\ell}$, random coins s and streaming access to α, sends a commitment com_m to the receiver. In the reveal stage the sender sends m and the random coins s with which it produced com_m, and the receiver verifies that com_m is a valid commitment to m by running a deterministic algorithm $\mathsf{Verify}(m, s, h, com_m)$.*

The commitment satisfies the following three conditions:

1. ***Correctness**. If the two parties act honestly,* $\Pr[\mathsf{Verify}(m, s, h, com_m) = 1] = 1$, *where the probability is taken with respect to the coins of the receiver and the sender.*

2. ***Statistical hiding**. For any two messages* $m, m' \in \{0,1\}^\ell$ *and* any α *streamed from the receiver to the sender, the statistical distance between the distributions of* com_m *and* $com_{m'}$ *is at most* δ_h.

3. ***Computational binding against adversarial senders with space*** $s_b(n)$. *For any ROBP* $\mathcal{A}$ *with space bounded by* $s_b(n)$*, the probability that given streaming access to the string* α*, the machine* $\mathcal{A}$ *outputs* m, m', s, s', com*, such that* $m \neq m'$ *and* $\mathsf{Verify}(m, s, h, com) = \mathsf{Verify}(m', s', h, com) = 1$ *is at most* $\delta_b(n)$ *with respect to the randomness of* $\mathcal{A}$ *and the receiver's randomness for generating* α *and computing* h.

The main result shown in this section is a construction of a commitment as above.

Theorem 4.2. *Let* $\epsilon > 0$. *For all* $\ell < \frac{n}{21}$, *there exists a* $\big(s_{\mathcal{S}}(n,\ell), s_{\mathcal{R}}(n,\ell), s_b(n), \delta_h(n), \delta_b(n)\big)$ *interactive, computationally binding, statistically hiding, streaming string commitment, where* $s_{\mathcal{S}}(n,\ell) = O(\ell \cdot n)$, $s_{\mathcal{R}}(n,\ell) = O(n^{1+\epsilon} + \ell \cdot n)$, $s_b = \frac{n^2}{21}$, $\delta_h(n) = 2^{-\Omega(n)}$, $\delta_b(n) = 2^{-\Omega(n^\epsilon)}$ *and whose communication complexity is* $O(n^2)$.

The rest of this section is devoted to the proof of Theorem 4.2.

4.1 Preliminaries and Linear Algebra Limitations for ROBP

In this subsection we present the definitions and lemmas that we will use in proving Theorem 4.2. Note that we restrict our definitions and proofs to the binary field, and so the inner products are all taken modulo 2.

We first state a simplified, restricted version of the Leftover Hash Lemma [HILL99], that will suffice for us.

Lemma 4.3. *Let* $\mathcal{X} \subseteq \{0,1\}^n$. *Let* S, R *be random variables, where* S *is sampled uniformly from* $\mathcal{X}$ *and* R *is sampled uniformly from* $\{0,1\}^n$. *Then, if* $|\mathcal{X}| \geq \frac{1}{2\epsilon^2}$,

$$\mathrm{SD}\big((R, \langle R, S\rangle), (R, U)\big) \leq \epsilon,$$

where SD *denotes the statistical distance operator, and* U *is a uniform random bit.*

In the proofs ahead we make use of the following simple fact:

Fact 4.4. *For any* $y \in \{0,1\}^n \setminus \{0^n\}$, $\Pr_{x \in \{0,1\}^n}[\langle x, y\rangle = 0] = \frac{1}{2}$.

The computational binding property of our construction relies on Raz's result for hardness of learning parities. We state Raz's theorem.

Theorem 4.5 ([Raz18, Theorem 1.1])**.** *For any* $c < \frac{1}{20}$, *there exists an* $\alpha > 0$ *such that the following holds: Let* x *be uniformly distributed over* $\{0,1\}^n$. *Let* $m \leq 2^{\alpha n}$. *Let* $\mathcal{A}$ *be a ROBP that is given as input a stream of samples,* $(a_1, b_1), \ldots, (a_m, b_m)$, *where each* a_i *is uniformly distributed over* $\{0,1\}^n$ *and for every* $i, b_i = a_i \cdot x$. *Assume that* $\mathcal{A}$ *has space at most* cn^2 *and outputs a string* $\tilde{x} \in \{0,1\}^n$. *Then,* $\Pr[\tilde{x} = x] \leq O(2^{-\alpha n})$.

We recast the problem described in Theorem 4.5 as the following game:

Definition 4.6 (Parity Learning game (PL))**.** *Let x be uniformly distributed over $\{0,1\}^n$. A learner $\mathcal{A}$ receives a stream of m samples $(a_1,b_1),\dots,(a_m,b_m)$ where $a_i \in \{0,1\}^n$ is chosen at random and $b_i = \langle a_i, x\rangle$. After $\mathcal{A}$ finished streaming the input, it outputs a string $\tilde{x} \in \{0,1\}^n$. We say that $\mathcal{A}$ won the game if $\tilde{x} = x$.*

Note that by Theorem 4.5 any algorithm that uses sub-quadratic space and streams a sub-exponential number of samples, wins the PL game with exponentially small probability.

A seemingly easier task, yet only deceivingly so, is to distinguish between real linear equations and noise. A solution to this problem can be found by winning the following game:

Definition 4.7 (Decisional Parity Learning game (DPL))**.** *Let x be uniformly distributed over $\{0,1\}^n$ and let $\beta \in \{0,1\}$. A learner $\mathcal{A}$ receives a stream of m samples $c_1,\dots,c_m$, $c_i \in \{0,1\}^{n+1}$. If $\beta = 0$, then $c_i = (a_i, b_i)$, where a_i is drawn from the uniform distribution over n bits, and $b_i = \langle a_i, x\rangle$. If $\beta = 1$, then c_i is drawn from the uniform distribution over $n+1$ bits. After $\mathcal{A}$ finished streaming the input, it outputs a bit $\tilde{\beta} \in \{0,1\}$. We say that $\mathcal{A}$ won the game if $\tilde{\beta} = \beta$.*

Similarly to the DPL game, the Kernel Search (KS) game also tests a learner's ability to infer hidden structures from streamed data.

Definition 4.8 (Kernel Search game (KS))**.** *A learner receives a stream of m samples $a_1, a_2, \dots, a_m$, where a_i is uniformly distributed over $\{0,1\}^n$. Let A be a matrix such that a_i is the i-th row of A. After $\mathcal{A}$ finished streaming the input, it outputs a vector $k \in \{0,1\}^m$. We say that $\mathcal{A}$ won the game if k is a non-trivial vector in the kernel of A^T, i.e. $k \neq 0$ and $k^T A = 0$.*

With the following chain of reductions stated in Lemmas 4.9 and 4.10, we prove that winning the KS game is as hard as winning the DPL game which in turn is as hard as winning the PL game. Ultimately we show that breaking the binding of our commitment is essentially equivalent to winning the KS game, and thus by Raz's theorem, an appropriately memory-bounded algorithm cannot do so.

The following lemma presents the initial reduction from the KS game to the DPL game.

Lemma 4.9. *Suppose there exists a space s ROBP that wins the KS game with probability ϵ after receiving m samples. Then, there exists a ROBP with space $s + m + \log \ell$ that wins the DPL game with probability $1 - \delta$ after streaming $\ell \cdot m$ samples, where $\ell = O\left(\frac{\log(1/\delta)}{\epsilon^2}\right)$.*

Proof. We shall first construct $\tilde{\mathcal{A}}$ with space $s + m$ that wins the DPL game with advantage $\epsilon/2$ after receiving m samples. Upon streaming samples $c_i = (a_i, b_i) \in \{0,1\}^n \times \{0,1\}$, $\tilde{\mathcal{A}}$ feeds a_i to $\mathcal{A}$ and stores the bits b_i. Let $b = (b_1, b_2, \dots, b_m)$, and let $k \in \{0,1\}^m$ be the output of $\mathcal{A}$. If $k \neq 0^m$, $\tilde{\mathcal{A}}$ outputs $\tilde{\beta} = \langle b, k\rangle$, otherwise it outputs a random bit.

Note that $\tilde{\mathcal{A}}$ uses $\mathcal{A}$ and stores the vector b, and thus requires $s + m$ space.

Consider first the case where $\beta = 1$, i.e. the c_i's come from the uniform distribution over $\{0,1\}^{n+1}$. In this case the vector b is uniform over $\{0,1\}^m$, and thus for any non-zero k that $\mathcal{A}$ outputs (whether it is in the kernel of A^T or not), according to Fact 4.4, $\Pr[\langle b, k\rangle = 1] = \frac{1}{2}$. Also in the case where $\mathcal{A}$ returns 0^m, then $\tilde{\mathcal{A}}$ returns $\tilde{\beta} = 1$ with probability $\frac{1}{2}$. Thus, in this case $\tilde{\mathcal{A}}$ wins with probability $\frac{1}{2}$.

Now consider the case where $\beta = 0$, i.e. $c_i = (a_i, b_i)$ where $b_i = \langle a_i, x\rangle$ for some uniformly sampled $x \in \{0,1\}^n$. Thus, we can write $b = Ax$. With probability ϵ, $\mathcal{A}$ produces a vector k in the kernel of A^T. Therefore, $\tilde{\beta} = \langle b, k\rangle = \langle Ax, k\rangle = k^T Ax = 0 = \beta$, and $\tilde{\mathcal{A}}$ wins. If $\mathcal{A}$ failed to produce the vector in the kernel of A^T, then by the fact that $\mathcal{A}$ is oblivious to x (where x is uniformly random), Fact 4.4 gives us $\Pr[\tilde{\beta} = 0] = \Pr[\langle b, x\rangle = 0] = \Pr[k^T Ax = 0] = \Pr[\langle k^T A, x\rangle = 0] = \frac{1}{2}$. From the law of total probability we get that in the case where $\beta = 0$, $\tilde{\mathcal{A}}$ wins with probability $\frac{1}{2} + \epsilon/2$. Therefore, in total we get an advantage of $\epsilon/2$.

Now we shall construct $\mathcal{A}'$ that wins the DPL game with probability $1 - \delta$. All it has to do is to run $\tilde{\mathcal{A}}$ as described earlier $\ell = O\left(\frac{\log(1/\delta)}{\epsilon^2}\right)$ times, keep score, and answer according to whether the fraction of times $\tilde{\mathcal{A}}$ won out of the total games is closer to $\frac{1}{2}$ ($\beta = 1$) or $\frac{1}{2} + \epsilon/2$ ($\beta = 0$).

The correctness of the construction follows directly from the Chernoff bound, and the fact that the runs of $\tilde{\mathcal{A}}$ are independent of one another. Note that in order to operate, $\mathcal{A}'$ must maintain a counter which costs $O(\log \ell)$ space, and thus in total requires $s + m + O(\log \ell)$ space. □

The following lemma presents the second reduction, from the DPL game to the PL game.

Lemma 4.10. *Suppose there exists a ROBP with space s that wins the DPL game with probability $1 - \delta$ after receiving m samples. Then, there exists a ROBP with space $s + n$ that wins the PL game with probability $1 - n \cdot \delta$ after receiving $n \cdot m$ samples.*

Proof. Let $\mathcal{A}$ be a ROBP with space s that wins the DPL game with probability $1-\delta$. We construct the following bounded-space ROBP $\mathcal{A}'$ that wins the PL game with probability $(1 - \delta)^n$. $\mathcal{A}'$ will run as follows: For all $i \in [n]$, run $\mathcal{A}$ in the following manner. For every sample that $\mathcal{A}$ requires, get a sample (a, b) and feed it with $(a + r \cdot e_i, b)$, where r is drawn randomly from $\{0, 1\}$ and e_i is the i-th standard basis over vector of length n. Set x_i to be the output of $\mathcal{A}$.

First, we note that each invocation of $\mathcal{A}$ requires m samples, and is invoked n times, thus in total $\mathcal{A}'$ requires nm samples. In addition, we note that the only extra space that $\mathcal{A}'$ must have over $\mathcal{A}$ is for storing x, and thus requires space $s + n$.

To show that the construction is correct, we must show that if $x_i = 0$ then $b = \langle a + re_i, x\rangle$ and if $x_i = 1$ then b is a random bit. Indeed, if $x_i = 0$, then $\langle a + re_i, x\rangle = \langle a, x\rangle + rx_i = \langle a, x\rangle = b$. If $x_i = 1$, then $\langle a + re_i, x\rangle = \langle a, x\rangle + rx_i = b + r$, which means that b is random with respect to $a + re_i$.

Now, $\mathcal{A}'$ wins the PL game if all the invocations of $\mathcal{A}$ result in a win in the DPL game. Since the invocations are independent, we get that $\mathcal{A}'$ wins with probability $(1 - \delta)^n$. □

Combining Lemmas 4.9 and 4.10 we get the following corollary.

Corollary 4.11. *Let $\mathcal{A}$ be a ROBP with space s that wins the KS game with probability ϵ after streaming m samples. There exists a ROBP $\mathcal{A}'$ with space $s + m + \log \ell + n$ that wins the PL game with probability $1 - \delta n$ after streaming $n \cdot m \cdot \ell$ samples, where $\ell = O\left(\frac{\log(1/\delta)}{\epsilon^2}\right)$.*

By combining Corollary 4.11 with Theorem 4.5, we obtain the following corollary.

Corollary 4.12. *There exists $\beta > 0$ such that any ROBP with space at most $\frac{n^2}{21}$ wins the KS game with probability at most $2^{-\beta n}$, when streaming $O(n)$ samples.*

Proof. Assume the contrary and let $\mathcal{A}$ be such an adversary that wins the KS with probability $2^{-\beta n}$ after streaming $k_1 n$ samples for some constant k_1. If so, according to Corollary 4.11 there exists a constant k_2 and a ROBP $\mathcal{A}'$ that wins the PL game with probability at least $\frac{1}{2}$ after streaming $k_1 k_2 n^2 \log(2n) \cdot 2^{2\beta n} = 2^{2\beta n + o(n)}$ samples and using space $\frac{n^2}{21} + (k_1 + 1)n + O(\log n)$ ($< cn^2$, where $c < 1/20$, for large enough n). Note that the space of $\mathcal{A}'$ does not depend on β.

Recall that according to Theorem 4.5, for this $c < 1/20$ there exists some $\alpha > 0$ such that no adversary with space at most cn^2 wins the PL game with non-negligible probability after streaming at most $2^{\alpha n}$ samples. Thus, for $\beta = \alpha/3$, our assumption contradicts Theorem 4.5. □

4.2 Statistically Hiding Streaming String Commitment

Next, we turn to our construction of the commitment scheme.

Construction 4.13. *Let $\epsilon > 0$. Set $\lambda = \Theta(n^\epsilon)$, and let $T \in \{0,1\}^{n\times\lambda}$ be λ column vectors of length n drawn at random by the receiver in the beginning of the protocol. Let $A \in \{0,1\}^{4n\times n}$ be a random matrix streamed by the receiver to the sender, row by row, and let $m \in \{0,1\}^\ell$ be the message to be committed. While streaming the matrix A, the receiver computes AT and sets $h = (AT, T) \in \{0,1\}^{4n\times\lambda} \times \{0,1\}^{n\times\lambda}$.*

The Commit algorithm performed by the sender is as follows: Sample random $(S, r) \in \{0,1\}^{\ell\times 4n} \times \{0,1\}^{4n}$. While streaming A, compute the value $\alpha = SA \in \{0,1\}^{\ell\times n}$. Set $com_m = (r, \alpha, \beta)$, where $\beta = Sr + m$.

The Verify *algorithm performed by the receiver is as follows: Given the message m, the coins of the sender (S, r) the commitment (r, α, β), and the internal state of the receiver (T, AT), accept if and only if $S \cdot (AT) = \alpha T$, and $Sr + m = \beta$.*

Note that the communication complexity of the commitment scheme is $O(n^2)$, as the major part of the communication is the streaming of the matrix $A \in \{0,1\}^{4n\times n}$.

We claim that Construction 4.13 satisfies Definition 4.1 and so proves Theorem 4.2. We first show that the commitment satisfies the space constraints of the honest parties and the correctness requirement.

Lemma 4.14. *For all ℓ, an honest execution of Construction 4.13 can be done by a ROTM sender with space $s_{\mathcal{S}}(n, \ell) = O(\ell \cdot n)$, a ROTM receiver with space $s_{\mathcal{R}}(n, \ell) = O(n^{1+\epsilon} + \ell \cdot n)$, and at the end the receiver will always accept.*

Proof. The only non-trivialities in the lemma are the facts that the sender computes $\alpha = SA$ in space $O(\ell \cdot n)$, and the receiver computes AT in space $O(n^{1+\epsilon} + \ell \cdot n)$ as the matrix A is streamed, row by row. For the former, as the sender receives the k-th row of A, where $k \in [4n]$, the sender computes a matrix $\Psi^k \in \{0,1\}^{\ell\times n}$ where $\Psi^k_{i,j} = S_{i,k} \cdot A_{k,j}$. Note that $SA = \sum_{k\in[4n]} \Psi^k$, and so the sender needs only to accumulate the Ψ^k's, which can be done in the allotted space. As to the latter, the receiver can compute AT row by row, as it streams the rows of A. □

In order to show that the construction satisfies the hiding and binding properties we first show that it does so for messages of length $\ell = 1$. We then use this result to prove that the construction also works for messages of length $\ell > 1$.

The following two lemmas state that the construction satisfies the computational hiding and statistical hiding properties, for commitments of messages of length $\ell = 1$, i.e. a bit commitment. Note that since we are considering the case where $\ell = 1$ in the following proofs, the matrices S, α from the construction are actually vectors. For notational convenience we denote the matrix $S \in \{0,1\}^{1\times 4n}$ as $s \in \{0,1\}^{4n}$.

Lemma 4.15. *For $\ell = 1$, Construction 4.13 is computationally binding against adversarial senders with space at most $s_b(n) = \frac{n^2}{21}$ per Definition 4.1 with $\delta_b(n) = 2^{-\Omega(n^\epsilon)}$.*

Proof. Assume there exists a bounded ROBP adversary $\mathcal{S}$ using $\frac{n^2}{21}$ memory bits that breaks the binding. This means that the adversary is able to find $s \neq s'$ such that $s^T AT = \alpha T = s'^T AT$. Rewritten, this means that $\mathcal{S}$ can find a non-zero vector $\tilde{s} = s - s'$ such that $\tilde{s}^T AT = 0$. We shall separate into two cases. If $\tilde{s}^T A = 0$, then it means that $\mathcal{A}$ finds a non-trivial vector in the kernel of A and by such can win the KS game. From Corollary 4.12, we conclude that this cannot be done with probability greater than $2^{-\beta n}$ for the constant $\beta > 0$ that we get from the corollary. If $\tilde{s}^T A \neq 0$, then $\langle \tilde{s}^T A, \tau_i \rangle = 0$ for all $i \in [\lambda]$, where τ_i is the i-th column of T. Since τ_i is a random vector, according to Fact 4.4, $\Pr[\langle \tilde{s}^T A, \tau_i \rangle = 0] = \frac{1}{2}$. Thus, $\Pr[\tilde{s}^T AT = 0] = 2^{-\lambda} = 2^{-\Theta(n^\epsilon)}$. Applying the union bound on the two cases, the lemma follows. □

Lemma 4.16. *For $\ell = 1$, Construction 4.13 is statistically hiding (per Definition 4.1) with $\delta_h(n) = 2^{-n}$.*

Proof. Set matrix $A \in \{0,1\}^{4n \times n}$ and set α in the image of A^T, i.e. $\alpha = s^T A$ for some s. Let $X \subseteq \{0,1\}^{4n}$ be the set of all possible pre-images for α. Since the dimension of the domain of A^T is $4n$ and the dimension of the range is n, the kernel dimension of A^T is at least $3n$. Therefore, $|X| \geq 2^{3n}$. Thus, using Lemma 4.3,

$$(r, \alpha, \langle s, r \rangle) \approx_{2^{-(n+1)}} (r, \alpha, b)$$

Where r is drawn uniformly at random from $\{0,1\}^{4n}$ and b is drawn uniformly at random from $\{0,1\}$.

Thus, $com_0 = (r, \alpha, \langle s, r \rangle + 0) \approx_{2^{-n}} (r, \alpha, \langle s, r \rangle + 1) = com_1$, as required. □

The next two lemmas state that the construction satisfies the computational hiding property, and the statistical hiding property, for commitments of messages of length $\ell = o(n)$. Note that according to the structure of the commitment, we have that $\alpha_i = s_i A, \beta_i = \langle s_i, r \rangle + m_i$, where s_i is the i-th row of S, α_i is the i-th row of α, and β_i, m_i are the i-th entries of β, m, respectively, where $i \in [\ell]$.

Lemma 4.17. *For any $\ell < \frac{n}{21}$, Construction 4.13 is computationally binding against adversarial senders with space at most $s_b(n) = \frac{n^2}{21}$ per Definition 4.1 with $\delta_b(n) = 2^{-\Omega(n^\epsilon)}$.*

Proof. Assume that there exists some adversarial ROBP sender $\mathcal{S}^*$ with space at most $\frac{n^2}{21}$ that breaks the binding of the commitment. We show that we can construct a ROBP sender $\mathcal{S}_1^*$ with similar space, that breaks the binding of the bit commitment, in contradiction to Lemma 4.15.

Given a stream A from the receiver, $\mathcal{S}_1^*$ will forward the stream to $\mathcal{S}^*$, which in turn will output m, m', S, S', com, where $m \neq m'$ and $com = (r, \alpha, \beta)$ such that with probability at least 2^{-n^ϵ} $S(AT) = S'(AT)$ and $Sr + m = S'r + m' = \beta$. In particular, $s_i^T AT = {s'}_i^T AT$ and $\langle s_i, r \rangle + m_i = \langle s'_i, r \rangle + m_i$ for all $i \in [\ell]$. Since $m \neq m'$, they must differ in at least a single entry i^*. Without loss of generality, assume $m_{i^*} = 0 \neq m'_{i^*} = 1$. In turn we have that $\langle s_{i^*}, r \rangle = \langle s'_{i^*}, r \rangle + 1$, and so it must be that $s_{i^*} \neq s'_{i^*}$. Thus, $\mathcal{S}_1^*$ outputs $0, 1, s_{i^*}, s'_{i^*}, com$ where $com = (r, \alpha_{i^*}, \beta_{i^*})$, and by doing so, breaks the binding of the commitment of a single bit. □

Lemma 4.18. *For all $\ell < \frac{n}{21}$, Construction 4.13 is statistically hiding per Definition 4.1 with $\delta_h(n) = 2^{-\Omega(n)}$.*

Proof. Let $\ell < \frac{n}{21}$. Fix some A streamed by the receiver and let $\alpha = SA$ such that $\alpha_i = s_i^T A$ for all $i \in [\ell]$. Consider the following hybrid distributions for $j \in \{0, 1, \ldots, \ell\}$:

$$H_j = \left(r, \{\alpha_i\}_{i=1}^{\ell}, \{\langle s_i, r\rangle\}_{i=1}^{j}, \{b_i\}_{i=j+1}^{\ell}\right).$$

Note that $H_0 = (r, \alpha, b)$ where $b \in \{0,1\}^{\ell}$ is a random vector and $H_\ell = (r, \alpha, Sr)$. For all $j \in \{0, 1, \ldots, \ell - 1\}$, according to the data-processing inequality for statistical distance we have that

$$\mathrm{SD}(H_j, H_{j+1}) \leq \mathrm{SD}\big((r, \alpha_j, b_j), (r, \alpha_j, \langle s_j, r\rangle)\big),$$

which according to Lemma 4.16, is further bounded by $2^{-(n+1)}$. Thus, from the triangle inequality we get that $\mathrm{SD}(H_0, H_\ell) \leq 2^{-(n-\ell+1)}$.

□

Theorem 4.2 now follows immediately from Lemmas 4.14, 4.17 and 4.18.

5 Linearly Homomorphic Statistically Binding Streaming Commitment

In this section we construct a *linearly homomorphic statistically binding streaming commitment*. Such a commitment enables a sender to commit to a bit vector, and later de-commit to arbitrary linear functions of the vector, without revealing any information regarding the vector, other than what can be inferred by the revealed linear combinations (i.e., correlations that are outside the span of the revealed linear functions remain hidden). From this general construction we will later deduce the *polynomial commitment scheme*, which is a statistically binding commitment scheme where a sender can commit to a polynomial and later de-commit to evaluations of the polynomial at a given set of points, while the evaluations of the polynomial at all other points remain hidden from the receiver.

We proceed to define the linearly homomorphic commitment scheme. The linear functions of the committed vector v to which the sender de-commits are represented as a matrix L, and so the de-commitment is to the value $v^T L$. Observe that in the simple case where the de-commitment is to the entire input vector v, the matrix L is the identity matrix of size $|v| \times |v|$. This explicit representation of the matrix can be rather wasteful in terms of space, a precious resource in the streaming model, and so we would like for the matrix L to have a succinct representation that could be sent between the participating parties with minimal communication complexity, and from which the parties can infer the values of the matrix without having to store the entire matrix explicitly.

Thus, we define the following:

Definition 5.1. *An* $s(n)$-strongly explicit[8] matrix *is a family of matrices* $\{M_n\}_{n\in\mathbb{N}}$*, where there exists a space-*$O(s(n))$ *algorithm, that given a representation* $\tilde{M}_n$ *of the matrix, of length* $|\tilde{M}_n| = O(s(n))$*, and indices* i, j*, outputs* $M_n[i, j]$*.*

We now proceed to define our commitment scheme.

[8] The term "strongly explicit" is borrowed from similar terminology in the context of succinctly representing graphs.

Definition 5.2. *Let n denote the security parameter and let ℓ be the length of the bit vector to which we wish to commit. An $(s_{\mathcal{S}}(n,\ell), s_{\mathcal{R}}(n), s_h(n), \delta_h(n), \delta_b(n))$* linearly homomorphic, statistically binding, streaming commitment *is a protocol executed in two stages called the commit stage and the reveal stage, between a probabilistic ROTM sender with space $s_{\mathcal{S}}(n,\ell)$ and a probabilistic ROTM receiver with space $s_{\mathcal{R}}(n)$. In the commit stage, the sender sends as a stream to the receiver the commitment com_v to its input bit vector $v \in \{0,1\}^\ell$. As the receiver reads com_v (which may be long), it computes a function $g(com_v)$, where $g : \{0,1\}^* \to \{0,1\}^{s_{\mathcal{R}}(n)}$ is a randomized function computable in space $s_{\mathcal{R}}(n)$.*

In the reveal stage the receiver sends to the sender a succinct representation $\tilde{L}$ of a binary $s_{\mathcal{R}}(n)$-strongly explicit matrix $L \in \{0,1\}^{\ell \times j}$, such that $0 \le j \le \ell$. In response, the sender sends a key κ_L, which may depend on L and its internal state, and the receiver runs a deterministic algorithm $\mathsf{Verify}(\tilde{L}, \kappa_L, g(com_v))$*, that either outputs a bit string of length j, or rejects.*

The commitment satisfies the following three conditions:

1. ***Correctness:*** *If the two parties act honestly, for every $v \in \{0,1\}^\ell$ and $L \in \{0,1\}^{\ell \times j}$ it holds that $\Pr[\mathsf{Verify}(\tilde{L}, \kappa_L, g(com_v)) = v^T L] = 1$, where the probability is taken over the coins of the receiver and the sender.*

2. ***Computational hiding against adversarial ROBP receivers with space $s_h(n)$:*** *For any function $g^* : \{0,1\}^* \to \{0,1\}^{s_h(n)}$ computable in space $s_h(n)$, any $0 \le j \le \ell$, any matrix $L \in \{0,1\}^{\ell \times j}$, and any v_0, v_1 such that $v_0^T L = v_1^T L$, the statistical distance between the distributions $\big(g^*(com_{v_0}), \kappa_L\big)$ and $\big(g^*(com_{v_1}), \kappa_L\big)$ is at most $\delta_h(n)$.*

3. ***Statistical Binding:*** *For any commitment com, there exists a vector v, such that for all strongly explicit matrices L and for all κ, $\Pr[\mathsf{Verify}(\tilde{L}, \kappa, g(com)) \in \{rej, v^T L\}] > 1 - \delta_b(n)$, where the probability is taken over the receiver's private randomness for computing $g(com)$.*

Remark 5.3 (More on the Hiding Property)**.** *Note that in case $j = 0$, for any $v_0, v_1 \in \{0,1\}^\ell$, it holds that $v_0^T L = v_1^T L$ (vacuously). Indeed, in such a case the hiding property guarantees that the committed message is fully hidden after the commit stage.*

Remark 5.4 (More on the Binding Property)**.** *The binding condition states that there is a unique value to which the sender can de-commit with non-negligible probability for a given commitment.*

The commitment scheme that we construct will actually satisfy a stronger notion of hiding than that in Definition 5.2, which will actually be more convenient to prove. Loosely speaking, the commit stage of our construction is divided into two phases, and the commitment is in fact hiding against *unbounded* adversaries, as long as the adversary can retain only $s_h(n)$ bits in its memory at the end of the first phase, going into the second. This is captured by the following definition.

Definition 5.5 (Strong Hiding)**.** *An $(s_{\mathcal{S}}(n,\ell), s_{\mathcal{R}}(n), s_h(n), \delta_h(n), \delta_b(n))$ linearly homomorphic, statistically binding, streaming commitment per Definition 5.2 satisfies* strong hiding *if the following holds:*

The commit stage happens in two phases. In the first phase, the sender sends to the receiver a long string A as a stream. As the receiver reads A, it computes a function $g(A)$, where $g : \{0,1\}^* \to \{0,1\}^{s_{\mathcal{R}}(n)}$ is a randomized function computable in space $s_{\mathcal{R}}(n)$. In the second phase the sender sends as a stream a commitment com_v to the vector v. We require that for any function $g^* : \{0,1\}^* \to \{0,1\}^{s_h(n)}$ (not necessarily computable in small space), any $0 \le j \le \ell$, any matrix

$L \in \{0,1\}^{\ell \times j}$, and any v_0, v_1 such that $v_0^T L = v_1^T L$, the statistical distance between the distributions $\big(g^*(A), com_{v_0}, \kappa_L\big)$ and $\big(g^*(A), com_{v_1}, \kappa_L\big)$ is at most $\delta_h(n)$.

The difference between the hiding property in Definition 5.2 and in Definition 5.5 is that the latter is hiding against *unbounded* adversaries with the requirement that the adversary retains only $s_h(n)$ bits from the computation of A when continuing to the second phase of the commitment. The former requires the receiver never to exceed a space of $s_h(n)$ bit during the entire protocol.

Proposition 5.6. *Strong hiding implies the hiding property of Definition 5.2.*

Proof. An adversary that breaks the "weak" hiding property of a commitment satisfying Definition 5.2 clearly does not retain more than $s_h(n)$ bits from the computation of A since it uses only $s_h(n)$ space, and so it is a valid adversary per Definition 5.5 which breaks the strong hiding property. □

Committing to Multiple Messages. In the following sections, our constructions incorporate multiple instances of the statistically hiding commitment within larger protocols. The following lemma demonstrates that these commitments maintain their hiding property, ensuring no information leaks between different instances or from the surrounding protocol to the commitments, even in this context.

Lemma 5.7. *Let $(X,Y),(X',Y')$ be joint distributions, each defined over a product space $\Omega_X \times \Omega_Y$, where Ω_Y is the space of all matrices $y \in \{0,1\}^{k \times \ell}$. Suppose the marginal distributions X, X' are identically distributed and define $com(y)$ to be the concatenation of statistically binding streaming commitments per Definition 5.2 of the k rows of y. Then, for all space-s_h ROBP distinguishers $\mathcal{D}$,*

$$\left| \Pr_{(x,y)\leftarrow(X,Y)}[\mathcal{D}(x, com(y))] = 1] - \Pr_{(x,y)\leftarrow(X',Y')}[\mathcal{D}(x, com(y))] = 1] \right| \le k \cdot \delta_h.$$

The proof, which is straightforward, is deferred to Appendix B.1.

Main Theorem. The main result that we show in this section is a construction of a linearly homomorphic, statistically binding, streaming commitment, which, moreover, satisfies the strong hiding property. This is captured by the following theorem.

Theorem 5.8. *Let n be a security parameter, let $\epsilon > 0$, let $\ell \le \frac{n}{8}$ be the length of the committed vector v,and let $j \le \min\left\{\ell, \frac{n}{24}\right\}$ be the length of the de-committed vector of linear evaluations of v. Then, there exists an $\big(s_{\mathcal{S}}(n,\ell), s_{\mathcal{R}}(n), s_h(n), \delta_h(n), \delta_b(n)\big)$ linearly homomorphic, statistically binding, streaming commitment, where $s_{\mathcal{S}}(n,\ell) = O(n \cdot (\ell + n^\epsilon))$, $s_{\mathcal{R}}(n) = O\big(n^{1+\epsilon}\big)$, $s_h(n) = \frac{n^2}{4}$, $\delta_h(n) = 2^{-\frac{n}{17}}$, $\delta_b(n) = 2^{-\Omega(n^\epsilon)}$ and whose communication complexity is $O(n^2 + n \cdot \ell)$. Moreover, the scheme satisfies strong hiding.*

In Section 5.1 we present the construction of the commitment scheme and by such prove Theorem 5.8. In Section 5.2, using our construction from Section 5.1 we construct a polynomial commitment scheme, in which a sender can commit to a polynomial and de-commit to evaluations of the polynomial without revealing any other information about the polynomial. The polynomial commitment will be integrated into our streaming zero-knowledge constructions in the following sections.

5.1 Linearly Homomorphic Statistically Binding Streaming Commitment

In this subsection we prove Theorem 5.8 by constructing an $\big(s_{\mathcal{S}}(n,\ell), s_{\mathcal{R}}(n), s_h(n), \delta_h(n), \delta_b(n)\big)$ strong linearly homomorphic, statistically binding, streaming commitment, for strings of length ℓ and de-commitments of length $j \leq \min\left\{\ell, \frac{n}{24}\right\}$, where $s_{\mathcal{S}}(n,\ell) = O(n \cdot \ell)$, $s_{\mathcal{R}}(n) = O(n^{1+\epsilon})$, $s_b(n) = \frac{n^2}{4}$, $\delta_h(n) = 2^{-\frac{n}{17}}$, and $\delta_b(n) = 2^{-\Omega(n^\epsilon)}$ with parameter $\epsilon > 0$.

Construction 5.9. *Let $\epsilon > 0$, let $\lambda = \Theta(n^\epsilon)$, and let $v \in \{0,1\}^\ell$ be the bit vector to be committed. Let $L \in \{0,1\}^{\ell \times j}$ be an $n^{1+\epsilon}$-strongly explicit matrix defining the linear transformation to be applied on v known only to the receiver, where $j \leq \min\left\{\ell, \frac{n}{24}\right\}$. The sender draws uniformly at random a matrix $K \in \{0,1\}^{n\times\ell}$, and streams to the receiver a random matrix $A \in \{0,1\}^{n\times n}$, row by row, while computing and storing the matrix $AK \in \{0,1\}^{n\times\ell}$. As the receiver receives the matrix A, it computes TA, for some randomly selected $T \in \{0,1\}^{\lambda\times n}$, which is* not *revealed to the sender. Subsequently, the sender sends (r, K, w) as a stream, where $w^T = r^T AK + v^T$, the vector $r \in \{0,1\}^n$ is drawn at random, and K is streamed* column by column. *As the receiver streams the matrix K, it computes $TAKL \in \{0,1\}^{\lambda\times j}$, and as it streams w it computes $w^T L \in \{0,1\}^j$.*

In order to de-commit, the receiver sends to the sender a succinct representation of L, and the sender sends in response the matrix $\kappa = AKL \in \{0,1\}^{n\times j}$ as a stream. As the receiver streams κ it computes $T\kappa$ and $r^T\kappa$. It verifies that $T\kappa = TAKL$, and if so it outputs $w^T L - r^T\kappa$, otherwise it rejects.

Note that the sender does not need to store the matrix A, and only needs enough space to store $K \in \{0,1\}^{n\times\ell}, AK \in \{0,1\}^{n\times\ell}, r \in \{0,1\}^n$, and $w \in \{0,1\}^\ell$, all of which take $O(n \cdot \ell)$ space, as required. The receiver does not need to store the matrices A and K, but only $T \in \{0,1\}^{\lambda\times n}$, $TA \in \{0,1\}^{\lambda\times n}$, $TAKL \in \{0,1\}^{\lambda\times j}$, $T\kappa \in \{0,1\}^{\lambda\times j}$, $r \in \{0,1\}^n$ and $w^T L \in \{0,1\}^j$, all of which require $O(n^{1+\epsilon})$ space. Also note that the matrix TA can be computed in an online fashion in small space from T while streaming A, as described in the proof of Lemma 4.14.

The matrix $TAKL$ can also be computed in an online fashion in small space while streaming K column by column, by the following method. Since the receiver has the matrix TA stored, for every column k_i of K that it streams it can compute TAk_i by standard matrix-vector multiplication. Thus, we can consider the streamed columns of the matrix K as a stream of rows of the matrix $(TAK)^T$. Therefore, the receiver can compute $L^T(TAK)^T$ (from which we can get $TAKL$ by transposing) as a cumulative sum of matrices of dimension $j \times \lambda$, once again, as described in the proof of Lemma 4.14.

The major part in the communication between the parties during the protocol is sending the matrix $A \in \{0,1\}^{n\times n}$ in the beginning of the commitment, and so the communication complexity of the protocol is $O(n^2)$.

Note that since the representation of L does not exceed $O\big(n^{1+\epsilon}\big)$ bits, sending and storing it does not add to the space complexity of either party.

We proceed to show that the construction satisfies the definition. We start with showing correctness.

Proposition 5.10. *Construction 5.9 is correct, i.e. when executed by an honest sender and receiver, the value revealed to the receiver is $v^T L$.*

Proof. Indeed, $T\kappa = T(AKL) = TAKL$ and so the receiver does not reject. Likewise, the output of the receiver as stated in the construction satisfies $w^T L - r^T\kappa = (r^T AK + v^T)L - r^T AKL = v^T L$, as required. □

5.1.1 Showing the Hiding Property

For convenience, we restate the computational strong hiding property of Definition 5.5 using a game-based definition. The following description of a game between an (unbounded) distinguisher $\mathcal{D}$ and a challenger captures the hiding property for this model.

1. $\mathcal{D}$ sends to the challenger two vectors $v_0, v_1 \in \{0,1\}^{\ell}$ and a matrix L, such that $v_0^T L = v_1^T L$.
2. The challenger sends a long string A in a streaming fashion to $\mathcal{D}$. The distinguisher $\mathcal{D}$ computes $g^*(A)$ for a function $g^* : \{0,1\}^* \to \{0,1\}^{s_h(n)}$.
3. The challenger randomly chooses $b \in \{0,1\}$, and sends to $\mathcal{D}$ a commitment com_{v_b}.
4. The challenger then sends a key κ as per Definition 5.2.
5. $\mathcal{D}$ outputs $b' \in \{0,1\}$.

We say that $\mathcal{D}$ won if $b = b'$. The strong hiding property of Definition 5.5, in terms of this game, is equivalent to the claim that any distinguisher has advantage at most δ_h in winning the game.

Recall that, following Definition 5.5, we do not impose computational limitations on $\mathcal{D}$ during the security game. In particular, in Step 2, $\mathcal{D}$ is allowed to compute *any* function $g : \{0,1\}^* \to \{0,1\}^{s_h(n)}$, including functions that require more space than $s_h(n)$ during their computation. The only restriction is that by the time $\mathcal{D}$ receives com_{v_b} in Step 3, it may only retain at most $s_h(n)$ bits from the computation of g.

The hiding (resp., strong hiding) property of Definition 5.2 (resp., Definition 5.5), implies that the commitment remains hiding even after the de-commitment, in the sense that the de-commitment reveals only the projections of the committed vector onto the requested space, and all other components of the vector remain hidden. In order to prove that the construction satisfies the hiding property, we first show that if the construction is hiding before the de-commitment (i.e., before the projection is revealed), then it is also hiding after the de-commitment. We formalize this idea in the following proposition:

Proposition 5.11. *Let $0 \le \eta < \ell$ be an integer. Suppose that Construction 5.9 is a linearly homomorphic, statistically binding, streaming commitment for vectors of length $\ell - \eta$, that is hiding against streaming adversaries with space $s + 2n \cdot \eta$ per Definition 5.2 where the receiver does not receive a key from the sender (i.e. $j = 0$ as in Remark 5.3)*[9]*. Then, Construction 5.9 is a linearly homomorphic, statistically binding, streaming commitment for vectors of length ℓ, with de-commitments of length η (i.e. $j = \eta$) that is hiding against streaming adversaries with space s.*

Proof. We may assume w.l.o.g that $\eta > 0$ since $\eta = 0$ corresponds to the case that the receiver does not get a key κ following the commitment, and so in this case the proposition is a tautology.

Let $0 < \eta < \ell$. Assume that there exists a distinguisher $\mathcal{D}$ with space s and a matrix $L \in \{0,1\}^{\ell \times \eta}$ such that $\mathcal{D}$ wins the commitment's hiding security game with $j = \eta$, with advantage greater than δ.

[9] For consistency with the hiding condition in Definition 5.5, as per Remark 5.3, we define that for $j = 0$, the distinguisher playing the game may choose in Step 1 any two vectors $v_0, v_1 \in \{0,1\}^{\ell}$ with no restrictions, and in Step 4, the challenger will only send the commitment com_{v_b} and *no* key κ.

Assume w.l.o.g that L is full rank (i.e., of rank η).[10] Thus, we may permute the rows of L such that $L = \begin{pmatrix} L_1 \\ L_2 \end{pmatrix}$ where $L_1 \in \{0,1\}^{(\ell-\eta)\times\eta}$, and $L_2 \in \{0,1\}^{\eta\times\eta}$ is a full rank square matrix. In particular, L_2^{-1} is well-defined.

Consider the following observation. For a vector $x \in \{0,1\}^\ell$, we can write $x^T = [\, x_1^T \,||\, x_2^T \,]$ where x_1 are the first $\ell - \eta$ entries of x, the $||$ sign denotes concatenation, and x_2 are the last η entries of x. For $x^T L = u^T$ we have that $x_1^T L_1 + x_2^T L_2 = u^T$, and so $x_2^T = (u^T - x_1^T L_1)L_2^{-1}$. Let $Q = \begin{pmatrix} L_2^{-1} \\ -L_1 L_2^{-1} \end{pmatrix}$, and so we can write $x^T = [\, x_1^T \,||\, (u^T \,||\, x_1^T)Q \,]$.

We construct a distinguisher $\hat{\mathcal{D}}$ with space $s + 2n \cdot \eta$ that wins the game described above for $j = 0$ with advantage greater than δ_h with the following changes: In Step 1 the vectors $\hat{v}_0, \hat{v}_1$ can be any vectors of length $\ell - \eta$, and we skip Step 4, i.e. the challenger does not send a key κ.

We proceed to describe the distingusher $\hat{\mathcal{D}}$. To pick the vectors $\hat{v}_0, \hat{v}_1 \in \{0,1\}^{\ell-\eta}$ in Step 1, $\hat{\mathcal{D}}$ invokes $\mathcal{D}$ to produce the vectors $v_0, v_1 \in \{0,1\}^\ell$ where $v_0^T L = v_1^T L = u^T \in \{0,1\}^\eta$. The distiguisher $\hat{\mathcal{D}}$ then sets $\hat{v}_b$ to be the first $\ell - \eta$ bits of v_b for $b \in \{0,1\}$.

In Step 2, the challenger sends to $\hat{\mathcal{D}}$ the matrix $A \in \{0,1\}^{n\times n}$ as a stream. $\hat{\mathcal{D}}$ draws a random matrix $\tilde{K} \in \{0,1\}^{n\times\eta}$ and while streaming the matrix A, it computes $A\tilde{K} \in \{0,1\}^{n\times\eta}$, and forwards the matrix A to $\mathcal{D}$ that computes $g^*(A)$. Denote $\hat{g}(A) = \big(g^*(A), \tilde{K}, A\tilde{K}\big)$. Note that $\tilde{K}, A\tilde{K} \in \{0,1\}^{n\times\eta}$, and $|g^*(A)| \le s$, and so $|\hat{g}(A)| \le s + 2n \cdot \eta$, as required.[11]

In Step 3 $\hat{\mathcal{D}}$ receives $com_{\hat{v}_b} = (r, K, r^T AK + \hat{v}_b^T)$ from the challenger, and computes $K' = (\tilde{K} \,||\, K)Q$ and $r^T A(K||K') + v_b^T$. We show how $\hat{\mathcal{D}}$ computes $r^T A(K||K') + v_b^T$. Recall that from the observation above, we can write $v_b^T = [\, \hat{v}_b^T \,||\, (u^T||\hat{v}_b^T)Q \,]$, and so we can rewrite $r^T A(K||K') + v_b^T$ as

$$[\,(r^T AK + \hat{v}_b^T) \,||\, (r^T AK' + (u^T||\hat{v}_b^T)Q)\,].$$

As $\hat{\mathcal{D}}$ gets $r^T AK + \hat{v}_b^T$ from the challenger, it is enough to show that it can compute $r^T AK' + (u^T||\hat{v}_b^T)Q$. This is not obvious since $\hat{\mathcal{D}}$ can compute K' only after it has received K from the challenger, and this happens after the streaming of A, and so it cannot directly compute AK'. Nevertheless, rewritten, we have that

$$\begin{aligned} r^T AK' + (u^T \,||\, \hat{v}_b^T)Q &= r^T A(\tilde{K} \,||\, K)Q + (u^T \,||\, \hat{v}_b^T)Q \\ &= (r^T A\tilde{K} + u^T \,||\, r^T AK + \hat{v}_b^T)Q, \end{aligned}$$

where u can be computed from v_0, v_1, L in Step 1, $r^T A\tilde{K}$ can be computed from the output of $\hat{g}(A)$ and r, and $r^T AK + \hat{v}_b^T$ is sent by the challenger.

The distinguisher $\hat{\mathcal{D}}$ forwards $com_{v_b} = (r, (K||K'), r^T A(K||K') + v_b^T)$ to $\mathcal{D}$. Note that this is indeed the input format which $\mathcal{D}$ expects to see from the challenger in Step 3 of the game. $\hat{\mathcal{D}}$ then simulates Step 4 for $\mathcal{D}$ by feeding it with the key $\kappa = A\tilde{K}$. Recall that $K' = (\tilde{K}||K)Q$, and so

$$K' = \tilde{K}L_2^{-1} - KL_1L_2^{-1} \quad\Rightarrow\quad \tilde{K} = KL_1 + K'L_2 \quad\Rightarrow\quad \tilde{K} = (K||K')L.$$

Thus, $\kappa = A(K||K')L$, and so κ is in the format that $\mathcal{D}$ expects to receive.

[10] We can assume this since if the rank of L were $r < \eta$, there would be $\eta - r$ columns in L that are linear combinations of the other r columns, and so could be inferred implicitly.

[11] Recall that we allow the computation of $g^*, \hat{g}$ not to be bounded in space, as long as the output size of the function is short.

Finally, $\hat{\mathcal{D}}$ will return b', the output of $\mathcal{D}$.

By construction, $\hat{\mathcal{D}}$ is correct on $\hat{v}_b$ if and only if $\mathcal{D}$ distinguishes v_b correctly. Thus, similarly to $\mathcal{D}$, the distinguisher $\hat{\mathcal{D}}$ wins the game with advantage greater than δ_h, in contradiction to the hiding property of the construction as stated in the proposition. □

5.1.2 Proof of Theorem 5.8

To prove Theorem 5.8, we show that Construction 5.9 is a commitment scheme per Definition 5.2 with the parameters in the theorem statement. We use the following lemma to prove that Construction 5.9 satisfies the strong hiding property of Definition 5.5.

Lemma 5.12. *Let $r \leftarrow \{0,1\}^n$, $K \leftarrow \{0,1\}^{n\times\ell}$ and $A \leftarrow \{0,1\}^{n\times n}$, where $\ell \leq \frac{n}{8}$ Let $f : \{0,1\}^{n\times n} \to \{0,1\}^s$ be an arbitrary function, where $s \leq \frac{n^2}{3}$. Then:*

$$\mathrm{SD}\big((r, K, r^T AK, f(A)), (\mathcal{U}, f(A))\big) \leq 2^{-\frac{n}{17}}$$

where $\mathcal{U}$ is the uniform distribution over binary vectors of length $n + n\ell + \ell$.

The proof of Lemma 5.12, which is inspired by the proof the Leftover Hash Lemma, is deferred to Section 5.3.

Using Lemma 5.12, we now prove the following lemma which shows that Construction 5.9 is a statistically binding, streaming vector commitment that satisfies *strong hiding*.

Lemma 5.13. *Let n be a security parameter, let $\epsilon > 0$, let $\ell \leq \frac{n}{8}$ be the length of the committed vector v, and let $j \leq \min\left\{\ell, \frac{n}{24}\right\}$ be the length of the de-committed vector of linear evaluations of v. Then, Construction 5.9 is a $\big(s_{\mathcal{S}}(n,\ell), s_{\mathcal{R}}(n), s_h(n), \delta_h(n), \delta_b(n)\big)$ statistically binding, streaming vector commitment that satisfies strong hiding, for strings of length ℓ, where $s_{\mathcal{S}}(n,\ell) = O(n \cdot \ell)$, $s_{\mathcal{R}}(n) = O(n^{1+\epsilon})$, $s_h(n) = \frac{n^2}{4}$, $\delta_h(n) = 2^{-\frac{n}{17}}$, and $\delta_b(n) = 2^{-\Omega(n^\epsilon)}$.*

Proof. The construction maintains the correctness of the commitment by Proposition 5.10.

We shall argue that the construction satisfies strong hiding per Definition 5.5 for adversaries with space up to $\frac{n^2}{4}$. Let $\mathcal{A}$ be some adversary with space $s \leq \frac{n^2}{3}$. Let $g(A)$ be the state of the adversary after streaming A. By Lemma 5.12, the statistical distance between $(r, K, r^T AK, g(A))$ and $(\mathcal{U}, g(A))$ is bounded by $2^{-\frac{n}{17}}$. Thus, we have that the commitment where $\mathcal{A}$ does not receive the key is strongly computationally hiding. Therefore, according to Proposition 5.11, the commitment is hiding for any adversary $\mathcal{A}'$ with space at most $\frac{n^2}{3} - 2n \cdot \frac{n}{24} = \frac{n^2}{4}$.

Finally, we show that the construction is statistically binding. Note that any bit string sent by a malicious sender can be interpreted as a matrix $A \in \{0,1\}^{n\times n}$ and $com = (r, K, w)$, and so the unique vector v which this commitment defines is $v^T = w^T - r^T AK$. Let L be the matrix chosen by the receiver and let $\kappa = AKL$ be the valid key for the commitment. By the correctness of the construction, given κ the receiver outputs $v^T L$. Let $\kappa' \neq AKL$ (we can assume that $|\kappa'| = |AKL|$, otherwise the receiver rejects), and denote $\kappa^* = AKL - \kappa' \neq 0$. For the receiver not to reject, κ' must satisfy $T\kappa' = (TAK)L$, which implies $T\kappa^* = 0$. This means that breaking the commitment is at least as hard as finding a vector in the kernel of T. Note that T is drawn at random and the adversary has no access to it. And so, by the random subset-sum principle, the probability that any non-zero column of κ^* is in the kernel of T is exactly $2^{-\lambda}$, where $\lambda = O(n^\epsilon)$ is the number of rows of T. Thus, the probability that $\kappa^* \neq 0$ satisfies $T\kappa^* = 0$ is at most $2^{-\Omega(n^\epsilon)}$, as required. □

Theorem 5.8 follows directly from Proposition 5.6 and Lemma 5.13.

5.2 Polynomial Commitments

In this section, we use the linearly homomorphic commitment scheme of Section 5.1 to construct a *polynomial commitment scheme*. In such a scheme, a sender can commit to a polynomial, and de-commit to evaluations of the polynomial at particular points such that the de-commitment does not reveal any other information about the polynomial. We consider the case of multivariate polynomials, but note that the specialization to univariate polynomials will be useful for us as well.

Definition 5.14. *Let n denote the security parameter, let $\mathbb{F} = (\mathbb{F}_n)_{n\in\mathbb{N}}$ be a finite field, let $d = d(n), m = m(n)$ be positive integers and let $\ell = (d+1)^m \cdot \log|\mathbb{F}|$. An $(s_{\mathcal{S}}(n,\ell), s_{\mathcal{R}}(n), s_h(n), \delta_h(n), \delta_b(n))$* statistically binding, streaming polynomial commitment *is a protocol executed in two stages called the commit stage and the reveal stage, between a probabilistic ROTM sender with space $s_{\mathcal{S}}(n,\ell)$ and a probabilistic ROTM receiver with space $s_{\mathcal{R}}(n)$.*

In the commit stage, the sender gets as an explicit input an m-variate polynomial $P : \mathbb{F}^m \to \mathbb{F}$ of individual degree d represented as a list of $(d+1)^m$ coefficients, and sends to the receiver as a stream a commitment com_P to the polynomial P. As the receiver reads com_P, it computes a function $g(com_P)$, where $g : \{0,1\}^* \to \{0,1\}^{s_{\mathcal{R}}(n)}$ is a randomized function. In the reveal stage, the receiver sends to the sender a set $X \subseteq \mathbb{F}$ of cardinality j, and in response, the sender sends a key κ. The receiver runs a deterministic algorithm $\mathsf{Verify}(X, \kappa, g(com_P))$, that either outputs a function $f : X \to \mathbb{F}$, or rejects.

The commitment satisfies the following three conditions:

1. ***Correctness.*** *If the two parties act honestly, for every $P : \mathbb{F}^m \to \mathbb{F}$ and $X \subseteq \mathbb{F}^m$, it holds that $\Pr[\mathsf{Verify}(X, \kappa, g(com_P)) = P|_X] = 1$, where $P|_X$ is the vector of evaluations of the polynomial P over the set X, and the probability is taken with respect to the coins of the receiver and the sender.*

2. ***Computational hiding against adversarial ROBP receivers with space $s_h(n)$.*** *For any function $g^* : \{0,1\}^* \to \{0,1\}^{s_h(n)}$ computable in small space, any set $X \subseteq \mathbb{F}^m$, and any two m-variate, individual degree d polynomials $P, Q : \mathbb{F}^m \to \mathbb{F}$ such that $P|_X = Q|_X$ the statistical distance between the distributions $\big(g^*(com_P), \kappa\big)$ and $\big(g^*(com_Q), \kappa\big)$ is at most $\delta_h(n)$.*

3. ***Statistical Binding.*** *For any com, there exists an m-variate polynomial P of individual degree d, such that for any set $X \subseteq \mathbb{F}^m$, and any κ, it holds that $\Pr[\mathsf{Verify}(X, \kappa, g(com)) \in \{rej, P|_X\}] > 1 - \delta_b(n)$, where the probability is taken over the receiver's private randomness for computing $g(com)$.*

Similarly to Definition 5.2, the conditions for hiding and binding as stated in Definition 5.14 naturally encapsulate the natural notions of hiding and binding as explained in Remarks 5.3 and 5.4. Here too, the strong hiding property can be restated as a game against a challenger, similarly to Section 5.1.1, where in Step 1 instead of sending a matrix L and two vectors v_0, v_1 such that $v_0^T L = v_1^T L$, the adversary will send a set of evaluation points X and two polynomials g_0, g_1 such that $g_0|_X = g_1|_X$.

Observe that the evaluation of a polynomial at a particular point can be represented as a linear function of the coefficient vector. Thus, Construction 5.9 naturally fits this purpose, with the exception that it is an implementation of a commitment for vectors over GF(2), and we would like to use it for polynomials that are defined over larger fields.

We circumvent this issue by restricting our attention to extension fields of $\mathrm{GF}(2)$ while leveraging the fact that linear functions over extension fields can be expressed as linear functions over the base field. This is formalized in the following proposition.

Proposition 5.15. *Let $\mathbb{F} = \mathrm{GF}(2^f)$, where $f \geq 1$ and fix a representation of the field. For $x \in \mathbb{F}$ we denote by $\bar{x} \in \{0,1\}^f$ the representation of x as a binary vector of length f. Any linear operation $\ell : \mathbb{F}^d \to \mathbb{F}$ can be represented as a linear operation $\overline{\ell} : \{0,1\}^{fd} \to \{0,1\}^f$ defined as $\overline{\ell}(\overline{v}) = \overline{\ell(v)}$, where $v \in \mathbb{F}^d$ and $\overline{v} \in \{0,1\}^{fd}$ is the concatenation of the vector representations over $\{0,1\}^f$ of the d elements of v.*

Proof. We show that $\overline{\ell}$ is a linear operation over $\mathrm{GF}(2)$. Indeed, for scalar multiplication observe that

$$\overline{\ell}(0 \cdot \overline{v}) = \overline{\ell}(\overline{0}) = \overline{\ell(0)} = \overline{0} = 0 \cdot \overline{\ell}(\overline{v})$$
$$\overline{\ell}(1 \cdot \overline{v}) = \overline{\ell}(\overline{v}) = 1 \cdot \overline{\ell}(\overline{v}).$$

Likewise, let $v, u \in \mathbb{F}^d$ and $\overline{v}, \overline{u}$ be their representations over $\{0,1\}^{fd}$. Let $\oplus$ denote addition over $\{0,1\}$ and let $+$ denote addition over $\mathbb{F}$. And so,

$$\overline{\ell}(\overline{v} \oplus \overline{u}) = \overline{\ell}(\overline{v + u}) = \overline{\ell(v + u)} = \overline{\ell(v) + \ell(u)} = \overline{\ell(v)} \oplus \overline{\ell(u)} = \overline{\ell}(\overline{v}) \oplus \overline{\ell}(\overline{u}).$$

□

Remark 5.16. *Expanding on Proposition 5.15, for $\mathbb{F} = \mathrm{GF}(2^f)$ we note that a matrix $L \in \mathbb{F}^{n \times m}$ representing a linear transformation $L : \mathbb{F}^n \to \mathbb{F}^m$ can be expressed as a matrix $\overline{L} \in \{0,1\}^{n \cdot f \times m \cdot f}$, where every entry in L becomes an $f \times f$ block in $\overline{L}$. The (i,j)-th entry in each such block corresponding to a single field element v, is the j-th coordinate of the vector representation of $v \cdot e_i$, where e_i is the field element corresponding to the i-th basis vector of the given field representation (and the multiplication is over $\mathbb{F}$). Note that a naive quadratic time algorithm can compute the multiplication in space linear in f. Thus, each such block is $O(f)$-strongly explicit.*

Lemma 5.17. *Let n be a security parameter, let $\epsilon > 0$, and let $\mathbb{F} = \mathrm{GF}(2^f)$ for some $f \geq 1$. Let j be the cardinality of the set of evaluations requested by the receiver. Let the committed polynomial be an m-variate polynomial of individual degree d, and let $\ell = (d+1)^m \cdot f$. For all $m, d \geq 1$ such that $\log(m) \leq n^{1+\epsilon}$ and $j \leq \min\left\{(d+1)^m, \frac{n}{24f}\right\}$, there exists an $\big(s_{\mathcal{S}}(n,\ell), s_{\mathcal{R}}(n), s_h(n), \delta_h(n), \delta_b(n)\big)$ statistically binding, streaming polynomial commitment, such that $s_{\mathcal{S}}(n,\ell) = O(n \cdot \ell)$, $s_{\mathcal{R}}(n) = O\big(n^{1+\epsilon}\big)$, $s_b(n) = \frac{n^2}{4}$, $\delta_h(n) = 2^{-\frac{n}{17}}$, and $\delta_b(n) = 2^{-\Omega(n^\epsilon)}$.*

Proof. Using Proposition 5.15, we can commit to m-variate polynomials of individual degree d defined over $\mathbb{F} = \mathrm{GF}(2^f)$ by committing to the binary representation of $(d+1)^m$ field elements (the polynomial's coefficients), as a binary vector v of length $(d+1)^m \cdot f$. Likewise, an evaluation of the polynomial can be obtained by applying a linear function on v, and interpreting the result as a binary representation of a field element.

Let $L \in \{0,1\}^{((d+1)^m \cdot f) \times (j \cdot f)}$ be a matrix representing the linear transformation operating on the binary representation of an m-variate, individual degree d polynomial defined over $\mathbb{F}$, that evaluates the polynomial over a set X of j field elements, and represented as the set X. We want to show that L is $O(f + \log(m))$-strongly explicit. We first argue that the matrix $L' \in \mathbb{F}^{(d+1)^m \times j}$ is

$O(f+\log(m))$-strongly explict, where L' is the matrix representing the evaluation of the polynomial over the set X, but over the field $\mathbb{F} = \mathrm{GF}(2^f)$ instead of over $\{0,1\}$.

The succinct representation of L' will be the list of j points at which the committed polynomial is to be evaluated, which takes up $j \cdot f$ space. Recall that the rows a of L' correspond to coefficients of the polynomial, and columns $b \in \mathbb{F}^m$ correspond to evaluation points. Thus, the (a,b)-entry, where $a = (i_1, \ldots, i_m)$ corresponds to the monomial $x_1^{i_1} \cdot \ldots \cdot x_m^{i_m}$ and $b = (b_1, \ldots, b_m)$ is the evaluation point, is simply $b_1^{i_1} \cdot \ldots \cdot b_m^{i_m}$, which can be computed in space $O(\log(m) + f)$.

The succinct representation of L will be similar to that of L'. In order to compute the entry at row a and column b of L, we first compute the value $v \in \mathbb{F}$ at row $a' = \lfloor a/f \rfloor$ and column $b' = \lfloor b/f \rfloor$ of L'. We return the bit at coordinate $(a \mod f, b \mod f)$ of the binary block corresponding to v as per Remark 5.16. Thus, Construction 5.9 with $\ell = (d+1)^m \cdot f$ is a polynomial commitment with the required parameters.

The hiding and binding properties follow immediately from the hiding and binding of Construction 5.9. □

5.3 Proof of Lemma 5.12

In this section we prove Lemma 5.12 restated as follows.

Lemma 5.18 (Lemma 5.12, restated)**.** *Let $r \leftarrow \{0,1\}^n$, $K \leftarrow \{0,1\}^{n\times\ell}$ and $A \leftarrow \{0,1\}^{n\times n}$, where $\ell \leq \frac{n}{8}$ Let $f : \{0,1\}^{n\times n} \to \{0,1\}^s$ be an arbitrary function, where $s \leq \frac{n^2}{3}$. Then:*

$$\mathrm{SD}\big((r, K, r^T AK, f(A)), (\mathcal{U}, f(A))\big) \leq 2^{-\frac{n}{17}}$$

where $\mathcal{U}$ is the uniform distribution over binary vectors of length $n + n\ell + \ell$.

The proof of Lemma 5.12 draws on several useful facts and propositions, which we state and prove below:

Fact 5.19. *Let A, B be distributions defined over a finite space Ω, and let f be some Boolean function. Then,*

$$\Pr_{(a,b)\leftarrow(A,B)}[f(a,b) = 1] = \mathbb{E}_{a'\leftarrow A}\left[\Pr_{(a,b)\leftarrow(A,B)}[f(a,b) = 1 | a = a']\right]$$

Proof.

$$\begin{aligned}\Pr_{(a,b)\leftarrow(A,B)}[f(a,b) = 1] &\overset{1}{=} \sum_{\omega\in\Omega} \Pr_{(a,b)\leftarrow(A,B)}[f(a,b) = 1 | a = \omega] \cdot \Pr_{a\leftarrow A}[a = \omega] \\ &\overset{2}{=} \mathbb{E}_{a'\leftarrow A}\left[\Pr_{(a,b)\leftarrow(A,B)}[f(a,b) = 1 | a = a']\right]\end{aligned}$$

□

Fact 5.20. *Let A, B be some events. Then,*

$$\Pr[A] \leq \Pr[B] + \Pr[A|\neg B].$$

Proof.

$$\begin{aligned}\Pr[A] &\overset{1}{=} \Pr[A \wedge B] + \Pr[A \wedge \neg B] \\ &\overset{2}{=} \Pr[A|B] \cdot \Pr[B] + \Pr[A|\neg B] \cdot \Pr[\neg B] \\ &\overset{3}{\leq} \Pr[B] + \Pr[A|\neg B].\end{aligned}$$

□

Fact 5.21. *Let (A, B) and (A, B') be two joint distributions that share the similar marginal distribution A. Then,*

$$\|(A, B) - (A, B')\|_1 = \mathbb{E}_A\left[\, \left\|(B|A) - (B'|A)\right\|_1 \right].$$

Proof.

$$\begin{aligned}\|(A, B) - (A, B')\|_1 &\overset{1}{=} \frac{1}{2} \sum_{a,b} \Big| \Pr[A = a, B = b] - \Pr[A = a, B' = b] \Big| \\ &\overset{2}{=} \frac{1}{2} \sum_a \sum_b \Big| \Pr[B = b|A = a] - \Pr[B' = b|A = a] \Big| \cdot \Pr[A = a] \\ &\overset{3}{=} \mathbb{E}_A \left[\frac{1}{2} \sum_b \Big| \Pr[B = b|A = a] - \Pr[B' = b|A = a] \Big| \right] \\ &\overset{1}{=} \mathbb{E}_A\left[\, \left\|(B|A) - (B'|A)\right\|_1 \right].\end{aligned}$$

□

The following fact follows readily from the Cauchy-Schwartz inequality.

Fact 5.22. *Let $X \in \{0, 1\}^N$. Then,*

$$\|X\|_1 \leq \sqrt{N} \cdot \|X\|_2.$$

Fact 5.23. *Let X be a probability distribution over a space Ω of cardinality $N = |\Omega|$, and let $\mathcal{U}$ be the uniform distribution over Ω. Then,*

$$\|X - \mathcal{U}\|_2^2 = \|X\|_2^2 - \frac{1}{N}.$$

Proof. Let $p_\omega = \Pr_X[X = \omega]$. Then by the definition of the ℓ_2 norm,

$$\begin{aligned}\|X - \mathcal{U}\|_2^2 &= \sum_{\omega \in \Omega} \left(p_\omega^2 - 2\frac{1}{N} p_\omega + \frac{1}{N^2} \right) \\ &= \sum_\omega p_\omega^2 - 2\frac{1}{N} \sum_\omega p_\omega + \sum_\omega \frac{1}{N^2} \\ &= \|X\|_2^2 - \frac{1}{N}.\end{aligned}$$

□

Proof.

$$\|X\|_2^2 = \sum_{\omega\in\Omega}\left(\Pr_{x\leftarrow X}[x=\omega]\right)^2 = \sum_{\omega}\Pr_{x\leftarrow X}[x=\omega]\Pr_{x'\leftarrow X}[x'=\omega] = \Pr_{x,x'\leftarrow X}[x=x'] = \mathrm{CP}(X),$$

□

Corollary 5.24. *Let X be a probability distribution over a space Ω of cardinality N, let $\mathcal{U}$ be the uniform distribution over Ω, and let $\epsilon > 0$. If $\mathrm{CP}(X) \le \frac{1+\epsilon}{N}$, then $\mathrm{SD}(X,\mathcal{U}) \le \frac{\sqrt{\epsilon}}{2}$.*

Proof. Recall that the collision probability of a distribution is equal to the ℓ_2 norm of its pdf. And so, by Facts 5.22 and 5.23

$$2\cdot\mathrm{SD}(X,\mathcal{U}) = \|X-\mathcal{U}\|_1 \le \sqrt{N}\cdot\|X-\mathcal{U}\|_2 \le \sqrt{N}\cdot\sqrt{\|X\|^2-\frac{1}{N}} = \sqrt{N\cdot\mathrm{CP}(X)-1} \le \sqrt{\epsilon}.$$

□

Proposition 5.25. *The number of matrices $A \in \{0,1\}^{m\times n}$ with rank at most r is at most $2^{(m+n)\cdot r}$.*

Proof. Consider the following *non-deterministic* algorithm for constructing a matrix $A \in \{0,1\}^{m\times n}$ of rank at most r. Choose r pairs of binary vectors of length m, n respectively, and compute the tensor product of each pair and output the sum of all products. Note that there are $2^{(m+n)\cdot r}$ possible different executions of the algorithm. Since every rank-r matrix can be expressed as the sum of r rank-1 matrices, and every rank-1 matrix can be expressed by a tensor product of two vectors, we get that for every possible matrix A with rank at most r, there exists an execution of the algorithm where it outputs A.

Thus, there can be no more than $2^{(m+n)\cdot r}$ matrices $M \in \{0,1\}^{m\times n}$ with rank at most r.

□

Proposition 5.26. *Let $f : \{0,1\}^{n\times n} \to \{0,1\}^s$ be an arbitrary function and for $A \in \{0,1\}^{n\times n}$ let $\Gamma(A)$ be the set of all matrices $A' \in \{0,1\}^{n\times n}$ such that $f(A) = f(A')$. Let $\mathcal{LR}$ be the set of all $n\times n$ binary matrices with rank at most $\frac{n}{4}$. Then,*

$$\mathop{\mathbb{E}}_{A\leftarrow\{0,1\}^{n\times n}}\left[\Pr_{A',A''\leftarrow\Gamma(A)}\left[(A'\oplus A'')\in\mathcal{LR}\right]\right] \le 2^{s-\frac{n^2}{2}}.$$

Proof. Denote by $\mathcal{LR}\oplus A$ the set of all matrices A' such that $A\oplus A' \in \mathcal{LR}$. Fix some $A \in \{0,1\}^{n\times n}$ and consider the expression inside the expectation. Using Fact 5.19 we can write

$$\Pr_{A',A''\leftarrow\Gamma(A)}\left[(A'\oplus A'')\in\mathcal{LR}\right] = \mathop{\mathbb{E}}_{A'\leftarrow\Gamma(A)}\left[\Pr_{A''\leftarrow\Gamma(A)}\left[(A'\oplus A'')\in\mathcal{LR}\right]\right]. \quad (1)$$

Fix some A' and once again consider the expression inside the expectation:

$$\Pr_{A''\leftarrow\Gamma(A)}\left[(A'\oplus A'')\in\mathcal{LR}\right] = \Pr_{A''\leftarrow\Gamma(A)}\left[A''\in\mathcal{LR}\oplus A'\right] \le \frac{|\mathcal{LR}|}{|\Gamma(A)|} \le \frac{2^{\frac{n^2}{2}}}{|\Gamma(A)|} \quad (2)$$

where the cardinality of $\mathcal{LR}$ is bounded according to Proposition 5.25.

Combining Eq. (1) and Eq. (2), we get

$$\Pr_{A',A''\leftarrow\Gamma(A)}\left[(A'\oplus A'')\in\mathcal{LR}\right]\le\frac{2^{\frac{n^2}{2}}}{|\Gamma(A)|}.$$

Plugging in the original expression of the Lemma's statement:

$$\begin{aligned}
\mathop{\mathbb{E}}_{A\leftarrow\{0,1\}^{n\times,n}}\left[\Pr_{A',A''\leftarrow\Gamma(A)}\left[(A'\oplus A'')\in\mathcal{LR}\right]\right] &\le \mathbb{E}_A\left[\frac{2^{\frac{n^2}{2}}}{|\Gamma(A)|}\right]\\
&= 2^{\frac{n^2}{2}}\cdot\mathbb{E}_A\left[\frac{1}{|\Gamma(A)|}\right]\\
&= 2^{\frac{n^2}{2}}\sum_{\alpha\in\{0,1\}^s}\Pr_A[f(A)=\alpha]\cdot\frac{1}{|f^{-1}(\alpha)|}\\
&= 2^{\frac{n^2}{2}}\sum_{\alpha\in\{0,1\}^s}\frac{|f^{-1}(\alpha)|}{2^{n^2}}\cdot\frac{1}{|f^{-1}(\alpha)|}\\
&= 2^{\frac{n^2}{2}}\cdot 2^s\cdot 2^{-n^2}\\
&= 2^{s-\frac{n^2}{2}}.
\end{aligned}$$

□

We are now ready to prove our main lemma.

Proof of Lemma 5.12. Let $\Gamma(A)$ be the set of all A' such that $f(A')=f(A)$, and let $\tilde{A}$ denote a random variable for a uniformly sampled matrix from $\Gamma(A)$. From Fact 5.21 and the definition of statistical difference we get

$$\begin{aligned}
\mathrm{SD}((r,K,r^TAK,f(A)),(\mathcal{U},f(A))) &= \frac{1}{2}\left\|(r,K,r^TAK,f(A))-(\mathcal{U},f(A))\right\|_1\\
&= \frac{1}{2}\mathop{\mathbb{E}}_{A}\left[\left\|(r,K,r^TAK|f(A))-(\mathcal{U}|f(A))\right\|_1\right]\\
&= \frac{1}{2}\mathop{\mathbb{E}}_{A}\left[\left\|(r,K,r^T\tilde{A}K)-\mathcal{U}\right\|_1\right]\\
&= \mathop{\mathbb{E}}_{A}\left[\mathrm{SD}((r,K,r^T\tilde{A}K),\mathcal{U})\right].
\end{aligned}\tag{3}$$

Using Corollary 5.24 we can bound the quantity above as follows:

$$\begin{aligned}
\mathop{\mathbb{E}}_{A}\left[\mathrm{SD}((r,K,r^T\tilde{A}K),\mathcal{U})\right] &\le \frac{1}{2}\mathop{\mathbb{E}}_{A}\left[\sqrt{2^{n+n\ell+\ell}}\cdot\sqrt{\mathrm{CP}(r,K,r^T\tilde{A}K)-2^{-(n+n\ell+\ell)}}\right]\\
&\le 2^{\frac{n+n\ell+\ell}{2}}\cdot\sqrt{\mathop{\mathbb{E}}_{A}\left[\mathrm{CP}(r,K,r^T\tilde{A}K)\right]-2^{-(n+n\ell+\ell)}},
\end{aligned}\tag{4}$$

where the last inequality follows from the Jensen inequality, with respect to the square root function, which is indeed concave.

Zooming in on the collision probability of $(r, K, r^T \tilde{A}K)$, we compute it as

$$\begin{aligned}
\mathrm{CP}(r, K, r^T \tilde{A}K) &= \Pr_{r_1,r_2,K_1,K_2,\tilde{A}_1,\tilde{A}_2}\left[r_1 = r_2, K_1 = K_2, r_1^T \tilde{A}_1 K_1 = r_2^T \tilde{A}_2 K_2\right] \\
&= \Pr_{r_1,r_2,K_1,K_2,\tilde{A}_1,\tilde{A}_2}\left[r_1^T \tilde{A}_1 K_1 = r_2^T \tilde{A}_2 K_2 \,|\, r_1 = r_2, K_1 = K_2\right] \cdot \Pr_{r_1,r_2}[r_1 = r_2] \cdot \Pr_{K_1,K_2}[K_1 = K_2] \\
&= 2^{-n-n\ell} \cdot \Pr_{r,K,\tilde{A}_1,\tilde{A}_2}\left[r^T(\tilde{A}_1 \oplus \tilde{A}_2)K = 0\right].
\end{aligned} \tag{5}$$

Let $\mathcal{LR}$ be the set of all $n \times n$ matrices with rank at most $\frac{n}{4}$. By Fact 5.20 we have that

$$\begin{aligned}
\Pr_{r,K,\tilde{A}_1,\tilde{A}_2}\left[r^T(\tilde{A}_1 \oplus \tilde{A}_2)K = 0\right] &\leq \Pr_{\tilde{A}_1,\tilde{A}_2}[(\tilde{A}_1 \oplus \tilde{A}_2) \in \mathcal{LR}] + \Pr_{r,K,\tilde{A}_1,\tilde{A}_2}\left[r^T(\tilde{A}_1 \oplus \tilde{A}_2)K = 0 \,|\, (\tilde{A}_1 \oplus \tilde{A}_2) \notin \mathcal{LR}\right] \\
&\leq \Pr_{\tilde{A}_1,\tilde{A}_2}\left[(\tilde{A}_1 \oplus \tilde{A}_2) \in \mathcal{LR}\right] \\
&\quad + \Pr_{r,\tilde{A}_1,\tilde{A}_2}\left[r \in \mathrm{Ker}(\tilde{A}_1 \oplus \tilde{A}_2)^T \,|\, (\tilde{A}_1 \oplus \tilde{A}_2) \notin \mathcal{LR}\right] \\
&\quad + \Pr_{r,K,\tilde{A}_1,\tilde{A}_2}\left[r^T(\tilde{A}_1 \oplus \tilde{A}_2)K = 0 \,|\, (\tilde{A}_1 \oplus \tilde{A}_2) \notin \mathcal{LR}, r \notin \mathrm{Ker}(\tilde{A}_1 \oplus \tilde{A}_2)^T\right].
\end{aligned} \tag{6}$$

The quantity in the left-hand side of Eq. (6) is bounded by the sum of three expressions. According to Proposition 5.26 the expectation over A of the first expression is bounded from above by $2^{s-\frac{n^2}{2}}$. Note that for a matrix $A \in \{0,1\}^{n \times n}$ of rank $\frac{n}{4}$ we have that

$$\Pr_{k \leftarrow \{0,1\}^n}[k \in \mathrm{Ker}(A)] = \frac{|\mathrm{Ker}(A)|}{2^n} = \frac{2^{n-\frac{n}{4}}}{2^n} = 2^{-\frac{n}{4}}.$$

Thus, the second expression is bounded from above by $2^{-\frac{n}{4}}$. The third expression can be written as

$$\Pr_K\left[u^T K = 0\right] \leq 2^{-\ell}$$

where u is a non-zero vector of length n, and the bound is due to the random subset-sum principle, and the fact that the columns of K are independent.

By combining Eqs. (5) and (6) we get the following bound:

$$\mathop{\mathbb{E}}_A\left[\mathrm{CP}(r, K, r^T \tilde{A}K)\right] \leq 2^{-(n+n\ell+\ell)} + 2^{s-\frac{n^2}{2}-n-n\ell} + 2^{-(\frac{5n}{4}+n\ell)} \leq 2^{-(n+n\ell+\ell)} + 2^{-(\frac{9n}{8}+n\ell+\ell-1)}, \tag{7}$$

where the second inequality results from the choice of parameters $s \leq \frac{n^2}{3}$ and $\ell \leq \frac{n}{8}$.

Finally, combining Eq. (7) with Eq. (3) and Eq. (4) we get

$$\mathrm{SD}\Big(\big(r, K, r^T AK, f(A)\big), \big(\mathcal{U}, f(A)\big)\Big) \leq 2^{\frac{n+n\ell+\ell}{2}} \cdot \sqrt{2^{-\left(\frac{9n}{8}+n\ell+\ell-1\right)}} \leq 2^{-\frac{n}{17}}.$$

This concludes the proof of Lemma 5.12. □

6 Zero-Knowledge Streaming Interactive Protocol for PEP

In this section, we present a zero-knowledge streaming interactive proof for the decision version of the Polynomial Evaluation Problem defined as the following language.

Definition 6.1 (Polynomial Evaluation Problem)**.** *Let $\mathbb{F}$ be a finite field. Let $P : \mathbb{F}^m \to \mathbb{F}$ be an m-variate polynomial of individual degree-d, represented as a vector of evaluations at $(d+1)^m$ distinct points in $\mathbb{F}$ Let $x \in \mathbb{F}^m$ and $\alpha \in \mathbb{F}$. Then,*

$$\mathsf{PEP} = \{(P, x, \alpha) \mid P(x) = \alpha\}\,.$$

We construct a zero-knowledge streaming interactive proof for PEP using a statistically hiding streaming commitment per Definition 4.1 and a polynomial commitment per Definition 5.14. Throughout this section and the next we denote the parameters of the statistically hiding commitment (resp. polynomial commitment) as $\left(s^s_{\mathcal{S}}, s^s_{\mathcal{R}}, s^s_b, \delta^s_b, \delta^s_h\right)$ (resp. $\left(s^p_{\mathcal{S}}, s^p_{\mathcal{R}}, s^p_h, \delta^p_b, \delta^p_h\right)$), and its communication complexity as cc^s (resp. cc^p).[12] Recall that $s^{\{s,p\}}_{\mathcal{S}}, s^{\{s,p\}}_{\mathcal{R}}$ indicate the minimal space required by the ROTM sender and receiver, respectively, to execute the commitment, s^s_b (resp. s^p_h) is the maximal space of a malicious ROBP sender (resp. receiver) against which the commitment is binding (resp. hiding), and that $\delta^{\{s,p\}}_b, \delta^{\{s,p\}}_h$ are the binding and hiding errors of the commitments, respectively. Note that these parameters are a function of the commitments' security parameters and length of the committed strings. The commitments' security parameters and length of the committed strings are in turn a function of the protocol's input length n, and so in order to abstract the inner workings of the commitments, we treat the parameters listed above as functions of n. As our protocol uses abstract commitments, we implicitly assume that when instantiating the protocol with particular commitment schemes, the prover and verifier also receive as auxiliary input the security parameters and any other information required to execute the commitments.

By constructing a protocol for PEP, we shall prove our first main theorem:

Theorem 6.2. *Let $m, d \in \mathbb{N}^+$, and let $\mathbb{F}$ be a finite field. Denote the space of the honest verifier by s. Let $\left(s^s_{\mathcal{S}}, s^s_{\mathcal{R}}, s^s_b, \delta^s_b, \delta^s_h\right)$ (resp. $\left(s^p_{\mathcal{S}}, s^p_{\mathcal{R}}, s^p_h, \delta^p_b, \delta^p_h\right)$) be the parameters of a statistically hiding streaming commitment per Definition 4.1 (resp. polynomial commitment per Definition 5.14), such that $s^s_{\mathcal{R}} = O(s), s^p_{\mathcal{S}} = O(s), s^s_b > s$, and $s^p_h > 2s^s_b + s^s_{\mathcal{R}} + s^p_{\mathcal{S}}$. There exists a streaming interactive proof for the polynomial evaluation problem (PEP) for m-variate polynomials of individual degree d defined over $\mathbb{F}$ with a ROTM verifier with space $s = O\big(m\log(|\mathbb{F}|)\big) + s^p_{\mathcal{R}} + s^s_{\mathcal{S}}$ that is zero-knowledge with error $12\sqrt{\delta^s_b} + 4\delta^p_h$ against adversaries with space s^s_b per Definition 3.6. The protocol satisfies perfect completeness and has a soundness error of $\frac{dm}{\mathbb{F}} + \delta^s_h + \delta^p_b$ and communication complexity $m \cdot \log(|\mathbb{F}|) + cc^s + cc^p$.*

From the theorem, applying our commitment schemes constructed in the previous sections, we obtain the following corollary.

[12] Note that the s and p superscripts stand for "statistically hiding commitment" and "polynomial commitment", respectively.

Corollary 6.3. *Let $n = ((d+1)^m + m + 1) \cdot \lceil \log(|\mathbb{F}|) \rceil$ denote the input length. There exists a streaming interactive proof for the polynomial evaluation problem (PEP) for m-variate polynomials of individual degree $d = O(\text{polylog}(n))$ defined over $\mathbb{F}$ such that $|\mathbb{F}| = O(2^{\log^2(n)})$, where the verifier has space $s = \text{polylog}(n)$ and the communication complexity is $\text{polylog}(n)$. The proof has perfect completeness, negligible soundness error, and negligible zero-knowledge error, secure against adversaries with space $s^{2-\delta}$ (for any $\delta > 0$).*

Proof. Let $\delta > 0$. Take Constructions 4.13 and 5.9 with security parameters $\lambda = \log^c(n)$ for some constant c, and $\epsilon > 0$, such that $\epsilon \le \frac{1}{2-\delta} - \frac{1}{2}$.

For $|\mathbb{F}| = O(2^{\log^2(n)})$, $d = O(\text{polylog}(n))$ we have that $(d+1)^m = O\left(\frac{n}{\log^2(n)}\right)$, and so $m = O(\log(n))$. Then, by Theorem 6.2 we get a streaming interactive proof for the polynomial evaluation problem with an honest verifier with space $s = O\big(m\log(|\mathbb{F}|)\big) + s^p_{\mathcal{R}} + s^s_{\mathcal{S}} = \text{polylog}(n) + \log^{c+3}(n) + \log^{c(1+\epsilon)}(n) \le \log^{c(1+2\epsilon)}(n)$, for large-enough constant c.

The protocol satisfies perfect completeness and soundness error $\frac{dm}{|\mathbb{F}|} + 2^{-\log^c(n)} + 2^{-\Omega(\log^{c\epsilon}(n))} = \mathsf{negl}(n)$ for $c \ge \frac{2}{\epsilon}$. It has a zero-knowledge error of $12\sqrt{2^{-\Omega(\log^{c\epsilon})}} + 4 \cdot 2^{-\frac{\log^c(n)}{17}} = \mathsf{negl}(n)$, for $c \ge \frac{2}{\epsilon}$, against adversaries with space $\log^{2c}(n) = (\log^{c(1+2\epsilon)}(n))^{\frac{2}{1+2\epsilon}} \ge s^{2-\delta}$. □

The remaining of this section will be dedicated to proving Theorem 6.2. In Section 6.1 we shall present our construction and show that it is a streaming interactive proof, and in Section 6.2 we will prove that the protocol is a streaming zero-knowledge proof.

6.1 Zero-Knowledge Streaming Interactive Proof for PEP

The protocol is presented in Fig. 1.

We begin by analyzing the complexity of the protocol. used in Protocol 6.4. The prover stores the entire input, and so requires $O\big(n\log(\mathbb{F})\big) = O\big(d^m\log(\mathbb{F})\big)$ space. The main space constraints of the verifier come from the execution of the two commitment schemes. The amount of additional storage used beyond the commitments is $O\big(dm\log(|\mathbb{F}|)\big)$. Thus, in total the verifier has space complexity $O\big(m\log(|\mathbb{F}|)\big) + s^p_{\mathcal{R}} + s^s_{\mathcal{S}}$.

In terms of communication complexity, the total number of bits transferred between the parties is the sum of the communication complexities of the two commitments, plus the single value $L(1) \in \mathbb{F}$ sent in Step 3b. Thus, in total, the communication complexity of the protocol is $m \cdot \log(|\mathbb{F}|) + cc^s + cc^p$.

Proposition 6.5. *Protocol 6.4 satisfies perfect completeness.*

Proof. Suppose $P(x) = \alpha$ and consider an execution of the protocol with the honest prover. In this case, by construction, $g(0) = P(x) = \alpha$ and $g(\mu) = P(r)$, and so the verifier always accepts. □

Lemma 6.6. *Protocol 6.4 has soundness error $\frac{dm}{\mathbb{F}} + \delta^s_h + \delta^p_b$.*

Proof. Suppose $P(x) \ne \alpha$ and fix a cheating prover strategy. Without loss of generality, assume that the prover is deterministic.[13]

[13] The soundness definition of the protocol applies to all unbounded provers, and in particular to provers that have access to the input prior to it being streamed. Thus, any such a randomized prover can be emulated by a deterministic one which has the optimal choice of randomness hardwired.

Protocol 6.4. Streaming interactive protocol for PEP.
Streaming Input: An m-variate polynomial P of individual degree d over field $\mathbb{F}$, streamed as a vector of evaluations at $(d+1)^m = n$ points, a point $x \in \mathbb{F}^m$, a value $\alpha \in \mathbb{F}$.
Output: 1 if $P(x) = \alpha$, otherwise 0.

1. **Statistically hiding commitment stage**
   (a) The verifier draws at random a point $r \in \mathbb{F}^m$.
   (b) The verifier sends to the prover a commitment to r using a statistically hiding streaming commitment per Definition 4.1.

2. **Streaming stage**
   (a) The prover and verifier stream the input polynomial. While streaming, the verifier evaluates P at point r using Lagrange interpolation. The prover, who has unbounded space, records the values of P at all points in $\mathbb{F}^m$.
   (b) Both parties read the point $x \in \mathbb{F}^m$ at which we want to evaluate the polynomial, and the value $\alpha \in \mathbb{F}$.

3. **Interactive stage**
   (a) The verifier draws at random a non-zero field element $\mu \in \mathbb{F} \setminus \{0\}$.
   (b) The verifier sends to the prover the line $L : \mathbb{F} \to \mathbb{F}^m$ s.t. $L(0) = x$ and $L(\mu) = r$. The line is represented as a single value $L(1)$.[a]
   (c) Using a polynomial commitment per Definition 5.14 with security parameter λ, the prover sends to the verifier a commitment to the univariate polynomial $g : \mathbb{F} \to \mathbb{F}$, defined as $g = P|_L$ (i.e., $g(\cdot) = P(L(\cdot))$) represented as a list of $dm + 1$ coefficients.

4. **De-commitment stage**
   (a) The verifier de-commits to r.
   (b) If the verifier's de-commitment was valid, the prover de-commits to $g(0), g(\mu)$, [b] otherwise it aborts.
   (c) The verifier accepts if and only if:
      i. $g(0) = \alpha$.
      ii. $g(\mu) = P(r)$.

[a] Since the prover already knows the value $L(0)$, that is enough for it to reconstruct the line.

[b] Once the prover knows r, it can compute $\mu = L^{-1}(r)$. Note that unless $r = x$, the function L is bijective and so L^{-1} is well-defined. In the unlikely event that $r = x$, the verifier will have already computed $P(x)$ and does not need to continue the protocol.

Figure 1: Protocol 6.4: Streaming Interactive Proof for PEP

Let u, v be the values to which the prover de-commits in Step 4b, which are to be interpreted by the verifier as $g(0), g(\mu)$ respectively. Note that these are random values that depend on the choices of the verifier. For the prover to pass the first verifier test in Step 4(c)i the first de-committed point must satisfy

$$u = \alpha \neq P(x),$$

and so any degree-dm univariate polynomial h such that $h(0) = u$ and $h(\mu) = v$ satisfies

$$h \neq P|_L.$$

By the definition of the polynomial commitment used by the prover in Step 3c, the commitment uniquely defines a degree-dm polynomial, which we denote by g^*. Denote by E_1 the event that the prover de-commits to evaluations u, v such that $u \neq g^*(0)$ or $v \neq g^*(\mu)$ in Step 4b. By the statistical binding of the polynomial commitment we have that

$$\Pr[E_1] = \delta_b^p.$$

Assume now that E_1 does not occur, and so the polynomial g^* to which the prover *commits* in Step 3c satisfies $g^* \neq P|_L$. Consider the following mental experiment where in Step 1b the verifier commits to zeros, instead of the random r with which it proceeds in the protocol. We emphasize that in this mental experiment r is still used in Step 3b, but the verifier does not commit to it in Step 1b.

Let E_2 denote the event that in the execution of the real protocol the value g^* to which the prover commits satisfies $g^*(L^{-1}(r)) = P(r)$, and let E_2' denote the same event, but in the mental experiment. We note that since in the mental experiment, the prover is completely oblivious to the random point r, and so if $g^* \neq P|_L$, by the fundamental theorem of algebra, the probability that E_2' will occur is at most $\frac{dm}{\mathbb{F}}$. By the statistically hiding property of the verifier's commitment, it must be so that $\Pr[E_2] \leq \Pr[E_2'] + \delta_h^s$. Thus, we conclude that assuming $g^* \neq P|_L$, it must be that $\Pr[E_2] \leq \frac{dm}{\mathbb{F}} + \delta_h^s$.

Hence, if E_1 does not occur, then the only way to pass the first verifier test is for the prover to *commit* to a polynomial $g^* \neq P_L$. Then, in order to pass the second verifier test in Step 4(c)ii ($g^*(\mu) = P(r)$), E_2 must occur. Thus, if both events do not occur, the verifier rejects. Therefore, for the prover to fool the verifier, at least one of the events from E_1, E_2 must occur. The lemma now follows from the union bound. □

6.2 Protocol 6.4 is Zero-Knowledge

To show that Protocol 6.4 is zero-knowledge (under the formal requirements of Definition 3.6) we need to show that the verifier's view during an interaction with the prover is indistinguishable from the output of a simulation of a space bounded machine. To specify the view we enumerate the components that the verifier sees in the protocol, in the order that they are seen:

1. Statistically binding streaming commitment to the value r.

2. Input P, x, α.

3. Polynomial commitment to the polynomial g.

4. De-commitment to g.

Recall that we define that every random bit of the verifier enters its view as soon as it is queried, and so the verifier's randomness is included in the view above, interlaced within the listed components.

6.2.1 Extracting r from the Verifier

To show that Protocol 6.4 satisfies zero-knowledge, for any malicious verifier we construct a simulator that simulates the verifier's view when interacting with the prover.

The key difficulty with simulating the view is that while an unbounded prover can store in its memory the entire input, a simulator cannot. This is a problem since in Step 3c of the protocol, the simulator must commit to a polynomial g that satisfies $g(\mu) = P(r)$, for some r chosen by the verifier, but the verifier reveals the value of r only after g has been committed. This happens only after the streaming of the input has been completed, and by that point it is too late for the simulator to query P at the point r. In traditional zero-knowledge proofs, the *rewinding* technique can be used to recover r and then re-execute the protocol. The problem is that in the streaming context rewinding is not an option (since the simulator can only read the input once).

Thus, rather than rewinding, we leverage the fact that the simulator (in contrast to the prover) has full access to the code of the verifier and in particular to potential randomness that it uses. If for example, the value of r is just taken directly from the randomness string, the simulator could simply read it off of that. The issue with this is that we do not know how the verifier will generate r in advance, and trying to analyze the "code" of the verifier seems to be extremely complex.

To get around this, we take advantage of our modeling of the simulator as a read-once branching program *that can execute arbitrarily complex computations* in order to decide how to transition from state to state. Hence, the simulator can *compute* a function $r' = r'(st_{\mathcal{V}^*}, st_{\mathcal{R}})$, where $st_{\mathcal{V}^*}$ is the contents of the tape of the verifier when it sends the commitment, and $st_{\mathcal{R}}$ is the contents of the tape of the honest receiver of the commitment, simulated by the simulator, when it receives the commitment. Our goal is for the function r' to predict the value of r to which the verifier will de-commit, depending on the state to which the parties involved in the commitment arrived at the time of the commitment. We emphasize that the verifier is only *computationally bound* to the value r and so it is not a priori clear that even an unbounded simulator can simply "break" the commitment to extract the value.

We stress that the value of r' depends only on the transcript that occurred *prior* to the streaming of the input, while the verifier's behavior at the time of the de-commitment can change depending on the input stream, and the rest of the communication transcript that comes between the time the verifier commits to r and the time it de-commits. Therefore, arguing that there exists such a value r' that satisfies the above is non-trivial and is captured by the following proposition.

Proposition 6.7. *Fix a cheating prover $\mathcal{V}^*$ with space up to s_b^s and denote by $\rho_{\mathcal{V}^*}$ the randomness used by $\mathcal{V}^*$ to execute the statistically hiding streaming commitment in Step 1b of Protocol 6.4. Denote by $\rho_{\mathcal{R}}$ the randomness used by the honest receiver to execute the commitment. Denote by $st_{\mathcal{V}^*}, st_{\mathcal{R}}$ the internal state of the verifier and the receiver, respectively, after sending the commitment using randomness $\rho_{\mathcal{V}^*}$ and $\rho_{\mathcal{R}}$. Let $\rho'_{\mathcal{P}}, \rho'_{\mathcal{V}^*}$ be the randomness of the prover and verifier, respectively, for the execution of Steps 3 through 4a of the protocol. Let $r = r(st_{\mathcal{V}^*}, P, x, \alpha, \rho'_{\mathcal{P}}, \rho'_{\mathcal{V}^*})$ be the value to which the verifier de-commits in Step 4a when committing with state $st_{\mathcal{V}^*}$, streaming the input (P, x, α), and exchanging messages in Steps 3 through 4a using $\rho'_{\mathcal{V}^*}$ with the prover using $\rho'_{\mathcal{P}}$. In case the receiver rejects the verifier's de-commitment in Step 4a then we define $r = \perp$.*

Then, there exists a function $r' : \{0,1\}^* \to \mathbb{F}^m \cup \{\bot\}$ such that for all $(P, x, \alpha, \rho'_{\mathcal{P}}, \rho'_{\mathcal{V}^*})$,

$$\Pr_{\rho_{\mathcal{V}^*}, \rho_{\mathcal{R}}}\Big[\big(r'(st_{\mathcal{V}^*}, st_{\mathcal{R}}) \neq r\big) \wedge \big(r \neq \bot\big)\Big] \leq 3\sqrt{\delta_b^s},$$

where $r = r(st_{\mathcal{V}^*}, P, x, \alpha, \rho'_{\mathcal{P}}, \rho'_{\mathcal{V}^*})$.

The key benefit of Proposition 6.7 is that the function r' can predict r based solely on the commitment to r made by the verifier *prior* to the streaming of the input. We note that Proposition 5.4 leaves the possibility that $r' \neq r$ when $r = \bot$. Fortunately, this case is not problematic for us since if $r = \bot$ the simulator does not need to open its de-commitment in Step 4b.

We proceed to the proof of the proposition.

Proof of Proposition 6.7. We define the function r' by presenting a (highly inefficient) algorithm that computes it. We stress that while r' is not efficiently computable, it depends *only* on the internal state of the verifier and the receiver of the commitment, and so the simulator will be able to compute it.

Since the verifier's behaviour may depend on everything it sees during the interaction, we catalogue the value to which it de-commits for any possible transcript of the protocol in a matrix M, where the columns represent all the possible pairs of states $(st_{\mathcal{V}^*}, st_{\mathcal{R}})$ at the time of the commitment, and the rows represent all possible protocol executions $(P, x, \alpha, \rho'_{\mathcal{P}}, \rho'_{\mathcal{V}^*})$ from after the commitment up to the de-commitment (which also include the streaming of the input). Denote the number of columns in M as n ($n = 2^{|st_{\mathcal{V}^*}|+|st_{\mathcal{R}}|}$) and the number of rows as m. The entry M_{ij} is the value that is de-committed by the verifier when committing with the state combination j and continuing the protocol with execution i. If the verifier fails to properly de-commit in this configuration, we set $M_{ij} = \bot$.

Note that the states at the time of the commitment depend on the randomness $\rho_{\mathcal{V}^*}, \rho_{\mathcal{R}}$ with which $\mathcal{V}^*$ and $\mathcal{R}$ execute the commitment. Thus, some state combinations may be more likely than others. To account for that, we define a distribution w over the columns of M, such that the probability of sampling a column j from w is equal to the probability of uniformly selecting a pair $(\rho_{\mathcal{V}^*}, \rho_{\mathcal{R}})$ that will result in the state combination represented by the column j.

To define r', we must assign a single value to each column of M. If the columns were entirely uniform (i.e., each column had the same value), this value would naturally be the sole candidate for r'. However, the verifier may behave differently for certain *rows*, leading to multiple possible candidates. To address this, we leverage the fact that the verifier is committed to r, ensuring that its behavior cannot be arbitrary, and by such uniquely isolating a specific r' for each column. The following outlines the procedure for selecting this value for r' for each column.

Let $\tilde{M}$ be a *binary* matrix of the same dimensions as M, where

$$\tilde{M}_{ij} = \begin{cases} 0, & M_{ij} = \bot \\ 1, & \text{otherwise} \end{cases}.$$

Consider the following claim.

Claim 6.7.1. *Let $\tilde{M} \in \{0,1\}^{m \times n}$ and let w be a distribution defined over $[n]$ We say that row $i \in [m]$ covers column $j \in [n]$ if $\tilde{M}_{ij} = 1$ and that a set of rows $I \subseteq [m]$ covers column j if there exists a row $i \in I$ that covers column j.*

For every $0 < \delta \leq 1$, there exists a set $I \subseteq [m]$ of rows such that $|I| \leq \lceil 1/\delta \rceil$, and for any row $i^* \in [m]$,

$$\Pr_{j \leftarrow w}[i^* \textit{ covers } j \textit{ and } I \textit{ does not cover } j] \leq \delta.$$

We will use Claim 6.7.1 to show that there exists a relatively small set of rows in M that covers (almost) all its columns, according to whose values we can set r'. The fact that the set is small will be helpful later on.[14]

Proof of Claim 6.7.1. We construct the set I using the following greedy algorithm:

1. Initialize $I \leftarrow \emptyset$.
2. If there exists a row $i \in [m] \setminus I$ such that

$$\Pr_{j \leftarrow w}[i \text{ covers } j \text{ and } I \text{ does not cover } j] > \delta,$$

   then, add i to I and goto Step 2.
3. Return I.

Denote by $p[S]$ the probability that a column drawn from w is covered by set of rows S. Restating the condition in Item 2 we add the row i to the set I if $p[I \cup \{i\}] - p[I] > \delta$, and so $p[I \cup \{i\}] > p[I] + \delta$, i.e. in each iteration the probability of drawing a column j from w such that I covers j, increases by at least δ. Since probabilities are upper bounded by 1, this implies that the number of iterations is at most $\lceil 1/\delta \rceil$. Since we add a single item in each iteration, $|I| \leq \lceil 1/\delta \rceil$, as required.

Second, the stopping condition is exactly what we require of the set I, and so at the end of the algorithm the set I satisfies what is required by the proposition. □

Let $\delta = \sqrt{\delta_b^s}$ and let I be the set of rows in $\tilde{M}$ of size at most $\lceil 1/\delta \rceil$ guaranteed to exist by Claim 6.7.1, and consider this set of rows in the matrix M. Recall that M is our original matrix which contains the de-commitment response, whereas $\tilde{M}$ was a binary matrix that indicated for which entries of M the prover accepted the verifier's de-commitments. Recall that for every row i, the probability that a column drawn from distribution w is covered by i and not by I is at most δ. We define the value $r'(j)$ as follows:

- If for all $i \in I$ it holds that $M_{ij} = \perp$ then set $r'(j) = \perp$.
- If there exist $i, i' \in I$ such that $M_{ij}, M_{i'j} \neq \perp$ but $M_{ij} \neq M_{i'j}$ then arbitrarily set $r'(j) = M_{ij}$.
- Otherwise there exists a unique $v \in \mathbb{F}^m$ such that for all $i \in I$ it holds that $M_{ij} \in \{\perp, v\}$. In this case set $r'(j) = v$.

We now show that r' as defined above satisfies the proposition. To do so it will be convenient to restate the proposition in terms of M and w. Thus, we need to show that for any row $i \in [m]$,

$$\Pr_{j \leftarrow w}\left[r'(j) \neq M_{ij} \wedge M_{ij} \neq \perp\right] \leq 3\sqrt{\delta_b^s}.$$

[14] Jumping ahead, we will be utilizing the binding of the commitment on this set of rows, and we can only afford a union bound on the binding error over a small set of rows.

Note that this event may occur for two reasons: either (1) $r'(j) \neq \perp$ and the value was defined by a row $i^* \in I$ such that $M_{ij} \neq M_{i^*j}$ or (2) $r'(j) = \perp$ as for all rows $i^* \in I$, $M_{i^*j} = \perp$. To account for these cases, we write

$$\Pr_{j \leftarrow w}\Big[\big(r'(j) \neq M_{ij}\big) \wedge \big(M_{ij} \neq \perp\big)\Big] =$$
$$\Pr_{j \leftarrow w}\Big[\big(r'(j) \neq M_{ij}\big) \wedge \big(M_{ij} \neq \perp\big) \wedge \big(r'(j) \neq \perp\big)\Big] + \Pr_{j \leftarrow w}\Big[\big(r'(j) \neq M_{ij}\big) \wedge \big(M_{ij} \neq \perp\big) \wedge \big(r'(j) = \perp\big)\Big].$$

We bound each case separately.

We address the former case first, via the following claim:

Claim 6.7.2. *For any two fixed rows in M, the probability over distribution w of selecting a column in which both rows take non-$\perp$ values and the values disagree is at most δ_b^s.*

Proof. Assume the contrary, and so we can construct an adversarial sender with space at most s_b^s that can break the binding of the commitment. The adversary will simulate the protocol on both transcripts that correspond to the two rows of M and a random column sampled from w, and output the two de-commitments. According to the assumption, this adversary will succeed with probability greater than δ_b^s, in contradiction to the binding guarantee of the commitment per Definition 4.1. □

Recall that the number of rows in I is at most $\lceil 1/\delta \rceil$, and so, by the union bound and Claim 6.7.2, given $i \in [m]$, the probability over selecting a column j from distribution w such that for any $i^* \in I$, it holds that $M_{ij}, M_{i^*j} \neq \perp$ and $M_{ij} \neq M_{i^*j}$ is at most $|I| \cdot \delta_b^s \leq (1/\delta + 1) \cdot \delta_b^s \leq 2\sqrt{\delta_b^s}$. According to the definition of r', in the case where $r'(j) \neq \perp$, there exists $i^* \in I$ such that $r'(j) = M_{i^*j} \neq \perp$, and so we have that

$$\Pr_{j \leftarrow w}\Big[\big(r'(j) \neq M_{ij}\big) \wedge \big(M_{ij} \neq \perp\big) \wedge \big(r'(j) \neq \perp\big)\Big] \leq \Pr_{j \leftarrow w}\Big[\exists i^* \in I : \big(M_{i^*j}, M_{ij} \neq \perp\big) \wedge \big(M_{i^*j} \neq M_{ij}\big)\Big]$$
$$\leq 2\sqrt{\delta_b^s}.$$

As for the second case, by Claim 6.7.1, and selecting the set I with $\delta = \sqrt{\delta_b^s}$, the probability over a selection of a column j from the distribution w that the column j is not covered by I is at most δ, and so

$$\Pr_{j \leftarrow w}\Big[\big(r'(j) \neq M_{ij}\big) \wedge \big(M_{ij} \neq \perp\big) \wedge \big(r'(j) = \perp\big)\Big] \leq \Pr_{j \leftarrow w}\Big[r'(j) = \perp\Big] \leq \sqrt{\delta_b^s}.$$

This concludes the proof of Proposition 6.7. □

Proposition 6.7 considered fixed $\rho'_{\mathcal{P}}, \rho'_{\mathcal{V}^*}$, i.e. the randomness of the prover and verifier for executing the Steps 2, 3 of the protocol. The following immediate corollary shows that it also holds when these are chosen at random.

Corollary 6.8. *Under the same setting as Proposition 6.7, there exists a function $r' : \{0,1\}^* \to \mathbb{F}^m \cup \{\perp\}$ such that for all inputs (P, x, α),*

$$\Pr_{\rho_{\mathcal{V}^*}, \rho_{\mathcal{R}}, \rho'_{\mathcal{P}}, \rho'_{\mathcal{V}^*}}\Big[\big(r'(st_{\mathcal{V}^*}, st_{\mathcal{R}}) \neq r\big) \wedge \big(r \neq \perp\big)\Big] \leq 3\sqrt{\delta_b^s},$$

where $r = r(st_{\mathcal{V}^*}, P, x, \alpha, \rho'_{\mathcal{P}}, \rho'_{\mathcal{V}^*})$.

Proof. Fix some input (P, x, α). Then by Proposition 6.7,

$$\begin{aligned}
&\Pr_{\rho_{\mathcal{V}^*}, \rho_{\mathcal{R}}, \rho'_{\mathcal{P}}, \rho'_{\mathcal{V}^*}} \left[\left(r'(st_{\mathcal{V}^*}, st_{\mathcal{R}}) \neq r(st_{\mathcal{V}^*}, P, x, \alpha, \rho'_{\mathcal{P}}, \rho'_{\mathcal{V}^*}) \right) \wedge \left(r \neq \bot \right) \right] \\
&\quad = \mathop{\mathbb{E}}_{\rho'_{\mathcal{P}}, \rho'_{\mathcal{V}^*}} \left[\Pr_{\rho_{\mathcal{V}^*}, \rho_{\mathcal{R}}} \left[\left(r'(st_{\mathcal{V}^*}, st_{\mathcal{R}}) \neq r(st_{\mathcal{V}^*}, P, x, \alpha, \rho'_{\mathcal{P}}, \rho'_{\mathcal{V}^*}) \right) \wedge \left(r \neq \bot \right) \right] \right] \\
&\quad < \mathop{\mathbb{E}}_{\rho'_{\mathcal{P}}, \rho'_{\mathcal{V}^*}} \left[3\sqrt{\delta^s_b} \right] \\
&\quad = 3\sqrt{\delta^s_b}.
\end{aligned}$$

□

Remark 6.9. *Note that the Proposition 6.7 remains valid even if we also consider interactions that are outside the support of the honest prover. Formally, let $\mathsf{View}_{\mathcal{V}^*}$ be any distribution over everything in the verifier's view* after *the statistically hiding commitment. Then the proposition holds also for $r = r(st_{\mathcal{V}^*}, \mathsf{View}_{\mathcal{V}^*})$.*

6.2.2 Constructing the Simulator

Using Proposition 6.7 we now construct our simulator $\mathcal{S}_{\mathcal{V}^*}$ for Protocol 6.4. Let $\mathcal{V}^*$ be a verifier with space at most s^s_b. We construct the simulator $\mathcal{S}_{\mathcal{V}^*}$ in Fig. 2.

Note that in order to simulate the protocol, the simulator must have enough space to simulate the malicious verifier, the receiver of the statistically hiding commitment and the sender of the polynomial commitment, and so in total requires space $s_{\mathcal{V}^*} + s^s_{\mathcal{R}} + s^p_{\mathcal{S}}$.

In the following proposition we argue that for any malicious verifier $\mathcal{V}^*$, Simulator 6.10 with access to $\mathcal{V}^*$ satisfies the zero-knowledge definition for Protocol 6.4, and by such we conclude the proof of Theorem 6.2.

Proposition 6.11. *Let n denote the length of a PEP instance (P, x, α) where P is an m-variate polynomial of individual degree d defined over a field $\mathbb{F}$. Let $\mathcal{V}^*$ be a malicious ROBP verifier with space at most $\min\left\{s^s_b, s^p_h\right\}$. Let $\mathcal{S}(P, x, \alpha)$ (resp. $\mathsf{View}^{\mathcal{P}}_{\mathcal{V}^*}(P, x, \alpha)$) be the output of Simulator 6.10 with access to $\mathcal{V}^*$ (resp. the view of $\mathcal{V}^*$ when interacting with the honest prover from Protocol 6.4) on input (P, x, α). Then, for all ROBP streaming distinguishers $\mathcal{D}$ with space at most $s^p_h - s^s_b$ and all inputs $(P, x, \alpha) \in \mathsf{PEP}$,*

$$\left| \Pr\left[\mathcal{D}(\mathcal{S}(P, x, \alpha)) = 1 \right] - \Pr\left[\mathcal{D}(\mathsf{View}^{\mathcal{P}}_{\mathcal{V}^*}(P, x, \alpha)) = 1 \right] \right| \leq 12\sqrt{\delta^s_b} + 4\delta^p_h.$$

Proof. Let $\delta = 12\sqrt{\delta^s_b} + 4\delta^p_h$ and assume toward a contradiction that there exist λ, a distinguisher $\mathcal{D}$, and input $(P, x, \alpha) \in \mathsf{PEP}$ for which

$$\left| \Pr[\mathcal{D}(\mathcal{S}(P, x, \alpha)) = 1] - \Pr[\mathcal{D}(\mathsf{View}^{\mathcal{P}}_{\mathcal{V}^*}(P, x, \alpha)) = 1] \right| > \delta.$$

For convenience, we henceforth denote the distributions $\mathcal{S}(P, x, \alpha)$ and $\mathsf{View}^{\mathcal{P}}_{\mathcal{V}^*}(P, x, \alpha)$ by A and B, respectively. We use $\mathcal{D}$ to construct a distinguisher $\mathcal{D}'$ that breaks the hiding guarantee of

Simulator 6.10. Simulator for PEP.
Streaming Input: An m-variate polynomial P of individual degree d over field $\mathbb{F}$, streamed as a vector of evaluations at $(d+1)^m = n$ points, a point $x \in \mathbb{F}^m$, a value $\alpha \in \mathbb{F}$.

1. **Statistically hiding commitment stage**
   (a) Engage with $\mathcal{V}^*$ in an interactive statistically hiding streaming commitment per Definition 4.1 and write the commitment's transcript on the output tape.
   (b) Let $st_{\mathcal{V}^*}$, $st_{\mathcal{R}}$ be the verifier's and receiver's state after sending the commitment, respectively. Compute the value $r' = r'(st_{\mathcal{V}^*}, st_{\mathcal{R}})$ as per Proposition 6.7.
2. **Streaming stage**
   (a) Stream the input, and forward it to $\mathcal{V}^*$ as it is streamed. While streaming, compute and record the value $P(r')$, and write the input stream on the output tape.
3. **Interactive stage**
   (a) Get $L : \mathbb{F} \to \mathbb{F}^m$ from $\mathcal{V}^*$.
   (b) Send to $\mathcal{V}^*$ a statistically binding streaming polynomial commitment per Definition 5.14 to an arbitrary univariate polynomial g' with the following two constraints: $g'(0) = \alpha$ and $g'(L^{-1}(r')) = P(r')$, represented as evaluations at $dm+1$ points (e.g., the line passing through these two points). Write the commitment's transcript to the output tape.
4. **De-commitment stage**
   (a) Get from $\mathcal{V}^*$ the de-commitment to r.
   (b) If the de-commitment is valid, and $r = r'$, write the de-commitment to $g'(0)$, $g'(L^{-1}(r))$ on the output tape, otherwise abort.

Figure 2: Simulator 6.10: Simulator for Protocol 6.4

the polynomial commitment.

The distinguisher $\mathcal{D}'$ has access to $\mathcal{D}$, the malicious verifier $\mathcal{V}^*$, and the input (P, x, α). Its goal is to win the hiding security game of the polynomial commitment (see Section 5.1.1) against the challenger. Note that according to the rules of the game, we do not need to impose any computational limitations on the distinguisher, with the exception that at a certain point during the streaming of the challenge, it may only retain s_h^p bits from its computations thus far. As a matter of fact, we will be more restrictive and the $\mathcal{D}'$ that we construct does hold more than s_h^p bits in memory from the very beginning of the challenger's stream. Take notice that since the malicious verifier $\mathcal{V}^*$ is limited to use at most s_h^s bits of memory and $\mathcal{D}$ at most $s_h^p - s_h^s$, then $\mathcal{D}'$ can run both $\mathcal{V}^*$ and $\mathcal{D}$ without exceeding its memory limitation.

We proceed to describe the distinguisher. First it needs to select the two polynomials $g_{\mathcal{S}}, g_{\mathcal{P}}$ from which the challenger will select its challenge, and the set of points at which the challenger will provide the de-commitment. In order to do so it starts engaging with $\mathcal{V}^*$ in the statistically hiding commitment to the value r as per Step 1b of the protocol. While doing so it streams the transcript of the interaction to $\mathcal{D}$. Similarly to the simulator in Step 1b, $\mathcal{D}'$ attempts to extract a value r' to which it predicts that $\mathcal{V}^*$ will de-commit later on. After simulating the verifier's commitment, it streams the input (P, x, α) (which it has hardcoded) to both $\mathcal{D}$ and $\mathcal{V}^*$, and continues to receive a value from $\mathcal{V}^*$ representing a line L as per Step 3b of the protocol. Now it constructs $g_{\mathcal{S}}$ and $g_{\mathcal{P}}$, which are the polynomials to be committed by the simulator and the prover, respectively. Recall that $g_{\mathcal{S}}$ is an arbitrary polynomial represented as evaluations at $(dm + 1)$ points that adheres to the constraints $g_{\mathcal{S}}(0) = \alpha$ and $g_{\mathcal{S}}(L^{-1}(r')) = P(r')$, and that $g_{\mathcal{P}} = P|_L$. Note that since the distinguisher has full access to P, it is able to construct $g_{\mathcal{P}}$ (and therefore also $g_{\mathcal{S}}$). Since $P(x) = P(L(0)) = \alpha$, we have that $g_{\mathcal{S}}(0) = g_{\mathcal{P}}(0)$, and so the two polynomials agree on the two evaluation points. To conclude the first step of the game, the distinguisher $\mathcal{D}'$ sends the polynomials $g_{\mathcal{S}}$, $g_{\mathcal{P}}$ and the points $\{0, L^{-1}(r')\}$ to the challenger.

The challenger chooses one of these two polynomials at random and sends a streaming polynomial commitment to the chosen polynomial. While receiving the commitment, $\mathcal{D}'$ forwards the stream to $\mathcal{D}$ and $\mathcal{V}^*$. At this point, $\mathcal{V}^*$ sends the de-commitment to r. If $r \neq r'$ then $\mathcal{D}'$ aborts. Otherwise, it receives the evaluation key from the challenger, forwards it to $\mathcal{D}$ and gets as a response $\mathcal{D}$'s decision bit. If $\mathcal{D}$ claims that the distribution comes from the simulation then $\mathcal{D}'$ outputs $g_{\mathcal{S}}$, otherwise it outputs $g_{\mathcal{P}}$.

Denote the event that $\mathcal{D}'$ correctly predicts the value r by E. Note that if the challenger sends a commitment to $g_{\mathcal{S}}$ then the transcript streamed to $\mathcal{D}$ is exactly a transcript of a simulation, otherwise it is a transcript of the view of the real protocol. Thus,

$$\Big|\Pr[\mathcal{D}' \text{ outputs 1 given } g_{\mathcal{S}}, E] - \Pr[\mathcal{D}' \text{ outputs 1 given } g_{\mathcal{P}}, E]\Big| = \Big|\Pr_{a \leftarrow A}[\mathcal{D}(a) = 1, E] - \Pr_{b \leftarrow B}[\mathcal{D}(b) = 1, E]\Big|.$$

Recall that by Corollary 6.8, $\Pr_{b \leftarrow B}[\neg E] \leq 3\sqrt{\delta_b^s}$. We argue that also when the polynomial commitment is taken from the simulation, the probability of E is not far from when it is taken from the real view.

Claim 6.11.1. $\Pr_{a \leftarrow A}[\neg E] \leq 3\sqrt{\delta_b^s} + \delta_h^p$.

Proof. Assume otherwise, and construct a distinguisher that breaks the polynomial commitment's hiding (that does not include the key reveal). The distinguisher is similar to the one described above, with the exception that when $\mathcal{V}^*$ de-commits to r, the distinguisher outputs 1 if $r' = r$ and

0 otherwise. According to the assumption, the distinguisher wins the game with probability greater than δ_h^p, contradicting the hiding property of the polynomial commitment. □

To tie it all together we use the following simple claim regarding probability distributions.

Claim 6.11.2. *Let X, Y be distributions defined over the same distribution space Ω. Let $E_1, E_2 \subseteq \Omega$ be events. Then,*

$$\Big| \big| \underset{X}{\Pr}[E_1] - \underset{Y}{\Pr}[E_1] \big| - \big| \underset{X}{\Pr}[E_1, E_2] - \underset{Y}{\Pr}[E_1, E_2] \big| \Big| \leq \underset{X}{\Pr}[\neg E_2] + \underset{Y}{\Pr}[\neg E_2].$$

Proof. From the law of total probability and the triangle inequality we have that

$$\begin{aligned}
\Big| \underset{X}{\Pr}[E_1] - \underset{Y}{\Pr}[E_1] \Big| &= \Big| \underset{X}{\Pr}[E_1, E_2] + \underset{X}{\Pr}[E_1, \neg E_2] - \underset{Y}{\Pr}[E_1, E_2] - \underset{Y}{\Pr}[E_1, \neg E_2] \Big| \\
&\leq \Big| \underset{X}{\Pr}[E_1, E_2] - \underset{Y}{\Pr}[E_1, E_2] \Big| + \underset{X}{\Pr}[E_1, \neg E_2] + \underset{Y}{\Pr}[E_1, \neg E_2] \\
&\leq \Big| \underset{X}{\Pr}[E_1, E_2] - \underset{Y}{\Pr}[E_1, E_2] \Big| + \underset{X}{\Pr}[\neg E_2] + \underset{Y}{\Pr}[\neg E_2].
\end{aligned}$$

Likewise, from the reverse triangle inequality we have that.

$$\begin{aligned}
\Big| \underset{X}{\Pr}[E_1] - \underset{Y}{\Pr}[E_1] \Big| &= \Big| \underset{X}{\Pr}[E_1, E_2] + \underset{X}{\Pr}[E_1, \neg E_2] - \underset{Y}{\Pr}[E_1, E_2] - \underset{Y}{\Pr}[E_1, \neg E_2] \Big| \\
&\geq \Big| \underset{X}{\Pr}[E_1, E_2] - \underset{Y}{\Pr}[E_1, E_2] \Big| - \Big| \underset{X}{\Pr}[E_1, \neg E_2] - \underset{Y}{\Pr}[E_1, \neg E_2] \Big| \\
&\geq \Big| \underset{X}{\Pr}[E_1, E_2] - \underset{Y}{\Pr}[E_1, E_2] \Big| - \underset{X}{\Pr}[\neg E_2] - \underset{Y}{\Pr}[\neg E_2].
\end{aligned}$$

The claim follows. □

Using Claims 6.11.1 and 6.11.2 we have that

$$\begin{aligned}
&\Big| \Pr[\mathcal{D}' \text{ outputs 1 given } g_{\mathcal{S}}] - \Pr[\mathcal{D}' \text{ outputs 1 given } g_{\mathcal{P}}] \Big| \\
&\quad \geq \Big| \Pr[\mathcal{D}' \text{ outputs 1 given } g_{\mathcal{S}}, E] - \Pr[\mathcal{D}' \text{ outputs 1 given } g_{\mathcal{P}}, E] \Big| - \underset{A}{\Pr}[\neg E] - \underset{B}{\Pr}[\neg E] \\
&\quad = \Big| \underset{a \leftarrow A}{\Pr}[\mathcal{D}(a) = 1, E] - \underset{b \leftarrow B}{\Pr}[\mathcal{D}(b) = 1, E] \Big| - \underset{A}{\Pr}[\neg E] - \underset{B}{\Pr}[\neg E] \\
&\quad \geq \Big| \underset{a \leftarrow A}{\Pr}[\mathcal{D}(a) = 1] - \underset{b \leftarrow B}{\Pr}[\mathcal{D}(b) = 1] \Big| - 2 \underset{A}{\Pr}[\neg E] - 2 \underset{B}{\Pr}[\neg E] \\
&\quad \geq \delta - 2 \cdot \left(3\sqrt{\delta_b^s} + \delta_h^p \right) - 2 \cdot 3\sqrt{\delta_b^s} \\
&\quad = 2\delta_h^p.
\end{aligned}$$

Thus, we have that $\mathcal{D}'$ violates the hiding property of the polynomial commitment.

□

7 Streaming Zero-Knowledge Proofs for Low-Depth Circuits

In this section we construct a zkSIP for any NP relation $\mathcal{R}$ that is decidable by a log-space uniform, polynomial size, polylogarithmic depth family of circuits. We do so by constructing a generic compiler, that takes a perfect zero-knowledge proof in the IPCP model for $\mathcal{R}$, along with a statistically hiding streaming commitment and a statistically binding streaming commitment, and turns them into a zkSIP for $\mathcal{R}$.

We combine our compiler to (an extension of) a zero-knowledge IPCP construction for low-depth relations, given by [CFS17], to get the following result:

Theorem 7.1. *Let n denote the input length and let $\lambda = \lambda(n), 0 < \epsilon < 1$ be security parameters. Let $\mathcal{R}$ be an NP relation and let $D(n), S(n)$ be the depth and size functions, respectively, of the circuit family that determines $\mathcal{R}$. Let $\mathcal{L}$ be the language defined by $\mathcal{R}$. Then, there exists a streaming interactive proof for $\mathcal{L}$ with perfect completeness, soundness error $O(D(n) \cdot \mathrm{polylog}(S(n))) \cdot 2^{-\Omega(\lambda^{\epsilon})}$, honest verifier space $O\big(D(n) \cdot \mathrm{polylog}(S(n)) \cdot \lambda + \lambda^{1+\epsilon}\big)$, and communication complexity $O(D(n) \cdot \mathrm{poly}(S(n)) \cdot \lambda^2)$. The proof is zeroknowledge per Definition 3.6 against adversaries with space at most $\frac{\lambda^2}{21}$ with zero-knowledge error $O(D(n) \cdot \mathrm{poly}(S(n))) \cdot 2^{-\Omega(\lambda^{\epsilon})}$.*

Section Organization. In Section 7.1 we lay out the definitions related to the IPCP model and zero-knowledge that we will use in the following subsections. Our compiler turning a IPCP to a streaming interactive proof is constructed in Section 7.2, and in Section 7.3 we show that it preserves zero-knowledge. Finally, in Section 7.4 we prove Theorem 7.1.

7.1 Zero-Knowledge IPCP

The interactive PCP (IPCP) model, introduced by Kalai and Raz [KR08] is an amalgam of the PCP model and the Interactive Proof model. A prover begins the protocol by sending a PCP to which the verifier has query access, followed by a message exchange between the two parties, at the end of which the verifier decides whether to accept or reject. Following work in the PCP and IOP literature we consider a variant of this model, called Holographic IPCP, in which the verifier has oracle access to the low degree extension of the input.

Definition 7.2 (Holographic IPCP)**.** *A* Holographic Interactive Probabilistically Checkable Proof (Holographic IPCP) *for a language $\mathcal{L}$ is a public-coin interactive protocol between an unbounded prover $\mathcal{P}$ and a PPT verifier $\mathcal{V}$, both of which are given oracle access to a multilinear extension $\hat{x}$ of an input x, defined over a field $\mathbb{F}$. The protocol proceeds as follows. The prover $\mathcal{P}$ sends a PCP π, to which the verifier has oracle access. This is followed by a k-round exchange of messages between the parties, where $\mathcal{V}$'s messages are its random coins $r = \{r_i\}_{i=1}^k$ and $\mathcal{P}$'s messages $\beta = \{\beta_i\}_{i=1}^k$ are read in full by $\mathcal{V}$. At the end of the interaction, the verifier makes queries to $\hat{x}$, π, and decides whether to accept or reject according to the answers to its queries, r and β. The queries to $\hat{x}$ and π are non-adaptive, i.e. they depend only on $\mathcal{V}$'s randomness, and not on the answers of previous queries, or the messages of $\mathcal{P}$.*

The protocol satisfies the following conditions:

1. ***Completeness:*** *for every $x \in \mathcal{L}$ the verifier accepts with probability* 1*.*
2. *ϵ-**Soundness:*** *for every $x \notin \mathcal{L}$ and every prover $\mathcal{P}^*$ the verifier accepts with probability at most ϵ.*

Denote the input length as $|x| = n$. Denote the PCP length as $|\pi| = \ell = \ell(n)$. Denote the communication complexity, which includes only the messages sent by the prover during the interactive stage (not including the PCP), by $cc = cc(n)$ (i.e. $cc = |\beta|$). Denote the randomness complexity of the verifier as $rc = |r|$, where $rc = rc(n)$. Denote the number of verifier queries to the input as q_x and the number of verifier queries to the PCP as q_π.

From this point forward, whenever we refer to an IPCP, we specifically mean a Holographic IPCP.

Remark 7.3. *Note that since the IPCP protocol is public-coin and uses a non-adaptive verifier, henceforth we assume without loss of generality that all honest verifier queries to the input occur at the very beginning of the protocol, before the interaction, and all the queries to the PCP occur at the very end of the protocol, after the interaction. This assumption will help us later on, when we compile IPCP protocols to streaming protocols, as this better fits the structure of the streaming protocol, where the input is streamed prior to the interaction, and the answers to the PCP queries are provided by the prover as the last step of the protocol.*

The definition of zero-knowledge that we use for the IPCP model is *perfect zero-knowledge via straightline simulators*. Intuitively, this means that everything that any verifier $\mathcal{V}^*$ sees during the interaction with the prover can be perfectly simulated by a computationally bounded simulator, that has *straightline access* to $\mathcal{V}^*$ as defined next.

Definition 7.4. *An algorithm B has* straightline *access to another algorithm A if B interacts with A, without rewinding, by exchanging messages with A and also answering any oracle queries along the way. We denote by $B_A(x)$ the output of algorithm B with straightline access to A on input x, which includes the randomness of A.*

Our goal is to construct a compiler from the IPCP model to the streaming model, where we are mainly concerned with the space constraints of the simulator rather than its time constraints. Thus, even if the verifier requires a super-polynomial amount of randomness, we can still consider simulators that provide the randomness for the verifier. In practice, whenever the verifier reads a bit from its randomness tape, it is provided by the simulator and simultaneously written on the simulator's output tape.

For our purposes, the importance of using a simulator with straightline access to the verifier lies in the fact that the simulator cannot rewind the verifier, as commonly done in constructions of zero-knowledge proofs. This is because in the streaming model, the input is only streamed *once*, and so rewinding the verifier to a state in which it was prior to the input stream, after the input has finished streaming would make the simulation task unfeasible (or at least much more challenging).

We proceed to present the definition of zero-knowledge in the IPCP model following [CFS17].

Definition 7.5. *An IPCP system $\langle \mathcal{P}, \mathcal{V} \rangle$ for a language $\mathcal{L}$ is* perfect zero-knowledge *(via straightline simulators) against query bound b with simulator overhead $s : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ if there exists a simulator algorithm $\mathcal{S}$ such that for every b-query algorithm $\mathcal{V}^*$ and input $x \in \mathcal{L}$, the output of the simulator $\mathcal{S}_{\mathcal{V}^*}(x)$ and $\mathrm{View}^{\mathcal{P}}_{\mathcal{V}^*}(x)$ are identically distributed. Moreover, $\mathcal{S}$ must run in time $O\big(s(n, q_{\mathcal{V}^*}(n))\big)$, where $q_{\mathcal{V}^*}(n)$ is $\mathcal{V}^*$'s query complexity.*

Actually, a more relaxed notion of zero-knowledge in the IPCP model, which we call *semi-malicious verifier zero-knowledge*, will suffice for us. A semi-malicious verifier is a verifier that selects its random bits from any sequence that is in the support of the honest verifier.

Definition 7.6. *An IPCP system $\langle \mathcal{P}, \mathcal{V} \rangle$ for a language $\mathcal{L}$ is* semi-malicious verifier perfect zero-knowledge *(via straightline simulators) with simulator overhead $s : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ if there exists a simulator algorithm $\mathcal{S}$ such that for every semi-malicious verifier $\mathcal{V}^*$ and input $x \in \mathcal{L}$, the output of the straightline simulator $\mathcal{S}_{\mathcal{V}^*}(x)$ and $\mathrm{View}^{\mathcal{P}}_{\mathcal{V}^*}(x)$ are identically distributed. Moreover, $\mathcal{S}$ must run in time $O\big(s(n, q_{\mathcal{V}}(n))\big)$, where $q_{\mathcal{V}}(n)$ is $\mathcal{V}$'s query complexity.*

Clearly, as long as the query bound b in Definition 7.5 is not less than the honest verifier's query complexity, then a protocol that satisfies Definition 7.5 also satisfies Definition 7.6.

7.2 From IPCP to Streaming

We describe a general transformation from proofs in the IPCP model, to proofs in the streaming model, where if the original IPCP is zero-knowledge in the IPCP setting (as per Definition 7.5) then the resulting streaming proof is a streaming zero-knowledge proof (as per Definition 3.6).

Similarly to our construction in Section 6, here too we construct our streaming protocol using streaming commitments. Thus, throughout this section we let $\big(s^h_{\mathcal{S}}, s^h_{\mathcal{R}}, s^h_b, \delta^h_h, \delta^h_b\big)$ be the parameters of the statistically hiding commitment (as per Definition 4.1), and let $\big(s^b_{\mathcal{S}}, s^b_{\mathcal{R}}, s^b_h, \delta^b_h, \delta^b_b\big)$ be the parameters of the statistically binding commitment (as per Definition 5.2).[15] Denote by cc^h, cc^b the communication complexities of the statistically hiding and statistically binding commitments, respectively. We consider all the parameters above to be functions of the input length.

The compiler from the IPCP model to the streaming model is presented in Fig. 3.

First, in Lemma 7.8, we demonstrate that the protocol is a valid interactive streaming protocol, satisfying completeness, soundness, and the specified complexity. Then, in Section 7.3, we prove that it is also zero-knowledge in the streaming model.

Lemma 7.8. *Suppose $\mathcal{L}$ has a k-round IPCP protocol $\langle \mathcal{P}_{\mathrm{IPCP}}, \mathcal{V}_{\mathrm{IPCP}} \rangle$ over a field $\mathbb{F}$ with soundness error δ, PCP length $|\pi| = \ell \cdot \log(|\mathbb{F}|)$ and communication complexity cc, and an IPCP verifier with space s that uses rc random bits and makes q_x queries to the input and q_π queries to the PCP.*

Let $\langle \mathcal{P}_{\mathrm{S}}, \mathcal{V}_{\mathrm{S}} \rangle$ be the output of Compiler 7.7 on $\langle \mathcal{P}_{\mathrm{IPCP}}, \mathcal{V}_{\mathrm{IPCP}} \rangle$ and commitments per Definition 4.1 and Definition 5.2. Denote the space of $\mathcal{V}_{\mathrm{S}}$ by $s_{\mathcal{V}}$. Let $\big(s^h_{\mathcal{S}}, s^h_{\mathcal{R}}, s^h_b, \delta^h_b, \delta^h_h\big)$ (resp. $\big(s^b_{\mathcal{S}}, s^b_{\mathcal{R}}, s^b_h, \delta^b_b, \delta^b_h\big)$) be the parameters of a statistically hiding streaming commitment (resp. statistically hiding streaming commitment), such that $s^h_{\mathcal{R}} = O(s_{\mathcal{V}}), s^b_{\mathcal{S}} = O(s_{\mathcal{V}}), s^h_b > s_{\mathcal{V}}$, and $s^p_h > s^s_b + k \cdot s^h_{\mathcal{R}} + q_\pi \cdot s^b_{\mathcal{S}}$.

Then, $\langle \mathcal{P}_{\mathrm{S}}, \mathcal{V}_{\mathrm{S}} \rangle$ is a streaming interactive proof for $\mathcal{L}$ with perfect completeness, soundness error $\delta + 3k \cdot \delta^h_h + q_\pi \cdot \delta^b_b$, ROTM verifier space $s_{\mathcal{V}} = s + q_\pi \cdot s^b_{\mathcal{R}} + k \cdot s^h_{\mathcal{S}} + q_x \cdot \lceil \log(|\mathbb{F}|) \rceil$, and communication complexity $cc + k \cdot cc^h + \ell \cdot cc^b$.

Proof. We shall first account for the amount of space that $\mathcal{V}_{\mathrm{S}}$ needs in order to execute the protocol.

For the statistically hiding commitments in Step 1b the verifier requires $k \cdot s^h_{\mathcal{S}}$ storage bits. To store the input values computed in Step 2 the verifier requires $q_x \cdot \log(|\mathbb{F}|)$ storage bits. To execute the statistically binding commitment of all the PCP elements, in Step 3 the verifier needs to store values related only to its query locations, and so requires $q_\pi \cdot s^b_{\mathcal{R}}$ space. To simulate $\mathcal{V}_{\mathrm{PCP}}$, the verifier $\mathcal{V}_{\mathrm{S}}$ requires s memory bits. Thus, in total, $\mathcal{V}_{\mathrm{S}}$ requires $s + q_\pi \cdot s^b_{\mathcal{R}} + k \cdot s^h_{\mathcal{S}} + q_x \cdot \log(|\mathbb{F}|)$ memory bits.

[15] Note that the h and b superscripts stand for "statistically hiding commitment" and "statistically binding commitment", respectively.

Compiler 7.7. Compiler from an IPCP protocol to a streaming interactive protocol for a language $\mathcal{L}$.
Ingredients: A k-round IPCP protocol $\langle \mathcal{P}_{\text{IPCP}}, \mathcal{V}_{\text{IPCP}} \rangle$, a statistically hiding streaming commitment $\langle \mathcal{S}_s, \mathcal{R}_s \rangle$, and a statistically binding commitment $\langle \mathcal{S}_p, \mathcal{R}_p \rangle$.
Output: Streaming interactive protocol $\langle \mathcal{P}_{\text{S}}, \mathcal{V}_{\text{S}} \rangle$.
We describe the output protocol on a given streamed input x.

1. **Statistically hiding commitment stage**
   (a) $\mathcal{V}_{\text{S}}$ draws the randomness $r = \{r_i\}_{i=1}^{k}$ for $\mathcal{V}_{\text{IPCP}}$ and saves the locations where it will query the input (since $\mathcal{V}_{\text{IPCP}}$ is non-adaptive these locations depend only on r).
   (b) For all $i \in [k]$, the verifier $\mathcal{V}_{\text{S}}$ sends to $\mathcal{P}_{\text{S}}$ a commitment to r_i using the interactive statistically hiding streaming string commitment per Definition 4.1.
2. **Streaming stage**
   (a) The prover and verifier stream the input. While streaming, $\mathcal{V}_{\text{S}}$ computes the values of the multilinear extension of the input at all points to be queried by $\mathcal{V}_{\text{IPCP}}$. The prover, who has unbounded space, records the entirety of the input.
3. **PCP stage**
   (a) Using the statistically binding streaming commitment (see Definition 5.2), the prover $\mathcal{P}_{\text{S}}$ sends to the $\mathcal{V}_{\text{S}}$ a commitment to each element of the PCP π generated by $\mathcal{P}_{\text{IPCP}}$. Note that each of the PCP bits are committed *separately*. The verifier $\mathcal{V}_{\text{S}}$, which already knows the points where $\mathcal{V}_{\text{IPCP}}$ will query (again, using the fact that it is non-adaptive), only stores the relevant commitment information for locations to which it wants to query, and discards the rest.
4. **Interactive stage**
   (a) $\mathcal{V}_{\text{S}}$ and $\mathcal{P}_{\text{S}}$ simulate the IPCP interactive stage using $\mathcal{V}_{\text{IPCP}}$ and $\mathcal{P}_{\text{IPCP}}$. For every round $i \in [k]$, when $\mathcal{V}_{\text{IPCP}}$ sends r_i, it is simulated by $\mathcal{V}_{\text{S}}$ by de-committing to r_i to which it committed in Step 1b. $\mathcal{P}_{\text{S}}$ responds with the message β_i of $\mathcal{P}_{\text{IPCP}}$ in the clear.
5. **Query and decision stage**
   (a) The prover de-commits to all the values of the PCP that will be queried by $\mathcal{V}_{\text{IPCP}}$.[a]
   (b) $\mathcal{V}_{\text{S}}$ outputs the decision of $\mathcal{V}_{\text{IPCP}}$, computed as follows.
      i. When $\mathcal{V}_{\text{IPCP}}$ queries π, the value is provided from the prover's de-commitment in the previous step.
      ii. When $\mathcal{V}_{\text{IPCP}}$ queries $\hat{x}$, the value is provided by $\mathcal{V}_{\text{S}}$, which is available since it was saved in Step 2.

[a] The query locations are determined by r, which at this point is fully available to the prover.

Figure 3: Compiler 7.7: From IPCP to Streaming

The communication complexity of the streaming protocol is the communication complexity of the original protocol, with the addition of the communication of the two commitments. Thus, in total the communication complexity is $cc + k \cdot cc^h + q_\pi \cdot cc^b$.

Completeness follows immediately from the streaming protocol's description together with the correctness of the commitments and the completeness of the IPCP protocol.

Finally, we argue that $\langle \mathcal{P}_\mathrm{S}, \mathcal{V}_\mathrm{S} \rangle$ has a soundness error of $\delta + 3k \cdot \delta_h^h + q_\pi \cdot \delta_b^b$. Let $x \notin \mathcal{L}$ and fix a malicious prover $\mathcal{P}^*$. By the binding condition of the statistically binding commitment used by the prover in Step 3, the commitments uniquely define a PCP, which we denote by π^*. Let I be the verifier's set of PCP queries, and denote by E the event that the prover de-commits to $\tilde{\pi}|_I \neq \pi^*|_I$. By the statistical binding property of the statistically binding commitment, and applying the union bound on all the queries, we have that

$$\Pr[E] \leq q_\pi \cdot \delta_b^b.$$

Now consider the following slightly altered protocol using an unbounded streaming verifier. In Step 1a the streaming verifier picks the randomness r for the IPCP verifier, as per the original streaming protocol. In Step 1b the verifier sends to the prover k commitments to *zeros*. Then, in Step 4, during the interactive stage, in every round i, the streaming verifier, using its unbounded powers, de-commits to r_i. The rest of the protocol is similar to the original.

Denote by E_i the event that the *unbounded* verifier fails to de-commit to r_i after committing to zero for all $i \in [k]$.

Claim 7.8.1. *For all $i \in [k]$,*

$$\Pr[E_i] \leq 2\delta_h^h.$$

Proof. Assume that there exists $i \in [k]$ for which $\Pr[E_i] > 2\delta_h^h$. Recall that in this setting both the verifier and the prover, who act as the sender and receiver, respectively, are unbounded, and in particular, have similar computational capabilities. Thus, if the verifier fails to de-commit to r_i, it means that there does not exist a key with which the verifier can de-commit, and since the prover is at least as strong as the verifier, it knows that, and so it knows that the committed message is not r_i.

Therefore, we have an unbounded distinguisher $\mathcal{D}$ that can distinguish between a commitment to zeros and a commitment to r_i. On a commitment *com*, check if there exists a key that opens *com* to r_i. If not, output 0, and if such a key exists, output either 0 or r_i, each with probability $\frac{1}{2}$. Note that in the case where the commitment is to r_i, there must exist a key that opens the commitment to r_i. We have that

$$\Big| \Pr[\mathcal{D}(com_{r_i}) = 0] - \Pr[\mathcal{D}(com_0) = 0] \Big| = \Big| \tfrac{1}{2} - \big(\Pr[E_i] + \tfrac{1}{2} \cdot \Pr[\neg E_i] \big) \Big| = \tfrac{1}{2} \cdot \Pr[E_i] > \delta_h^h,$$

in contradiction to the commitment's hiding property. □

Note that in the altered protocol, the prover is completely oblivious to the randomness of the IPCP verifier (up to the point where each r_i is revealed, in accordance with the IPCP protocol), and in particular this is true at the time when the prover commits to π^*. Thus, if we also assume that the event E does not occur, it holds that from the perspective of the inner IPCP verifier, it is a perfect recreation of the IPCP setting. Note that even if any event E_i does occur, while it may affect the behavior of the prover, from the perspective of the inner IPCP verifier, it is still a perfect recreation if the IPCP setting. Therefore, by the soundness guarantee of the IPCP protocol, we have that in the altered protocol the IPCP verifier (and by such also the streaming verifier) accepts with probability at most δ.

Let $j \in \{0, 1, \ldots, k\}$, and consider a hybrid protocol such that in the j-th hybrid, in Step 1b, for all $i \in [j]$, the streaming verifier commits to r_i, and the rest of the $k - j$ commitments are to zero vectors. Denote by H_j the event that the verifier accepts in the j-th hybrid. Note that H_0 and H_k are the events that the verifier accepts in the altered protocol and the real protocol, respectively. According to our conclusion from above, we have that

$$\Pr[H_0] \leq \Pr[H_0, \neg E] + \Pr[E] \leq \delta + q_\pi \cdot \delta_b^b.$$

Assume towards a contradiction that $\Pr[H_k] > \delta + 3k \cdot \delta_h^h + q_\pi \cdot \delta_b^b$. If so, we have that $\big|\Pr[H_k] - \Pr[H_0]\big| > 2k \cdot \delta_h^h$, and by such there exists $j \in [k]$ such that $\big|\Pr[H_j] - \Pr[H_{j-1}]\big| > 3\delta_h^h$. By the hiding property of the statistically hiding commitment, we have that $\big|\Pr[H_j, \neg E_j] - \Pr[H_{j-1}, \neg E_j]\big| \leq \delta_h^h$. Note that in the j-th hybrid, the j-th commitment is to the real r_j and so $\Pr[H_j, \neg E_j] = \Pr[H_j]$. Thus, using the triangle inequality and Claim 7.8.1, we have the following contradiction

$$3\delta_h^h < \big|\Pr[H_j] - \Pr[H_{j-1}]\big| \leq \big|\Pr[H_j] - \Pr[H_{j-1}, \neg E_j]\big| + \Pr[E_j] \leq 3\delta_h^h.$$

□

7.3 Compiler 7.7 Preserves Zero-Knowledge

In this section we prove the following lemma.

Lemma 7.9. *Let $\langle \mathcal{P}_{\text{IPCP}}, \mathcal{V}_{\text{IPCP}} \rangle$ be a IPCP protocol for $\mathcal{L}$ that is zero-knowledge per Definition 7.5, let $\langle \mathcal{S}_h, \mathcal{R}_h \rangle$ be a statistically hiding streaming commitment per Definition 4.1, let $\langle \mathcal{S}_b, \mathcal{R}_b \rangle$ be a statistically binding commitment per Definition 5.2, and let $\langle \mathcal{P}_{\text{S}}, \mathcal{V}_{\text{S}} \rangle$, be the output of Compiler 7.7 on ingredients $\langle \mathcal{P}_{\text{IPCP}}, \mathcal{V}_{\text{IPCP}} \rangle$, $\langle \mathcal{S}_s, \mathcal{R}_s \rangle$, $\langle \mathcal{S}_p, \mathcal{R}_p \rangle$. Then, $\langle \mathcal{P}_{\text{S}}, \mathcal{V}_{\text{S}} \rangle$ is a streaming interactive proof for $\mathcal{L}$ that is zero-knowledge per Definition 3.6.*

The general idea for proving the lemma is to construct a streaming simulator that simulates the malicious streaming verifier's view, using the IPCP simulator that we get from the IPCP zero-knowledge protocol.

As a matter of fact, it suffices for us to use an IPCP that is *semi-maliciously* secure, even though the streaming verifier can be arbitrarily malicious. The reason for this is that the malicious streaming verifier begins by committing to the random coins for the underlying IPCP verifier. By doing so, the only influence it can exert corresponds to semi-malicious behavior – it can arbitrarily choose the honest verifier's coins but cannot deviate beyond that.

For these reasons we rewrite Lemma 7.9 as the following equivalent lemma.

Lemma 7.10. *Let $\langle \mathcal{P}_{\text{IPCP}}, \mathcal{V}_{\text{IPCP}} \rangle$ be a IPCP protocol for $\mathcal{L}$ that is semi-malicious verifier zero-knowledge per Definition 7.6, let $\langle \mathcal{S}_h, \mathcal{R}_h \rangle$ be a statistically hiding streaming commitment per Definition 4.1, let $\langle \mathcal{S}_b, \mathcal{R}_b \rangle$ be a statistically binding commitment per Definition 5.2, and let $\langle \mathcal{P}_{\text{S}}, \mathcal{V}_{\text{S}} \rangle$, be the output of Compiler 7.7 on input $\langle \mathcal{P}_{\text{IPCP}}, \mathcal{V}_{\text{IPCP}} \rangle$. Then, $\langle \mathcal{P}_{\text{S}}, \mathcal{V}_{\text{S}} \rangle$ is a streaming interactive proof for $\mathcal{L}$ that is zero-knowledge per Definition 3.6.*

To prove Lemma 7.10 we need to show that the streaming verifier's view during an interaction with the prover is indistinguishable from the output of a streaming simulator, and so we first specify the view of the streaming verifier. To do so we enumerate the components that the verifier sees in the protocol, in the order that they are seen:

1. Statistically binding streaming commitment to the randomness r.
2. Input x.
3. Polynomial commitment to the PCP π.
4. The messages of $\mathcal{P}_{\text{IPCP}}$.
5. De-commitments to $\{\pi_q\}_{q\in X}$ where X is the set of the PCP query locations determined by r.

We recall that in our model, every time the verifier uses a random bit, it is written as the next bit in the view's transcript. Thus, all the verifier's messages are a deterministic function of all that preceded them in the view and need not be incorporated into the view.

We shall now construct a small-space simulator $\mathcal{S}_{\text{S}}$ that simulates the view of a malicious verifier $\mathcal{V}^*$. Let $\langle \mathcal{P}_{\text{IPCP}}, \mathcal{V}_{\text{IPCP}}\rangle$ be a zero-knowledge IPCP protocol for $\mathcal{L}$. Thus, there exists a PPT simulator $\mathcal{S}_{\text{IPCP}}$ that perfectly simulates the view of any semi-malicious IPCP verifier $\mathcal{V}^*_{\text{IPCP}}$, when it is given straightline black box access to $\mathcal{V}^*_{\text{IPCP}}$.

We present the following small-width branching-program simulator $\mathcal{S}_{\text{S}}$ with access to a streaming verifier $\mathcal{V}^*_{\text{S}}$ in Fig. 4.

Let $s_{\mathcal{S}}$ be the amount of space that $\mathcal{S}_{\text{IPCP}}$ requires in order to run, and $s_{\mathcal{V}^*}$ the amount of space that $\mathcal{V}^*_{\text{S}}$ requires to run. From its description, we see that Simulator 7.11 requires enough space to run the statistically hiding streaming commitment k times, $\mathcal{S}_{comm}$, $\mathcal{S}_{\text{IPCP}}$, and $\mathcal{V}^*_{\text{S}}$. This requires a total space of

$$s_{\mathcal{S}} + s_{\mathcal{V}^*} + k \cdot s^h_{\mathcal{R}} + q_\pi \cdot s^b_{\mathcal{S}}.$$

We shall now argue that the output of Simulator 7.11 is indistinguishable from the streaming verifier's view when interacting with the streaming prover.

Proposition 7.12. *Let n denote the length of the input x. Let $\mathcal{V}^*$ be a malicious streaming verifier with space at most $s_{\mathcal{V}^*} = \min\left\{s^h_b, s^b_h\right\}$. Let $\mathcal{S}(x)$ (resp. $\mathsf{View}^{\mathcal{P}}_{\mathcal{V}^*}(x)$) be the output of Simulator 7.11 with access to $\mathcal{V}^*$ (resp. the view of $\mathcal{V}^*$ when interacting with the honest prover from the output of Compiler 7.7) on input x. Then, for all ROBP distinguishers $\mathcal{D}$ with space at most s^b_h and all inputs $x \in \mathcal{L}$,*

$$\left|\Pr\left[\mathcal{D}(\mathcal{S}(x)) = 1\right] - \Pr\left[\mathcal{D}(\mathsf{View}^{\mathcal{P}}_{\mathcal{V}^*}(x)) = 1\right]\right| \leq 9k\sqrt{\delta^h_b} + \ell \cdot \delta^b_h.$$

Proof. Consider $\mathcal{S}(x)$ (resp. $\mathsf{View}^{\mathcal{P}}_{\mathcal{V}^*}(x)$) as a joint distribution A, B (resp. (A', B')), where B (resp. B') is the distribution of the commitments to the PCP values that are *not* opened at the end of the protocol, and A (resp. A') is the distribution of everything else in the simulator's transcript (resp. the verifier's view).

Let $\widetilde{\mathcal{S}(x)}$ be a joint distribution $(\widetilde{A}, \widetilde{B})$, such that the marginal distributions $\widetilde{A}$ and A are identically distributed. Define $\widetilde{B}$ as follows. The distribution $\widetilde{B}$ is supported on $\text{Supp}(B') \cup \{\perp\}$ for all $\omega_A \in \text{Supp}(A')$, $\omega_B \in \text{Supp}(B')$, define

$$\Pr_{(a,b)\leftarrow(\widetilde{A},\widetilde{B})}[b = \omega_B | a = \omega_A] = \Pr_{(a,b)\leftarrow(A',B')}[b = \omega_B | a = \omega_A].$$

Simulator 7.11. Simulator for Compiler 7.7, where $\langle \mathcal{P}_{\mathrm{IPCP}}, \mathcal{V}_{\mathrm{IPCP}} \rangle$ is an IPCP over a field $\mathbb{F}$ that is semi-malicious verifier zero-knowledge with simulator $\mathcal{S}_{\mathrm{IPCP}}$. The simulator has the code of $\mathcal{V}^*$ as a fixed auxiliary input.
Streaming Input: $x \in \mathcal{L}$.

1. **Statistically hiding commitment stage**
   (a) Engage with $\mathcal{V}^*$ in k interactive statistically hiding streaming commitments as in Definition 4.1 and write the commitments' transcript on the output tape.
   (b) For all $i \in [k]$, let $st^i_{\mathcal{V}^*}$, $st^i_{\mathcal{R}}$ be the verifier's and receiver's state after sending the i-th commitment, respectively. Compute the value $r'_i = r'(st^i_{\mathcal{V}^*}, st^i_{\mathcal{R}})$ using Proposition 6.7.
   (c) Determine the set I_{pcp} of locations where the IPCP verifier will query the PCP, as well as the set I_{inp} of points where the verifier will query the low-degree extension of the input. If $r'_i = \perp$, then set it to some arbitrary value.
2. **Streaming stage**
   (a) Stream the input, forward it to $\mathcal{V}^*_{\mathrm{S}}$ and compute and record all the values at the points in set I_{inp}. Recall that a value of the low-degree extension of the input can be computed in small space in a streaming manner, using online Lagrange interpolation.
   (b) As the input is being streamed, copy it to the output tape.
3. **IPCP simulation stage**
   (a) Run $\mathcal{S}_{\mathrm{IPCP}}$. Whenever $\mathcal{S}_{\mathrm{IPCP}}$ expects the i-th message from $\mathcal{V}^*_{\mathrm{IPCP}}$, feed it with r'_i. Whenever $\mathcal{S}_{\mathrm{IPCP}}$ queries the input, provide it with the appropriate value stored in the previous step.
   (b) Store the simulated output $(\{\beta_i\}_{i=1}^k, \{\pi_q\}_{q \in I_{pcp}})$ of $\mathcal{S}_{\mathrm{IPCP}}$ where β_i are the simulated prover messages and π_q are the answers to the PCP queries.
4. **PCP commitment stage**
   (a) Generate an arbitrary PCP π that agrees with all values $\{\pi_q\}_{q \in I_{pcp}}$ in the output of $\mathcal{S}_{\mathrm{IPCP}}$ from Step 3). Concretely, we fix the values of the PCP outside of I_{pcp} to be zero.
   (b) Using that statistically binding commitment (see Definition 5.2) commit to the PCP π, where each bit of π is committed separately.
5. **Interactive stage** For $i \in [k]$ rounds:
   (a) Get a de-commitment from $\mathcal{V}^*_{\mathrm{S}}$ to r_i. If the verifier fails to de-commit ($r_i = \perp$), or if $r'_i \neq r_i$, abort the simulation.
   (b) Write β_i on the output tape.
6. **De-commitment stage**
   (a) De-commit to all the PCP values in the query set I_{pcp}.
   (b) Write the de-commitment on the output tape.

Figure 4: Simulator 7.11: Simulator for Output of Compiler 7.7

Also define for all $\omega_A \in \mathrm{Supp}(A')$

$$\Pr_{(a,b)\leftarrow(\widetilde{A},\widetilde{B})}[b=\perp | a=\omega_A] = 0,$$

and for all $\omega_A \in \big(\mathrm{Supp}(\widetilde{A}) \setminus \mathrm{Supp}(A')\big)$, define

$$\Pr_{(a,b)\leftarrow(\widetilde{A},\widetilde{B})}[b=\perp | a=\omega_A] = 1.$$

Thus, we get that for all $\omega_b \in \mathrm{Supp}(\widetilde{B})$,

$$\Pr_{b\leftarrow\widetilde{B}}[b=\omega_B] = \sum_{\omega_a\in\mathrm{Supp}(A')} \Pr_{(a,b)\leftarrow(A',B')}[b=\omega_B|a=\omega_A]\cdot \Pr_{a\leftarrow\widetilde{A}}[a=\omega_A] + \sum_{\omega_A\in\mathrm{Supp}(\widetilde{A})\setminus\mathrm{Supp}(A')} \Pr_{\widetilde{A}}[a=\omega_A]\cdot I_{\omega_B=\perp},$$

where $I_{\omega_B=\perp}$ is an indicator that equals 1 if $\omega_b=\perp$, else 0.

By Lemma 5.7, we get that for all streaming distinguishers $\mathcal{D}$ with space at most s_h^b,

$$\begin{aligned}
&\Big|\Pr[\mathcal{D}(\mathcal{S}(x))=1] - \Pr[\mathcal{D}(\widetilde{\mathcal{S}(x)})=1]\Big| \\
&\quad= \Big|\Pr_{(a,b)\leftarrow(A,B)}[\mathcal{D}(a,b)]=1|a=\omega] - \Pr_{(a,b)\leftarrow(\widetilde{A},\widetilde{B})}[\mathcal{D}(a,b)]=1|a=\omega]\Big| \\
&\quad\le \ell\cdot\delta_h^b.
\end{aligned}$$

Note that by the construction of the simulator and the fact that the inner simulator $\mathcal{S}_{\mathrm{IPCP}}$ simulates the IPCP verifier's view *perfectly*, we have that as long as the simulator predicts the verifier's randomness correctly in Step 1b, the output of the simulator, excluding the commitments to the PCP entries that are not revealed, is identical to the view of the verifier.

Let E be the event that simulator predicts the verifier's randomness correctly. By Proposition 6.7 and applying the union bound on all k verifier commitments, we have that $\Pr_{\mathsf{View}_{\mathcal{V}^*}^{\mathcal{P}}(x)}[\neg E] \le k\cdot 3\sqrt{\delta_b^h}$. Similarly, by Remark 6.9 we have that $\Pr_{\widetilde{\mathcal{S}(x)}}[\neg E] \le k\cdot 3\sqrt{\delta_b^h}$.

Thus, formally, the conditional distributions $(A'|E)$ and $(\widetilde{A}|E)$ are identical, and so by the definition of $\widetilde{B}$ we get that $(\widetilde{\mathcal{S}(x)}|E)$ and $(\mathsf{View}_{\mathcal{V}^*}^{\mathcal{P}}(x)|E)$ are also identical. Therefore, for all distinguishers $\mathcal{D}$,

$$\begin{aligned}
&\Big|\Pr\big[\mathcal{D}(\widetilde{\mathcal{S}(x)})=1\big] - \Pr\big[\mathcal{D}(\mathsf{View}_{\mathcal{V}^*}^{\mathcal{P}}(x))=1\big]\Big| \\
&\quad\le \Big|\Pr\big[\mathcal{D}(\widetilde{\mathcal{S}(x)})=1|E\big] - \Pr\big[\mathcal{D}(\mathsf{View}_{\mathcal{V}^*}^{\mathcal{P}}(x))=1|E\big]\Big| \\
&\qquad\qquad + \Big|\Pr_{\widetilde{\mathcal{S}(x)}}[E] - \Pr_{\mathsf{View}_{\mathcal{V}^*}^{\mathcal{P}}(x)}[E]\Big| + \Pr_{\widetilde{\mathcal{S}(x)}}[\neg E] + \Pr_{\mathsf{View}_{\mathcal{V}^*}^{\mathcal{P}}(x)}[\neg E] \\
&\quad\le 9k\sqrt{\delta_b^h}.
\end{aligned}$$

In total, by the triangle inequality, we have that

$$\begin{aligned}
&\left|\Pr\left[\mathcal{D}(\mathcal{S}(x)) = 1\right] - \Pr\left[\mathcal{D}(\mathsf{View}^{\mathcal{P}}_{\mathcal{V}^*}(x)) = 1\right]\right| \\
&\qquad \le \left|\Pr\left[\mathcal{D}(\mathcal{S}(x)) = 1\right] - \Pr\left[\mathcal{D}(\widetilde{\mathcal{S}(x)}) = 1\right]\right| + \left|\Pr\left[\mathcal{D}(\widetilde{\mathcal{S}(x)}) = 1\right] - \Pr\left[\mathcal{D}(\mathsf{View}^{\mathcal{P}}_{\mathcal{V}^*}(x)) = 1\right]\right| \\
&\qquad \le 9k\sqrt{\delta^h_b} + \ell \cdot \delta^b_h,
\end{aligned}$$

as required.

□

7.4 Bringing It All Together: zkSIP for Low-Depth NP Relations

We start with the following theorem by Chiesa *et al.* from which we get a zero-knowledge proof for any language that is decidable by a family of low-depth circuits. To align with our requirements, we state the theorem in the context of holographic IPCPs that are zero-knowledge against semi-malicious verifiers. The original paper presents the theorem for malicious verifiers, which inherently includes the semi-malicious setting we are considering. Although the term "holographic" is not explicitly used in the original paper, it is clear that the theorem applies to this context as well.

Theorem 7.13 ([CFS17, Theorem 11.1])**.** *Let $\mathcal{L}$ be a language decidable by a family of $O(\log S(n))$-space uniform Boolean circuits of size $S(n)$ and depth $D(n)$. Then $\mathcal{L}$ has a (public-coin and non-adaptive) Holographic Interactive PCP that is perfect zero-knowledge against all semi-malicious verifiers. The proof has a soundness error $\frac{1}{2}$. Its round complexity, query complexity, and verifier space is $D(n) \cdot \mathrm{polylog}(S(n))$, and the proof complexity is $D(n) \cdot \mathrm{poly}(S(n))$. The simulator runs in space $\mathrm{poly}(D(n), \log(S(n)))$.*

By applying the techniques from [CFS17] with minor adjustments to the construction it is straightforward to extend the IPCP to *non-deterministic* low-depth computations. This is captured by the following theorem.

Theorem 7.14. *Let $\mathcal{R}$ be an NP relation decidable by a family of $O(\log S(n))$-space uniform Boolean circuits of size $S(n)$ and depth $D(n)$, and let $\mathcal{L} \in \mathsf{NP}$ be the language defined by $\mathcal{R}$. Then $\mathcal{L}$ has a (public-coin and non-adaptive) Holographic Interactive PCP that is perfect zero-knowledge against all semi-malicious verifiers. The proof has a soundness error $\frac{1}{2}$. Its round complexity, query complexity, verifier space and simulator space is $\mathrm{poly}(D(n) \cdot \mathrm{polylog}(S(n)))$, and the proof complexity is $\mathrm{poly}(D(n) \cdot \mathrm{poly}(S(n)))$.*

As the proof is a simple extension of the techniques of [CFS17], we provide a proof sketch in Appendix A.

We are now ready to prove our main theorem, Theorem 7.1.

Proof of Theorem 7.1. Let $\mathcal{R}$ be an NP relation determined by a $\log(S(n))$-space uniform family of circuits of size $S(n) = \mathrm{poly}(n)$ and depth $D(n) = O(\mathrm{polylog}(n))$ and let $\mathcal{L} \in \mathsf{NP}$ be the language defined by $\mathcal{R}$. By Theorem 7.14, $\mathcal{L}$ has a perfect semi-malicious zero-knowledge IPCP protocol where the honest verifier space, the honest verifier's randomness, communication complexity, round complexity and the simulator space are $O(D(n) \cdot \mathrm{polylog}(S(n)))$. The PCP proof length is $O(D(n) \cdot \mathrm{poly}(S(n)))$.

The soundness error of the IPCP protocol is $\frac{1}{2}$, so we apply $t = \mathrm{polylog}(n)$ parallel independent repetitions to reduce the soundness error to $\mathsf{negl}(n)$. Each repetition includes both the PCP string and the subsequent interactive part of the protocol. This increases all the relevant parameters by a factor of t, except for the round complexity. It is important to note that *semi-malicious* zero-knowledge is preserved under parallel repetition, unlike fully malicious zero-knowledge, as discussed in [GK96].

Let λ, ϵ be security parameters such that $\lambda^\epsilon \geq \log^2(n)$ and $\lambda = \Omega(D(n) \cdot \mathrm{polylog}(S(n)))$. Take Construction 4.13 for a streaming statistically hiding commitment with parameters $s^h_{\mathcal{S}} = O\big(\lambda \cdot D(n) \cdot \mathrm{polylog}(S(n))\big), s^h_{\mathcal{R}} = O\big(\lambda^{1+\epsilon} + \lambda \cdot D(n) \cdot \mathrm{polylog}(S(n))\big), s^h_b = \frac{\lambda^2}{21}, \delta^h_h = 2^{-\Omega(\lambda)}, \delta^h_b = 2^{-\Omega(\lambda^\epsilon)}$, and take Construction 5.9 for a streaming statistically binding commitment with parameters $s^b_{\mathcal{S}} = O\big(\lambda^{1+\epsilon}\big), s^b_{\mathcal{R}} = O\big(\lambda^{1+\epsilon}\big), s^b_h = \frac{\lambda^2}{4}, \delta^b_h = 2^{-\Omega(\lambda)}, \delta^b_b = 2^{-\Omega(\lambda^\epsilon)}$

Apply Compiler 7.7 with the ingredients above to get a zero-knowledge streaming interactive proof for $\mathcal{L}$ with perfect completeness, soundness error $\mathsf{negl}(n) + O(t \cdot D(n) \cdot \mathrm{polylog}(S(n))) \cdot 2^{-\Omega(\lambda^\epsilon)}$, honest verifier space $O\big(t \cdot D(n) \cdot \mathrm{polylog}(S(n)) \cdot \lambda + \lambda^{1+\epsilon}\big)$, and communication complexity $O(t \cdot D(n) \cdot \mathrm{poly}(S(n)) \cdot \lambda^2)$.

By Proposition 7.12, the proof is zero-knowledge per Definition 3.6 against adversaries with space $\frac{\lambda^2}{21}$ with zero-knowledge error $O(D(n) \cdot \mathrm{poly}(S(n))) \cdot 2^{-\Omega(\lambda^\epsilon)}$ □

Choose $\lambda = \log^c(n)$ for a large-enough constant c, and we get the following corollary.

Corollary 7.15. *Let $\mathcal{R}$ be an* NP *relation, and let $\delta > 0$. There exists a zero-knowledge streaming interactive proof for $\mathcal{R}$ with perfect completeness, soundness error* $\mathsf{negl}(n)$*, honest verifier space $s = \mathrm{polylog}(n)$, and communication complexity* $\mathrm{poly}(n)$*. The proof is zero-knowledge against adversaries with space $s^{2-\delta}$, with a negligible zero-knowledge error.*

References

[Bar01] Boaz Barak. How to go beyond the black-box simulation barrier. In *42nd Annual Symposium on Foundations of Computer Science, FOCS 2001, 14-17 October 2001, Las Vegas, Nevada, USA*, pages 106–115. IEEE Computer Society, 2001. 5

[BSCF+17] Eli Ben-Sasson, Alessandro Chiesa, Michael A Forbes, Ariel Gabizon, Michael Riabzev, and Nicholas Spooner. Zero knowledge protocols from succinct constraint detection. In *Theory of Cryptography: 15th International Conference, TCC 2017, Baltimore, MD, USA, November 12-15, 2017, Proceedings, Part II 15*, pages 172–206. Springer, 2017. 16, 64, 65

[CCM+15] Amit Chakrabarti, Graham Cormode, Andrew McGregor, Justin Thaler, and Suresh Venkatasubramanian. Verifiable stream computation and Arthur-Merlin communication. *Leibniz international proceedings in informatics (LIPIcs)*, pages 217–243, 2015. 3

[CCM+19] Amit Chakrabarti, Graham Cormode, Andrew McGregor, Justin Thaler, and Suresh Venkatasubramanian. Verifiable stream computation and Arthur–Merlin communication. *SIAM Journal on Computing*, 48(4):1265–1299, 2019. 8

[CDGH23] Graham Cormode, Marcel Dall'Agnol, Tom Gur, and Chris Hickey. Streaming zero-knowledge proofs. *arXiv preprint arXiv:2301.02161*, 2023. 3, 4, 5, 6, 7, 8, 9

[CFS17] Alessandro Chiesa, Michael A Forbes, and Nicholas Spooner. A zero knowledge sumcheck and its applications. *arXiv preprint arXiv:1704.02086*, 2017. 14, 16, 52, 53, 61, 64, 65

[CGT20] Amit Chakrabarti, Prantar Ghosh, and Justin Thaler. Streaming verification for graph problems: Optimal tradeoffs and nonlinear sketches. *arXiv preprint arXiv:2007.03039*, 2020. 3

[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, 2012. 3

[CTY11] Graham Cormode, Justin Thaler, and Ke Yi. Verifying computations with streaming interactive proofs. *arXiv preprint arXiv:1109.6882*, 2011. 3

[GK96] Oded Goldreich and Ariel Kahan. How to construct constant-round zero-knowledge proof systems for NP. *Journal of Cryptology*, 9(3):167–189, 1996. 9, 62

[GKR15] Shafi Goldwasser, Yael Tauman Kalai, and Guy N Rothblum. Delegating computation: interactive proofs for Muggles. *Journal of the ACM (JACM)*, 62(4):1–64, 2015. 4, 14

[GMR89] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal on Computing*, 18(1):186–208, 1989. 3, 4

[Gol08] Oded Goldreich. *Computational complexity - a conceptual perspective.* Cambridge University Press, 2008. 5

[Gol18] Oded Goldreich. On doubly-efficient interactive proof systems. *Found. Trends Theor. Comput. Sci.*, 13(3):158–246, 2018. 4

[GR15] Tom Gur and Ran Raz. Arthur–Merlin streaming complexity. *Information and Computation*, 243:145–165, 2015. 3

[GZ19] Jiaxin Guan and Mark Zhandary. Simple schemes in the bounded storage model. In *Advances in Cryptology–EUROCRYPT 2019: 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Darmstadt, Germany, May 19–23, 2019, Proceedings, Part III 38*, pages 500–524. Springer, 2019. 10

[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. 10, 12, 21

[KR08] Yael Tauman Kalai and Ran Raz. Interactive PCP. In *International Colloquium on Automata, Languages, and Programming*, pages 536–547. Springer, 2008. 14, 52

[Raz18] Ran Raz. Fast learning requires good memory: A time-space lower bound for parity learning. *Journal of the ACM (JACM)*, 66(1):1–18, 2018. 11, 20, 21

A PZK Holographic IPCP for Low-Depth NP Relations

Given a semi-malicious verifier PZK Holographic IPCP for low-depth uniform boolean circuits, denoted $\langle \mathcal{P}, \mathcal{V} \rangle$, we construct a semi-malicious verifier PZK Holographic IPCP for low-depth NP relations, denoted by $\langle \mathcal{P}', \mathcal{V}' \rangle$.

Let $\mathbb{F}$ and $H \subseteq \mathbb{F}$ be the parameters over which the original holographic IPCP holds such that $|H|$ is greater than the runtime of $\mathcal{V}$ (polylogarithmic in the case of low-depth circuits).

We proceed to describe the IPCP relative to an input x and witness w. Assume without loss of generality that $|x| = |w| = 2^m$ for some $m \in \mathbb{N}$,[16] and let $\hat{x} : \mathbb{F}^m \to \mathbb{F}$ be the individual-degree $|H| - 1$ extension of x, where the m-dimensional boolean hypercube contains the values of x, and the other $|H|^m - 2^m$ entries are zeros.[17] Let $\hat{w} : \mathbb{F}^m \to \mathbb{F}$ be a randomized individual-degree $|H| - 1$ extension of w, for $\{0, 1\} \subsetneq H \subsetneq \mathbb{F}$. Let $\widehat{xw} : \mathbb{F}^{m+1} \to \mathbb{F}$ be an individual-degree (at most) $|H| - 1$ extension of (x, w), defined as $\widehat{xw}(b, \alpha) = b \cdot \hat{x}(\alpha) + (1 - b) \cdot \hat{w}(\alpha)$, for $b \in \mathbb{F}$, $\alpha \in \mathbb{F}^m$.

The protocol $\langle \mathcal{P}', \mathcal{V}' \rangle$ relative to the input x and witness w proceeds by running $\langle \mathcal{P}, \mathcal{V} \rangle$ relative to the concatenated input (x, w). The PCP sent by $\mathcal{P}'$ includes the PCP $\tilde{\pi}$ of $\mathcal{P}$, along with other components to be specified later. When $\mathcal{V}$ queries the PCP, $\mathcal{V}'$ redirects the query to $\tilde{\pi}$. Whenever $\mathcal{V}$ makes an input query to $\widehat{xw}$ at point $(b, \alpha) \in \mathbb{F} \times \mathbb{F}^m$, the verifier $\mathcal{V}'$ provides it with the answer in the following manner. Any point (b, α) in $\widehat{xw}$ can be computed by a simple linear combination of $\hat{x}(\alpha)$ and and $\hat{w}(\alpha)$. In order to get $\hat{x}(\alpha)$, it simply queries its input (observe that this indeed constitutes holographic access to the input). Clearly, in order to preserve zero-knowledge, the verifier may not query $\hat{w}$ on any value in $\{0, 1\}^m$, and so if $\mathcal{V}$ requests to see a point there, $\mathcal{V}'$ immediately accepts. Intuitively, because $\hat{w}$ was padded with random elements, for up to $|H| - 1$ queries, querying $\hat{w}$ outside of the boolean hypercube does not yield any information about w. Thus, if $\mathcal{V}$ requests a legal (i.e., not a Boolean-valued) point in the low-degree extension of the witness, $\mathcal{V}'$ obtains the requested value as discussed next.

Obviously, if $\mathcal{V}'$ requests the value directly from $\mathcal{P}'$, it will not maintain the soundness the protocol. Likewise, if $\mathcal{P}'$ sends all of $\hat{w}$ as a part of the PCP in the beginning, then a malicious verifier could query $\hat{w}$ in the boolean hypercube, and by such learn the witness. To circumvent this, $\mathcal{P}'$ will send as a part of the PCP an "algebraic commitment" to $\hat{w}$, whose binding and hiding properties resolve the issues described above. Then $\mathcal{P}'$ can de-commit at the specific queried point without revealing more information about $\hat{w}$.

In more detail, the algebraic commitment consists of an $(m + m')$-variate polynomial $Z : \mathbb{F}^{m+m'} \to \mathbb{F}$, such that for all $\alpha \in \mathbb{F}^m$, it holds that $\sum_{\beta \in G^{m'}} Z(\alpha, \beta) = \hat{w}(\alpha)$, for some $G \subsetneq \mathbb{F}$. Thus, in order to prove the claim $\hat{w}(\alpha) = v$, the prover and verifier will engage in the zero-knowledge Sumcheck protocol, per [BSCF+17, Theorem 5.3] on the claim $\sum_{\beta \in G^{m'}} Z(\alpha, \beta) = v$. Note that for the execution of the zero-knowledge Sumcheck, the prover must provide an auxiliary random polynomial π of the appropriate dimension and degree, which will be appended to the PCP.

In sum, the PCP sent by the prover $\mathcal{P}'$ will contain the original PCP of $\mathcal{P}$, the algebraic commitment to $\hat{w}$ as the polynomial Z, and the auxiliary random polynomials $\{\pi_i\}$ (one polynomial for each query). Note that in addition to the protocol as specified thus far, $\mathcal{V}'$ will need to perform a low-degree test on Z, and query them using a self-correction procedure.[18]

[16] Otherwise pad with zeros.

[17] For our streaming purposes, we may assume that without loss of generality the IPCP verifier has access to such an extension of x as specified, since when converted into a streaming verifier, it can compute any point in the low-degree extension in small space given streaming access to x.

[18] Chiesa *et al.* [CFS17] offer a comprehensive account of this technique. A detailed example of a similar protocol

We now proceed to give a proof sketch that the protocol above is indeed a semi-honest verifier perfect zero-knowledge Holographic IPCP.

Completeness. The perfect completeness of $\langle \mathcal{P}', \mathcal{V}' \rangle$ relies on the perfect completeness of $\langle \mathcal{P}, \mathcal{V} \rangle$ and that of the zero-knowledge interactive proof of [BSCF+17].

Soundness. A soundness error in $\langle \mathcal{P}', \mathcal{V}' \rangle$ may arise from four sources. The first is the constant error of $\langle \mathcal{P}, \mathcal{V} \rangle$. Second, recall that according to our definition of $\langle \mathcal{P}', \mathcal{V}' \rangle$, if $\mathcal{V}$ queries a forbidden value, $\mathcal{V}'$ immediately accepts. Thus, each such query incurs an error of $\left(\frac{2}{\mathbb{F}}\right)^m$, which is (at most) an inverse polynomial in our parameter regime. Third, the execution of the zero-knowledge proof for each value of $\hat{w}$ has an inverse polynomial error, and lastly, more inverse polynomial errors may occur during the low-degree testing, and the self-correction algorithm used to query the PCP. Recall that the number of queries performed by the verifier is bounded by its runtime, which, in the case of protocols over circuits of polylogarithmic depth, is polylogarithmic. Thus, after applying the union bound on all of the above, we are still left with a constant error. In order to reduce the error we may use repetition, in which case we can reduce the error to become negligible, while not incurring a significant increase in the other parameters of the protocol.[19]

Semi-Malicious Zero-Knowledge. To argue zero-knowledge, we construct a simulator. The simulator $\mathcal{S}'$ for $\langle \mathcal{P}', \mathcal{V}' \rangle$ will run the simulator $\mathcal{S}$ of $\langle \mathcal{P}, \mathcal{V} \rangle$. When $\mathcal{V}'$ makes the i-th query to $\hat{w}$, the simulator $\mathcal{S}'$ will use the efficient conditional sampling algorithm of Ben-Sasson *et al.* [BSCF+17, Corollary 4.10] to provide a sample $\{v_i\}$ from a *random* polynomial R of dimension and degree similar to those of $\hat{w}$, conditioned on all the samples $v_j, j \in [i-1]$ that it has already provided. It will then use the simulator of [BSCF+17] to simulate the zero-knowledge Sumcheck protocol for the value v_i.

We show that $\mathcal{S}'$ simulates the view of $\mathcal{V}'$ perfectly. Consider $\mathcal{S}''$ that operates similar to $\mathcal{S}'$, but has full access to $\hat{w}$, and instead of sending random values, it would send the real ones. In this case it is clear that the simulation is perfect since all other simulators provide perfect simulation. Now, as mentioned earlier, due to the padding of $\hat{w}$, given a limited number of queries, it is distributed uniformly. Thus, the distribution produced by $\mathcal{S}'$ is identical to that produced by $\mathcal{S}'$, which in turn is identical to the real view of the verifier.

B Deferred Proofs

B.1 Proof of Lemma 5.7

Lemma B.1 (Lemma 5.7, restated)**.** *Let $(X, Y), (X', Y')$ be joint distributions, each defined over a product space $\Omega_X \times \Omega_Y$, where Ω_Y is the space of all matrices $y \in \{0,1\}^{k \times \ell}$. Suppose the marginal distributions X, X' are identically distributed and define com(y) to be the concatenation of statistically binding streaming commitments per Definition 5.2 of the k rows of y. Then, for all space-s_h ROBP distinguishers $\mathcal{D}$,*

$$\left| \Pr_{(x,y)\leftarrow(X,Y)}[\mathcal{D}(x, com(y))] = 1] - \Pr_{(x,y)\leftarrow(X',Y')}[\mathcal{D}(x, com(y))] = 1] \right| \leq k \cdot \delta_h.$$

appears in their proof of [CFS17, Theorem 7.2], which presents the PZK-IPCP protocol for NEXP.

[19] Recall that semi-malicious zero-knowledge is closed under parallel repetition.

Proof. First, consider the following claim.

Claim B.1.1. *For any streaming distinguisher $\mathcal{D}$ with space s_h,*

$$\left| \Pr_{y \leftarrow Y}[\mathcal{D}(com(y)) = 1] - \Pr_{y \leftarrow Y'}[\mathcal{D}(com(y)) = 1] \right| \leq k \cdot \delta_h.$$

Proof. We first claim that for any two fixed matrices $y, y' \in \{0,1\}^{k \times \ell}$ and any streaming distinguisher $\mathcal{D}$ with space s_h,

$$\left| \Pr[\mathcal{D}(com(y)) = 1] - \Pr[\mathcal{D}(com(y')) = 1] \right| \leq k \cdot \delta_h.$$

Assume the contrary and let y, y' be matrices and $\mathcal{D}$ be a distinguisher for which the claim does not hold. Consider for all $i \in [k]$ the hybrid matrix $(y, y')_i$ which consists of the first i rows of y and the last $k - i$ rows of y'. Note that $(y, y')_k = y$ and $(y, y')_0 = y'$. If so, there exists $i \in [k]$ such that

$$\left| \Pr[\mathcal{D}(com((y, y')_i)) = 1] - \Pr[\mathcal{D}(com((y, y')_{i-1})) = 1] \right| > \delta_h.$$

Construct the following streaming distinguisher $\mathcal{D}'$ with space s_h that has the first $i - 1$ rows of y and the last $k - i$ rows of y' hardcoded and streaming access to $\mathcal{D}$. Given a streaming commitment of a vector $v \in \{0,1\}^\ell$, for all $j \in [i-1]$, the distinguisher $\mathcal{D}'$ will stream commitments of the $i - 1$ rows of y to $\mathcal{D}$, followed by the input streaming commitment to v and streaming commitments to the last $k - i$ rows of y'. The output of $\mathcal{D}'$ will correspond to that $\mathcal{D}$. Thus, by construction, $\mathcal{D}'$ distinguishes with probability greater than δ_h between com_{y_i} and $com_{y'_i}$, in contradiction to the hiding property of the statistically binding commitment.

Since we showed that the claim is true for all fixed pairs of matrices $y, y' \in \{0,1\}^{k \times \ell}$, the lemma follows from an averaging argument, details follow.

Suppose there exist distributions Y and Y' over matrices for which D distinguishes with some gap ϵ. Assuming wlog that $\Pr[D(Y) = 1] > \Pr[D(Y') = 1]$, this means that:

$$\Pr[D(Y) = 1] - \Pr[D(Y') = 1] \geq \epsilon.$$

We can rewrite this as:

$$\mathbb{E}_{y \leftarrow Y, y' \leftarrow Y'} \big[\Pr[D(y) = 1] - \Pr[D(y') = 1] \big] \geq \epsilon.$$

Which means in turn that there exist fixed strings y and y' for which $\Pr[D(y) = 1] - \Pr[D(y') = 1] \geq \epsilon$. □

By the law of total probability and the definition of conditional probability, we have that

$$\begin{aligned}
&\left| \Pr_{(x,y) \leftarrow (X,Y)}[\mathcal{D}(x, com(y))] = 1] - \Pr_{(x,y) \leftarrow (X',Y')}[\mathcal{D}(x, com(y))] = 1] \right| \\
&= \Big| \sum_{\omega \in \Omega_X} \Big(\Pr_{(x,y) \leftarrow (X,Y)}[\mathcal{D}(x, com(y))] = 1 | x = \omega] \cdot \Pr_{x \leftarrow X}[x = \omega] \\
&\qquad - \Pr_{(x,y) \leftarrow (X',Y')}[\mathcal{D}(x, com(y))] = 1 | x = \omega] \cdot \Pr_{x \leftarrow X'}[x = \omega] \Big) \Big|.
\end{aligned}$$

Let $\mathcal{D}_x$ be a distinguisher that has x hardcoded and runs on input $com(y)$ similarly to $\mathcal{D}$ on $(x, com(y))$. Since X, X' are identically distributed, we can upper-bound the expression above by

$$\sum_{\omega \in \Omega_X} \Pr_{x \leftarrow X}[x = \omega] \cdot \left| \Pr_{(x,y) \leftarrow (X,Y)}[\mathcal{D}_x(com(y))] = 1 | x = \omega] - \Pr_{(x,y) \leftarrow (X',Y')}[\mathcal{D}_x(com(y))] = 1 | x = \omega] \right|,$$

which, using Claim B.1.1, can be upper bounded by $k \cdot \delta_h$. □