Perfect Zero-Knowledge PCPs for #P

Tom Gur
University of Cambridge
United Kingdom
tom.gur@cl.cam.ac.uk

Jack O'Connor
University of Cambridge
United Kingdom
jo496@cam.ac.uk

Nicholas Spooner
University of Warwick
United Kingdom
NYU
USA
nicholas.spooner@warwick.ac.uk

ABSTRACT

We construct perfect zero-knowledge probabilistically checkable proofs (PZK-PCPs) for every language in #P. This is the first construction of a PZK-PCP for any language outside BPP. Furthermore, unlike previous constructions of (statistical) zero-knowledge PCPs, our construction simultaneously achieves non-adaptivity and zero knowledge against arbitrary (adaptive) polynomial-time malicious verifiers.

Our construction consists of a novel masked sumcheck PCP, which uses the combinatorial nullstellensatz to obtain antisymmetric structure within the hypercube and randomness outside of it. To prove zero knowledge, we introduce the notion of locally simulatable encodings: randomised encodings in which every local view of the encoding can be efficiently sampled given a local view of the message. We show that the code arising from the sumcheck protocol (the Reed–Muller code augmented with subcube sums) admits a locally simulatable encoding. This reduces the algebraic problem of simulating our masked sumcheck to a combinatorial property of antisymmetric functions.

CCS CONCEPTS

• **Theory of computation → Complexity classes**; **Interactive proof systems**; **Error-correcting codes**.

KEYWORDS

Computational Complexity, Coding Theory, Cryptography

ACM Reference Format:
Tom Gur, Jack O'Connor, and Nicholas Spooner. 2024. Perfect Zero-Knowledge PCPs for #P. In *Proceedings of the 56th Annual ACM Symposium on Theory of Computing (STOC '24), June 24–28, 2024, Vancouver, BC, Canada.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3618260.3649698

1 INTRODUCTION

The notion of a *zero-knowledge (ZK) proof*, a proof that conveys no information except the truth of a statement, is one of the most influential ideas in cryptography and complexity theory of the past four decades.

Zero knowledge was originally defined by Goldwasser, Micali and Rackoff [18] in the context of interactive proofs (IPs). The deep and beautiful insight in this work is that it is possible to rigorously *prove* that an interaction does not convey information, by exhibiting an efficient algorithm called the *simulator* which can generate the distribution of protocol transcripts without interacting with the prover. In that work, the authors identify three different notions of zero knowledge, depending on the quality of the simulation. These are: (1) *perfect* zero knowledge (PZK), where the simulator's distribution is *identical* to the real distribution of transcripts; (2) *statistical* zero knowledge (SZK), where the distributions are inverse-exponentially close; and (3) *computational* zero knowledge (CZK), where the distributions are computationally indistinguishable.

This hierarchy led naturally to the study of the complexity classes PZK, SZK, CZK of languages admitting ZK interactive proofs. Seminal results show that PZK contains interesting "hard" languages, including quadratic residuosity and nonresiduosity [18] and graph isomorphism and nonisomorphism [17]. Despite this, the structure of the class PZK, and its relation to other complexity classes, remains poorly understood. In light of this difficulty, the study of zero knowledge has followed two main routes: (1) studying the "relaxed" notions of SZK and CZK, and (2) studying zero knowledge in other models.

The former line of work has proved highly fruitful. A seminal result of [17] showed that CZK = IP = PSPACE, which launched the cryptographic study of ZK proofs, yielding a plethora of theoretical and practical results [29]. The study of SZK has revealed that this class has a rich structure, and deep connections within complexity theory (cf. [30]).

The present work lies along the second route, also hugely influential. The seminal work of Ben-Or, Goldwasser, Kilian, and Wigderson [8] introduced the model of multi-prover interactive proofs (MIP) in order to achieve perfect zero-knowledge without any computational assumptions [27, 15]. This in turn inspired some of the most important models and results in contemporary theoretical computer science, including entangled-prover interactive proofs (MIP*), interactive oracle proofs (IOPs), and most notably, probabilistically checkable proofs (PCPs). Indeed, the celebrated PCP theorem [6, 7, 14] is widely recognised as one of the most important achievements of modern complexity theory [2].

Zero-knowledge PCPs. Recall that a PCP is a proof which can be verified, with high probability, by a verifier that only makes a small number of queries (even $O(1)$) to the proof (cf. [5]). A zero-knowledge PCP is a *randomised* proof which, in addition to being locally checkable, satisfies a zero knowledge condition: the view of any efficient (malicious) verifier can be efficiently simulated (see [31] for a survey). Similarly to the IP case, we can distinguish PCPs

STOC '24, June 24–28, 2024, Vancouver, BC, Canada

ACM ISBN 979-8-4007-0383-6/24/06
https://doi.org/10.1145/3618260.3649698

with perfect, statistical and computational zero knowledge. Note that there appears to be no formal relationship between ZK-IPs and ZK-PCPs: standard transformations from IP to PCP spoil zero knowledge.

The first *statistical* zero-knowledge PCPs appeared in the seminal work of Kilian, Petrank and Tardos [26]. Later works [21, 22] simplified this construction, and extended it to PCPs of proximity. These constructions operate via a 2-step compilation of (non-zero-knowledge) PCP: first the PCP is endowed with honest-verifier statistical zero-knowledge (HVSZK), and then the latter property is boosted into full statistical zero knowledge by "forcing" the malicious verifier's query pattern to be similarly distributed to that of the honest verifier, using an object called a locking scheme. Kilian, Petrank and Tardos use this transformation to prove that NEXP has SZK-PCPs (with a polynomial-time verifier) that are zero knowledge against all polynomial-time malicious verifiers; i.e., SZK-PCP[poly, poly] = NEXP.

Unfortunately, locking schemes have two inherent drawbacks: they require the honest PCP verifier to be adaptive and they cannot achieve perfect zero knowledge. Another line of work, motivated by cryptographic applications, focuses on obtaining SZK-PCPs for NP with a *non-adaptive* honest verifier from leakage resilience. These results come with caveats, achieving either a weaker notion of zero knowledge known as *witness indistinguishability* [23], or simulation against adversaries making only quadratically many more queries than the honest verifier [20].

The complexity landscape of perfect zero knowledge is subtle. As discussed above, we know that PZK-MIP = MIP = NEXP [8], where MIP is the class of languages with a multi-prover interactive proof. The quantum analogue of this result, PZK-MIP* = MIP* = RE, is also known to hold [12, 19]. We know a similar result for interactive PCPs (IPCPs) [25], an interactive generalisation of PCPs (and special case of IOPs): PZK-IPCP = IPCP = NEXP [13]. For IPs, the picture is very different. We know that PZK ⊆ SZK ⊆ AM ∩ coAM [16, 3], and so it is unlikely that PZK = IP (= PSPACE), or even NP ⊆ PZK. It is unknown whether PZK = SZK; indeed, it was recently shown that there is an oracle relative to which this equality does not hold [10].

In contrast, *nothing at all* is known about the class PZK-PCP except for the trivial inclusion BPP ⊆ PZK-PCP. In particular, the following key question in the study of perfect zero knowledge remains open:

*Do perfect zero-knowledge PCPs exist for **any** non-trivial language?*

1.1 Our Results

In this work, we give strong positive answer to this question, showing that there exist perfect zero-knowledge PCPs (PZK-PCPs) for all of #P.

Theorem 1 (Informally stated). #P ⊆ PZK-PCP[poly, poly].

We prove Theorem 1 by constructing a PZK-PCP for the #P-complete language

$$\#\mathsf{SAT} = \{(\Phi, N) : \Phi \text{ is a CNF}, \sum_{x\in\{0,1\}^n} \Phi(x) = N\}.$$

This is the first construction of a PZK-PCP for a language (believed to be) outside BPP. Furthermore, unlike previous constructions of zero-knowledge PCPs, our construction simultaneously achieves *non-adaptivity* for the honest verifier and zero knowledge against arbitrary (adaptive) polynomial-time malicious verifiers. We stress that Theorem 1 is unconditional and does not rely on any cryptographic assumptions.

Remark 1.1. *As with its IP counterpart* PZK, PZK-PCP *is a class of* decision *problems, and so the inclusion* #P ⊆ PZK-PCP *refers to the decision version of* #P *(for which* #SAT *is complete). This class contains* UP *and* coNP *(in a natural way) but may not contain* NP. *Toda's theorem, that* PH ⊆ $\mathsf{P}^{\#\mathsf{P}}$*, does not directly imply that* PH ⊆ PZK-PCP*, as the* #P *oracle in this inclusion is a function oracle.*

On the way to proving Theorem 1, we solve the following general algebraic-algorithmic problem, which we consider to be of independent interest.

Problem 1 (Local simulation of low-degree extensions). *Given oracle access to a function $f\colon \{0,1\}^m \to \mathbb{F}$, where $\mathbb{F}$ is a finite field, efficiently simulate oracle access to a uniformly random* **degree-d extension** *of f; that is, a function $\tilde{f}$ drawn uniformly at random from the set of m-variate polynomials over $\mathbb{F}$ of individual degree d that agree with f on $\{0,1\}^m$.*

By "efficient" we mean polynomial in m, d and $\log|\mathbb{F}|$. Chen et al. [11] give a *query-efficient* simulator for $d \geq 2$, based on observations of Aaronson and Wigderson [1]. (For $d = 1$, there is a query lower bound of 2^m [24].) In this work we give a *computationally* efficient simulator for $d \geq 2$.

Theorem 2 (informal). *There is an efficient algorithm solving Problem 1 for $d \geq 2$.*

2 TECHNIQUES

We outline the various techniques we employ in this work. Please refer to the full version of the paper for the technical details and proofs.

We start by outlining how to construct (non-ZK) PCPs for #SAT via the sumcheck protocol [28]. Recall that the prover and verifier in the sumcheck protocol have oracle access to an m-variate polynomial P (the arithmetisation of Φ) of individual degree d over a finite field $\mathbb{F}$, and the prover wishes to convince the verifier that $\sum_{\vec{a}\in\{0,1\}^m} P(\vec{a}) = \gamma$ for some $\gamma \in \mathbb{F}$. This is achieved via a $2m$-message protocol, in which for every $i \in [m]$,

- the $2i$-th message, sent by the verifier, is a uniformly random challenge $c_i \in \mathbb{F}$; and
- the $(2i-1)$-th message, sent by the prover, is the univariate polynomial

$$g_i(X) = \sum_{\vec{a}\in\{0,1\}^{m-i}} P(c_1, \ldots, c_{i-1}, X, a_{i+1}, \ldots, a_m).$$

The verifier accepts if (a) $g_1(0)+g_1(1) = \gamma$; (b) for each $1 \leq i \leq m-1$, $g_i(c_i) = g_{i+1}(0)+g_{i+1}(1)$; and (c) $g_m(c_m) = P(c_1, \ldots, c_m)$. This protocol is complete and sound for sufficiently large $\mathbb{F}$. Moreover, we can "unroll" the interactive sumcheck protocol into an (exponentially large) PCP by writing down the prover's answers to each possible (sub-)sequence of challenges.

Unfortunately, the sumcheck PCP is *clearly not zero knowledge*: even the prover's first message g_1 reveals information about P

which is #P-hard to compute. In this overview, we will explain how to modify the sumcheck PCP in order to achieve perfect zero knowledge.

Zero knowledge IOPs. Our approach to constructing a PZK-PCP for sumcheck is inspired by the [9] construction of a perfect zero knowledge sumcheck in the *interactive oracle proof* (IOP) model. The IOP model is an interactive generalisation of PCPs, in which the prover and verifier interact across multiple rounds, with the prover sending a PCP oracle in each round.

In their IOP, the prover first sends the evaluation table of a uniformly random m-variate polynomial R such that $\sum_{\vec{a}\in\{0,1\}^m} R(\vec{a}) = 0$, which will be used as a mask. Next, to ensure soundness in case R does not sum to zero, the verifier sends a challenge α. Finally, the prover sends a sumcheck PCP for the statement $\sum_{\vec{a}\in\{0,1\}^m} \alpha P(\vec{a}) + R(\vec{a}) = \alpha\gamma$.

Intuitively, this protocol does not leak much information about P, because $\alpha P + R$ is a *uniformly random* polynomial that sums to $\alpha\gamma$. [9] shows, using techniques from algebraic complexity theory, that this IOP is indeed perfect zero knowledge; we will make use of these techniques later in our construction.

Unfortunately, the interaction in this protocol is crucial in order to balance soundness and zero knowledge. Indeed, one could imagine "unrolling" this IOP into a PCP in the same way as for sumcheck itself: writing down, for each $\alpha \in \mathbb{F}$, a sumcheck PCP Π_α for $\alpha P + R$. This preserves soundness, but completely breaks zero knowledge: since the sumcheck PCP is a linear function of the underlying polynomial, given any *two* $\Pi_\alpha, \Pi_{\alpha'}$ for distinct α, α', we can recover the sumcheck PCP for P as $(\Pi_\alpha - \Pi_{\alpha'})/(\alpha - \alpha')$.

On the other hand, if we have the prover send only *one* Π_α, then soundness is lost, as the prover can easily cheat by sending R that does not sum to zero. We could attempt to fix the soundness issue by having the prover additionally *prove*, via another sumcheck PCP Π_R, that $\sum_{\vec{a}} R(\vec{a}) = 0$. While this would restore soundness, we once again lose zero knowledge, as the sumcheck PCP for P can be recovered as $(\Pi_\alpha - \Pi_R)/\alpha$.

2.1 Structure Versus Randomness

From the above discussion, we see that the central obstacle to obtaining a zero knowledge PCP is that the prover can break soundness by sending some R with $\sum_{\vec{a}} R(\vec{a}) \neq 0$, and known methods for detecting this strategy (without interaction) break zero knowledge.

Our approach is to *prevent* this malicious prover strategy by choosing a polynomial R so that $\sum_{\vec{a}} R(\vec{a}) = 0$ *by definition*. Of course, we must take care to ensure that R still hides certain information about P. But what kind of information? We observe that, without loss of generality, we can view the sumcheck PCP for $P + R$ as an oracle Π such that

$$\Pi(c_1, \ldots, c_i) = \sum_{\vec{a}\in\{0,1\}^{m-i}} (P + R)(c_1, \ldots, c_i, a_{i+1}, \ldots, a_m) ,$$

for all $i \in [m]$ and $c_1, \ldots, c_i \in \mathbb{F}$. Computing the "subcube sum" $\sum_{\vec{a}\in\{0,1\}^{m-i}} P(c_1, \ldots, c_i, a_{i+1}, \ldots, a_m)$ requires 2^{m-i} queries to P. Hence, in order to hope to simulate, we must ensure that R hides such subcube sums for any $1 \leq i < m - O(\log m)$. So, at a minimum, it should be the case that for all such i, and all $c_1, \ldots, c_i \in \mathbb{F}$, the subcube sum $\sum_{\vec{a}\in\{0,1\}^{m-i}} R(c_1, \ldots, c_i, a_{i+1}, \ldots, a_m)$ is marginally uniform.

In short: (1) for soundness, we need that the *full* sum of the masking polynomial R over the hypercube $\{0,1\}^n$ is fixed, and (2) for zero knowledge, other than sum over the hypercube, we would like R to be distributed as random as possible.

We start by ensuring that any *partial* subcube sum of R is distributed uniformly at random. Our key observation here is that while the full sum is *invariant* under any reordering of the variables $X_1, \ldots, X_m$, any nontrivial subcube sum ($i < m$) is *not*. This leads us to the following choice for the masking polynomial:

$$R(X_1, \ldots, X_m) := Q(X_1, X_2, \ldots, X_m) - Q(X_m, X_{m-1}, \ldots, X_1) ,$$

where Q is a uniformly random m-variate polynomial of individual degree d.

Clearly, by the permutation invariance of the complete sum, $\sum_{\vec{a}\in\{0,1\}^m} R(\vec{a}) = 0$. On the other hand, consider the special case of a subcube sum of R on the hypercube, i.e., for $(c_1, \ldots, c_i) \in \{0,1\}^i$:

$$\sum_{\vec{a}\in\{0,1\}^{m-i}} Q(c_1, \ldots, c_i, a_{i+1}, \ldots, a_m) - \sum_{\vec{a}\in\{0,1\}^{m-i}} Q(a_m, \ldots, a_{i+1}, c_i, \ldots, c_1) .$$

This expression can only be identically zero for all Q if $\{c_1\} \times \cdots \times \{c_i\} \times \{0,1\}^{m-i} = \{0,1\}^{m-i} \times \{c_i\} \times \cdots \times \{c_1\}$, which is clearly only possible if $i \in \{0, m\}$. By linearity, if this subcube sum is not identically zero for all Q, then it is uniform when Q is uniform.

Still, compared to the IOP case, the masking polynomial R has extra structure: it satisfies an "antisymmetry" $R(\alpha_1, \ldots, \alpha_m) = -R(\alpha_m, \ldots, \alpha_1)$ for all $(\alpha_1, \ldots, \alpha_m) \in \mathbb{F}^m$. This structure means that $P + R$ is no longer uniformly random and may leak nontrivial information about P.

To mitigate this, we modify R to make it "as random as possible" while preserving antisymmetry on $\{0,1\}^m$ and low-degree structure over $\mathbb{F}^m$. Our final choice of R is as follows:

$$R(\vec{X}) = Q(\vec{X}) - Q(\vec{X}_{\text{rev}}) + \sum_{i=1}^{m} X_i(1 - X_i)T_i(\vec{X}) , \qquad (1)$$

where Q is as before, $\vec{X}_{\text{rev}} := (X_m, \ldots, X_1)$, and each T_i is a uniformly random m-variate polynomial where $\deg_{X_i}(T_i) = d - 2$ and $\deg_{X_j}(T_i) = d$ for $j \neq i$. What is this extra term? It was observed by [11], via the combinatorial nullstellensatz [4], that $Z(\vec{X}) := \sum_{i=1}^m X_i(1 - X_i)T_i(\vec{X})$ is a uniformly random polynomial subject to the condition that, for all $\vec{a} \in \{0,1\}^m$, $Z(\vec{a}) = 0$. Adding this to R retains the antisymmetric structure on the hypercube, but has the effect of "masking out" any structure that appears outside of $\{0,1\}^m$; in particular, R is no longer antisymmetric over all of $\mathbb{F}^m$.

Key challenge: local simulation of combinatorial and algebraic structure. Showing the completeness and soundness of the PCP obtained via the foregoing approach is straightforward. Hence, it remains to prove that the perfect zero-knowledge condition holds.

Starting with the trivial case, note that our observation about subcube sums in the discussion above provides a simple strategy for simulating any *single* query $(c_1, \ldots, c_i)$ *within the hypercube* $\{0,1\}^i$ to Π. Alas, this clearly does not suffice, as even the honest sumcheck

verifier makes multiple queries to the proof, for $(c_1, \ldots, c_i) \notin \{0,1\}^i$, Hence, we must simulate multiple queries beyond the hypercube.

Indeed, the simulation of multiple queries to the low-degree extension of the hypercube is where the key challenge arises. More specifically, the difficulty is that our masking polynomial R has both *algebraic* structure arising from the polynomial degree bound and *combinatorial* structure arising from the antisymmeric reordering of the variables. In order to simulate, we must not only understand both types of structure, but crucially, also *how they interact.*

In the remainder of this proof overview, we explain how we design an efficient simulator that can simulate responses to any number of queries to Π (the sumcheck PCP for $P + R$), over the entire space $\mathbb{F}^{\leq m}$. Towards this end, we introduce the notion of *locally-simulatable encodings*.

Loosely speaking, locally-simulatable encodings are randomised encodings in which every local view of the encoding can be efficiently sampled from a local view of the message. More precisely, we say that a randomised encoding function $\mathsf{ENC}\colon (H \to \mathbb{F}) \to (D \to \mathbb{F})$ is locally simulatable if there is an algorithm $\mathcal{S}$ which, given oracle access to a message $m\colon H \to \mathbb{F}$ and a set $S \subseteq D$, samples from the distribution of $\mathsf{ENC}(m)|_S \subseteq \mathbb{F}^S$ in time polynomial in $|S|$ (which may be much less than $|H|$).

The notion of locally-simulatable encodings is tightly connected to ZKPCPs: we can view the mapping from instances to corresponding PCPs as a randomised encoding. A local simulator for this encoding is a zero knowledge simulator for the PCP system.[1] In the following sections, we will provide an overview of the proofs that the combinatorial antisymmetric structure of our masking, the algebraic structure of the Reed-Muller code, and their augmentations with subcube sums that arise in the sumcheck protocol admit locally simulatable encodings.

2.2 Combinatorial Structure of Antisymmetric Functions

As discussed above, the response to any *single* $(c_1, \ldots, c_i) \in \{0,1\}^i$ to Π can be perfectly simulated. In this section, we explain how to simulate any number of queries to Π, provided that those queries lie in the set $\{0,1\}^{\leq m} := \cup_{i=1}^{m} \{0,1\}^i$. Note that evaluated over the hypercube, the last term of Eq. (1) cancels, hence we can think of the masking polynomials as $R(\vec{X}) = Q(\vec{X}) - Q(\vec{X}_{\mathrm{rev}})$, where $\vec{X}_{\mathrm{rev}} := (X_m, \ldots, X_1)$.

Observe that, when restricted to $\{0,1\}^m$, the distribution of R is well structured; namely, it is a uniformly random element of the vector space AntiSym of "antisymmetric functions"[2], i.e., functions $f\colon \{0,1\}^m \to \mathbb{F}$ such that for all $\vec{c} \in \{0,1\}^m$, $f(\vec{c}) = -f(\vec{c}_{\mathrm{rev}})$, where $\vec{c}_{\mathrm{rev}} = (c_m, \ldots, c_1)$ is the reverse of $\vec{c}$. We can view the restriction of Π to $\{0,1\}^{\leq m}$ as the output of a randomised encoding function $\mathsf{ENC}_{\Sigma\mathrm{AntiSym}}\colon (\{0,1\}^m \to \mathbb{F}) \to (\{0,1\}^m \to \mathbb{F})$, applied to the restriction of P to $\{0,1\}^m$. The function $\mathsf{ENC}_{\Sigma\mathrm{AntiSym}}$ has the following description.

$\underline{\mathsf{ENC}_{\Sigma\mathrm{AntiSym}}(f)\colon}$

(1) Choose a uniformly random $r \in \mathrm{AntiSym}$.
(2) Output the function $\Sigma[f + r]$, given by

$$\Sigma[f+r](c_1, \ldots, c_i) := \sum_{\vec{a} \in \{0,1\}^{m-i}} (f+r)(c_1, \ldots, c_i, a_{i+1}, \ldots, a_m)\,, \tag{2}$$

for all $i \in [m]$; i.e., the function $f + r$, augmented with all of its subcube sums.

In this perspective, our task is now to provide a local simulator for $\mathsf{ENC}_{\Sigma\mathrm{AntiSym}}$. That is, for a query set $S \subseteq \{0,1\}^{\leq m}$, we want to efficiently simulate $\Sigma[f + r]|_S$ for a uniformly random $r \in \Sigma\mathrm{AntiSym}$, given oracle access to f.

Let $\Sigma\mathrm{AntiSym}|_S$ denote the vector space $\{\Sigma[r]|_S : r \in \mathrm{AntiSym}\}$, and let B be a basis for its dual code $(\Sigma\mathrm{AntiSym}|_S)^{\perp}$; that is, $v \in \Sigma\mathrm{AntiSym}|_S$ if and only if $Bv = 0$. Then, by linearity, $\Sigma[f + r]|_S$ is distributed as a uniformly random vector $w\colon S \to \mathbb{F}$ such that $Bw = B(\Sigma[f]|_S)$.

With this in mind, our local simulator $\mathcal{S}$ could work as follows.

$\underline{\mathcal{S}^f(S)\colon}$

(1) Compute a basis B for $(\Sigma\mathrm{AntiSym}|_S)^{\perp}$.
(2) Compute $y = B(\Sigma[f]|_S)$ by querying f as necessary.
(3) Output a random w such that $Bw = y$.

Correctness is straightforward. The only potential issue is efficiency: evaluating $\Sigma[f]$ at a point in S may require exponentially many queries to f. Note, however, that if the column of B corresponding to $\vec{c} := (c_1, \ldots, c_i)$ is all zero, then we do not need to know $\Sigma[f](c_1, \ldots, c_i)$ to compute $B(\Sigma[f]|_S)$. Hence, for efficiency, it would suffice to (efficiently) find a basis B for $(\Sigma\mathrm{AntiSym}|_S)^{\perp}$ such that $\sum_{\vec{c} \in \mathrm{supp}(B)} 2^{m-\ell(\vec{c})} = \mathrm{poly}(|S|)$, where $\mathrm{supp}(B)$ is the set of nonzero columns of B and $\ell(\overrightarrow{()}c) = i$ for $c \in \{0,1\}^i$.

Does such a basis exist? Not exactly: for example, if $S = \{(0), (1)\}$, then $\mathrm{supp}(B) = \{(0), (1)\}$, since $\Sigma[r](0) + \Sigma[r](1) = \sum_{\vec{a} \in \{0,1\}^{m-1}} r(0, \vec{a}) + \sum_{\vec{a} \in \{0,1\}^{m-1}} r(1, \vec{a}) = 0$ for all $r \in \mathrm{AntiSym}$. However, this counterexample does not actually cause a problem for zero knowledge, since for this basis B, it holds that $B(\Sigma[P]|_S) = \sum_{\vec{a} \in \{0,1\}^m} P(\vec{a}) = \gamma$, which is part of the input to the problem. Our key technical result in this section is that all possible counterexamples are essentially of this form.

Lemma 1. *There is a polynomial p such that, for any prefix-free*[3] *$S \subseteq \{0,1\}^{\leq m}$, there exists a basis B of $(\Sigma\mathrm{AntiSym}|_S)^{\perp}$ where each row b_i of B is a 0-1 vector, and letting $T(b_i) := \{(\vec{c}, \vec{a}) : b_i(\vec{c}) = 1, \vec{a} \in \{0,1\}^{m-|\vec{c}|}\}$, either*

$$|T(b_i)| \leq p(|S|) \qquad \text{or} \qquad |T(b_i)| \geq 2^m - p(|S|)\,.$$

Moreover, B can be efficiently computed from S.

For the purposes of this overview, we will assume that S is indeed prefix-free; in the full proof we show that this holds without loss of generality.

To gain some intuition for this result, let us consider the 2-dimensional case of antisymmetric functions $A\colon [n]^2 \to \mathbb{F}$; i.e., $n \times n$ antisymmetric matrices A over $\mathbb{F}$. Suppose that $X \subseteq [n] \times [n]$ is a set consisting of r full rows and t individual entries (so $|X| = rn + t$) such that for all antisymmetric A, it holds that $\sum_{(i,j) \in X} a_{ij} = 0$.

[1] To handle malicious verifiers that make adaptive queries, we will actually require a stronger notion of local simulation that allows for conditional sampling; see ?? for details.

[2] This is terminology that we introduce, which is distinct from the usual notion of antisymmetry for polynomials.

[3] A set $S \subseteq \{0,1\}^{\leq m}$ is *prefix-free* if for any $\vec{c}_1, \vec{c}_2 \in S$, if $\vec{c}_1$ is a prefix of $\vec{c}_2$ then $\vec{c}_1 = \vec{c}_2$.

$$\begin{pmatrix} 0 & a_{12} & a_{13} & a_{14} & a_{15} \\ -a_{12} & 0 & a_{23} & a_{24} & a_{25} \\ -a_{13} & -a_{23} & 0 & a_{34} & a_{35} \\ -a_{14} & -a_{24} & -a_{34} & 0 & a_{45} \\ -a_{15} & -a_{25} & -a_{35} & -a_{45} & 0 \end{pmatrix}$$

Figure 1: A general antisymmetric matrix ($n = 5$), with an element of the dual highlighted ($r = 2, t = 7$).

The latter condition implies that the indicator matrix for X must be symmetric; one possible choice of X is highlighted in Fig. 1. It is straightforward to show that this symmetry implies $t \geq r \cdot (n - r)$, from which we obtain the pair of inequalities

$$r \leq \frac{1}{2} \cdot n \left(1 - \sqrt{1 - \frac{4t}{n^2}}\right) \approx \frac{t}{n} + \frac{t^2}{n^3}, \quad \text{or}$$
$$r \geq \frac{1}{2} \cdot n \left(1 + \sqrt{1 - \frac{4t}{n^2}}\right) \approx n - \frac{t}{n} - \frac{t^2}{n^3}.$$

This in turn yields appropriate bounds on $|X|$: $|X| \lesssim 2t + t^2/n^2$ or $|X| \gtrsim n^2 - t^2/n^2$. The proof of Lemma 1 generalises this approach to higher dimensions.

Lemma 1 suggests the following modified local simulator, which we additionally provide with the total sum γ:

$\underline{\mathcal{S}^f_{\text{AntiSym}}(S; \gamma)}$:
(1) Compute a basis B for $(\Sigma\text{AntiSym}|_S)^\perp$ as in Lemma 1.
(2) For each row b_i of B:
  (a) if $|T(b_i)| \leq 2^{m-1}$, compute $y_i = \sum_{\vec{x} \in T(b_i)} f(\vec{x})$;
  (b) otherwise, compute $y_i = \gamma - \sum_{\vec{x} \in \{0,1\}^m \setminus T(b_i)} f(\vec{x})$.
(3) Output a random w such that $Bw = y$.

Clearly, the number of queries to f required is $\min(|T(b_i)|, 2^m - |T(b_i)|)$, which is $\text{poly}(|S|)$ by Lemma 1; hence the overall running time of the simulator is $\text{poly}(|S|)$.

2.3 Local Simulation of Random Low-Degree Extensions

In the previous section we addressed the *combinatorial* problem of simulating queries to Π that lie in the set $\{0,1\}^{\leq m}$, which exhibits antisymmetric strucutre. In this section we consider the *algebraic* problem of simulating queries to Π that also lie outside of the hypercube (i.e., general point queries), which exhibits pseudorandom structure. We then bring these two parts together in Section 2.4.

Recall that our choice of R is

$$R(\vec{X}) = Q(\vec{X}) - Q(\vec{X}_{\text{rev}}) + \sum_{i=1}^{m} X_i(1 - X_i)T_i(\vec{X}),$$

where Q is as before, $\vec{X}_{\text{rev}} := (X_m, \ldots, X_1)$, and each T_i is a uniformly random m-variate polynomial where $\deg_{X_i}(T_i) = d - 2$ and $\deg_{X_j}(T_i) = d$ for $j \neq i$.

Denote by $\text{RM}[\mathbb{F}, m, d] \subseteq \mathbb{F}^m \to \mathbb{F}$ the *Reed–Muller* code of evaluations of m-variate polynomials of individual degree d over $\mathbb{F}$. For a function $f\colon \{0,1\}^m \to \mathbb{F}$, denote by $\text{LD}[f, d] := \{\hat{f} \in \text{RM}[\mathbb{F}, m, d] : \hat{f}(x) = f(x)\ \forall \vec{x} \in \{0,1\}^m\}$ the set of *degree-d extensions* of f. With the above modification, we can describe the sumcheck PCP for $P + R$, restricted to $\mathbb{F}^m$, as the output of the following randomised encoding:

$\underline{\text{ENC}_{\text{RMAntiSym}}(P)}$:
(1) Sample a random antisymmetric function $f\colon \{0,1\}^m \to \mathbb{F}$.
(2) Sample R uniformly at random from $\text{LD}[f, d]$.
(3) Output $P + R$.

Our problem now becomes to design a local simulator for $\text{ENC}_{\text{RMAntiSym}}$.

We will do this by solving a much more general problem: *we prove that random low-degree extensions are locally simulatable.* More precisely, let $\text{ENC}^d_{\text{RM}}(f)$ be the encoding function that outputs a uniformly random element of $\text{LD}[f, d]$. We prove that ENC^d_{RM} has a local simulator $\mathcal{S}^d_{\text{RM}}$ for any $d \geq 2$. We can then easily obtain a simulator for $\text{ENC}_{\text{RMAntiSym}}$ by composing $\mathcal{S}_{\text{RM}}$ with the local simulator for AntiSym described in the previous section.

Our starting point. We will build on the "stateful emulator" that was constructed in [11] (for a particular oracle model that they introduced), which can be conceptualised as an *inefficient* local simulator for ENC_{RM}. To describe it, we first introduce some notation.

For $w \in \{0,1\}^m$, we denote by δ_w the unique multilinear polynomial satisfying $\delta_w(w) = 1$ and $\delta_w(x) = 0$ for all $x \in \{0,1\}^m \setminus \{w\}$. For a set $S \subseteq \mathbb{F}^m$, we say that w is *S-good* if there exists an m-variate polynomial q_w of individual degree at most d such that (i) $q_w(x) = 0$ for every $x \in \{0,1\}^m \setminus \{w\}$; (ii) $q_w(z) = 0$ for every $z \in S$; and (iii) $q_w(w) = 1$. We say that w is *S-bad* if it is not S-good. Intuitively, when $\hat{f}$ is a uniformly random low-degree extension of f, $\hat{f}|_S$ only conveys information about $f(w)$ for S-bad points w. For example, any point in $\{0,1\}^m \cap S$ is trivially S-bad. Less trivially, if S consists of sufficiently many points on a curve passing through $w \in \{0,1\}^m$, then w may be S-bad even if $\{0,1\}^m \cap S$ is empty.

The simulator of Chen et al. works roughly as follows:[4]

$\underline{\mathcal{S}^f_{\text{RM}}(S)}$:
(1) Compute the set of S-bad points W.
(2) Query $f(w)$ at all $w \in W$, sample a uniformly random polynomial P such that $P(w) = f(w)$ for all $w \in W$, and output $P|_S$.

Chen et al. prove that $\mathcal{S}_{\text{RM}}$ is correct, and moreover that it is *query-efficient*, provided that $d \geq 2$: the number of queries it makes to f (equal to $|W|$) is at most $|S|$ (this follows from [1, Lemma 4.3]).

However, they leave open the question of whether $\mathcal{S}_{\text{RM}}$ can be made *computationally* efficient. We note that **??** 2 can be achieved efficiently using an algorithm of [9]. The issue is in **??** 1; namely, it is not clear how to compute the set W from S. Indeed, at first glance this appears to be hopeless: naively, computing whether a single w is S-bad requires solving an exponentially large linear system, and there are 2^m possible choices for w. Nonetheless, we will present an efficient algorithm which, given S, outputs a list of all S-bad points.[5]

[4] Their analysis requires sampling P in a different (and more complicated) way.
[5] Chen et al. follow a different route: roughly, they show that because their f is *extremely sparse*, it is very unlikely that $f(w) \neq 0$ for any $w \in W \setminus S$, and so for simulation one can pretend that f is zero on $\{0,1\}^m \setminus S$. This does not work for us for two reasons:

Step 1: solving the decision problem. Our first step is to consider the *decision* variant of this problem: given a set S, does there *exist* an S-bad w? Our key insight here is that we can relate the existence of an S-bad w to the dimension of the vector space $\mathsf{LD}[Z,d]|_S$, where Z is the constant zero function. We show that all w are S-good if and only if $\mathsf{LD}[Z,d]|_S = \mathrm{RM}[\mathbb{F},m,d]|_S$. Indeed, if every w is S-good, then for any degree-d polynomial P, the polynomial $P - \sum_{w\in\{0,1\}^m} P(w)\cdot q_w$ is a degree-d extension of Z that agrees with P on S. On the other hand, if $\mathsf{LD}[Z,d]|_S = \mathrm{RM}[\mathbb{F},m,d]|_S$, then for every w, there is a $\hat{Z}_w \in \mathsf{LD}[Z,d]$ that agrees with δ_w on S; hence we can take $q_w = \delta_w - \hat{Z}_w$, and so w is S-good.

Hence to solve the decision problem it suffices to compute (the dimensions of) $\mathrm{RM}[\mathbb{F},m,d]|_S$ and $\mathsf{LD}[Z,d]|_S$. The former can be achieved efficiently (in time $\mathrm{poly}(|S|,m,d,\log|\mathbb{F}|)$) using the *succinct constraint detector* for Reed–Muller discovered by [9].

To compute the latter, we build on ideas introduced in [11]. As discussed above, in that work the authors observe that to sample a vector $v \leftarrow \mathsf{LD}[Z,d]|_S$, it suffices to (lazily) sample random polynomials T_i, $i\in[m]$, of the appropriate degrees, and then set $v(\vec{\alpha}) := \sum_{i=1}^m \alpha_i(1-\alpha_i)T_i(\vec{\alpha})$ for each $\vec{\alpha}\in S$. Similarly, we show how to compute a basis for $\mathsf{LD}[Z,d]|_S$ by combining bases for the subspaces $\mathcal{P}_i := \{T_i|_S \in \mathrm{RM}[\mathbb{F},d,m]|_S : \deg_{X_i}(T_i) = d-2\}$, for $i\in[m]$. These bases can also be computed using the succinct constraint detector for Reed–Muller; see **??**.

Step 2: search-to-decision reduction. Next, we will build upon the techniques we developed in Step 1 to solve the original search problem. A natural strategy is to employ a binary search: for each $b\in\{0,1\}$, test if $\{b\}\times\{0,1\}^{m-1}$ contains any S-bad points, and recurse if it does. Since the number of S-bad points is bounded by $|S|$, the recursion will terminate quickly. It therefore suffices to give an efficient algorithm that, given a set A of the form $\{a_1\}\times\cdots\times\{a_i\}\times\{0,1\}^{m-i}$, determines whether A contains any S-bad point.

For $A\subseteq\{0,1\}^m$, let $\mathsf{LD}[Z_A,d]|_S := \{P\in\mathrm{RM}[\mathbb{F},m,d] : P(\vec{x}) = 0\,\forall\vec{x}\in A\}$. Our algorithm for computing a basis for $\mathsf{LD}[Z,d]|_S$ can be straightforwardly extended to compute a basis for $\mathsf{LD}[Z_A,d]|_S$. We can also extend the reasoning from Step 1 to show that if every $w\in A$ is S-good, then $\mathsf{LD}[Z_A,d]|_S = \mathrm{RM}[\mathbb{F},m,d]|_S$. Unfortunately, the converse does not hold: it may be that A contains S-bad points but $\mathsf{LD}[Z_A,d]|_S = \mathrm{RM}[\mathbb{F},m,d]|_S$. This can happen, for example, if S contains points on a curve passing through both $w\in A$ and $w'\in\{0,1\}^m\setminus A$. In particular, the argument from Step 1 fails: for $\hat{Z}_w\in\mathsf{LD}[Z_A,d]$, $\delta_w - \hat{Z}_w$ is not necessarily zero on $\{0,1\}^m\setminus A$.

To obtain our final algorithm, we will instead exploit the algebraic relationship between $\mathrm{RM}[\mathbb{F},m,d]$ and its subcode $\mathrm{RM}[\mathbb{F},m,d-1]$ to derive a *sufficient* condition for every $w\in A$ being S-good. Observe that if $\mathsf{LD}[Z_A,d-1]|_S = \mathrm{RM}[\mathbb{F},m,d-1]|_S$, then for every $w\in A$ there is a $\hat{Z}_w\in\mathsf{LD}[Z_A,d-1]$ such that $p_w := \delta_w - \hat{Z}_w$ satisfies $p_w(w)=1$ and $p_w(\vec{\alpha})=0$ for all $\vec{\alpha}\in S$. It follows that all $w\in A$ are S-good, as we can set $q_w = p_w\cdot\delta_w$.

Thus, our efficient test for whether A contains any S-bad point is as follows: compute bases for $\mathsf{LD}[Z_A,d-1]|_S$ and $\mathrm{RM}[\mathbb{F},m,d-1]|_S$, and output "no" if they are of the same dimension; otherwise output "maybe". By the above discussion, this test does not have any false negatives; however, there may be false positives. Provided there are not too many false positives, this does not cause a problem (we can either include them in W or perform an extra test to filter them out). We bound the number of false positives by noting that if the test says "maybe", this in fact means that there exists an "S-bad" point in A with respect to polynomials of degree $d-1$. By [1], provided $d\geq 2$, there are at most $|S|$ such points.

2.4 Local Simulation of Subcube Sums of LDES

So far, we have built a simulator that can answer arbitrary queries to our sumcheck PCP Π, provided those queries lie within the set $\{0,1\}^{\leq m}\cup\mathbb{F}^m$, handling the combinatorial structure on the hypercube and the algebraic, pseudorandom algebraic structure outside of it. In this final part of the overview, we outline how this simulator can be extended to handle all queries; i.e., queries in the set $\mathbb{F}^{\leq m}$.

To do this, it suffices to show that the code $\Sigma\mathrm{RM}$ admits a locally-simulatable encoding, where $\Sigma\mathrm{RM}[\mathbb{F},m,d] = \{\Sigma[P] : P\in\mathrm{RM}[\mathbb{F},m,d]\}$ and $\Sigma[P]$ is defined as in Eq. (2) (where $(c_1,\ldots,c_i)$ now ranges over $\mathbb{F}^i$). Specifically, we show that the following encoding function is locally simulatable:

$\underline{\mathsf{ENC}^d_{\Sigma\mathrm{RM}}(\Sigma[f]):}$

(1) Sample $\hat{f}$ uniformly at random from $\mathsf{LD}[f,d]$.
(2) Output $\Sigma[\hat{f}]$.

Note that the *message* (input to $\mathsf{ENC}_{\Sigma\mathrm{RM}}$) is $\Sigma[f]$, even though $\mathsf{ENC}_{\Sigma\mathrm{RM}}$ operates only on f. This is necessary for local simulation, since individual locations in $\Sigma[\hat{f}]$ depend on partial sums of f which cannot be computed from few queries to f itself.

We would like to follow the strategy from the previous section: given a set $S\subseteq\mathbb{F}^{\leq m}$ on which we want to simulate $\mathsf{ENC}_{\Sigma\mathrm{RM}}(\Sigma[f])$, compute the set of all "bad" points $W\subseteq\{0,1\}^{\leq m}$, i.e., $w\in W$ if $\mathsf{ENC}_{\Sigma\mathrm{RM}}(\Sigma[f])|_S$ conveys information about $\Sigma[f](w)$. Unfortunately, unlike in the "plain" Reed–Muller case, it is not clear how to even bound the *size* of W, let alone compute it.

The issue here is that a single evaluation of $\Sigma[\hat{f}](\vec{\alpha})$ for $\vec{\alpha}\in\mathbb{F}^i$ depends on $\hat{f}(\vec{\beta})$ for every point $\vec{\beta}\in\vec{\alpha}\times\{0,1\}^{m-i}$. Naively applying the lemma of [1] to this set of points yields a set $W\subseteq\{0,1\}^m$ of size 2^{m-i}. To bound $|W|$ by a polynomial, we must therefore crucially make use of the fact that making a few queries to $\Sigma[\hat{f}]$ can reveal only a few (possibly large) *partial sums* of $\hat{f}$ — which we can hope to deduce from a small number of queries to $\Sigma[f]$.

To achieve this we will give a *decomposition* of $\Sigma\mathrm{RM}$ as a sequence of RM codes on different numbers of variables, "tied together" with constraints that enforce summation structure. In more detail, let $T\subseteq\mathbb{F}^{\leq m}$ be a set with the following special structure, which we call "closed": if $(s_1,\ldots,s_i)\in S$, then its "parent" $(s_1,\ldots,s_{i-1})$ and its "siblings" $(s_1,\ldots,s_{i-1},0)$ and $(s_1,\ldots,s_{i-1},1)$ are also in S. The set $\{0,1\}^{\leq m}$ is closed; it is also easy to see that any set T can be made closed by adding at most $3m|T|$ points. We prove the following theorem about the structure of restrictions of $\Sigma\mathrm{RM}$ to closed sets T:

Theorem 2.1 (Informally stated, see **??**). *For any closed set T, any constraint $z\in(\Sigma\mathrm{RM}[\mathbb{F},m,d]|_T)^\perp$ lies in the span of the following two types of constraint:*

first, a random antisymmetric function is not sparse with overwhelming probability, and second, this strategy introduces a small statistical error in simulation, which would spoil perfect zero knowledge.

- summation constraints, *which ensure that* $\Sigma[\hat{f}](t_1, \ldots, t_i) = \Sigma[\hat{f}](t_1, \ldots, t_i, 0) + \Sigma[\hat{f}](t_1, \ldots, t_i, 1)$ *for* $(t_1, \ldots, t_i) \in T$, $i < m$; *and*
- low-degree constraints, *which are given by, for each* i, *the* i*-variate Reed–Muller constraints on the set* $T_i := T \cap \mathbb{F}^i$*; i.e.,* $(\mathrm{RM}[\mathbb{F}, i, d]|_{T_i})^{\perp}$.

If we take $T = \{0,1\}^{\leq m} \cup S$, then $T_i = \{0,1\}^i \cup (S \cap \mathbb{F}^i)$ (assuming S is closed). Using this decomposition, we show that we can take as the "bad" set $W := \bigcup_{i=1}^{m} W_i$, where W_i is the set of $(S \cap \mathbb{F}^i)$-bad points in $\mathrm{RM}[\mathbb{F}, i, d]$. Each W_i can be computed efficiently using the algorithm described in Section 2.3.

We conclude by giving some brief intuition on how we prove Theorem 2.1. We show that any constraint z supported on $T_i \cup T_{i+1} \cup \cdots \cup T_m$ can be "flattened" into a constraint supported on T_i only. Such constraints belong to $(\mathrm{RM}[\mathbb{F}, i, d]|_{T_i})^{\perp}$. We then use a dimension-counting argument to show that these, augmented with the summation constraints, span $(\Sigma\mathrm{RM}[\mathbb{F}, m, d]|_T)^{\perp}$.

ACKNOWLEDGEMENTS

The authors thank Alessandro Chiesa, Hendrik Waldner and Arantxa Zapico for helpful discussions. TG is supported by UKRI Future Leaders Fellowship MR/S031545/1, EPSRC New Horizons Grant EP/X018180/1, and EPSRC RoaRQ Grant EP/W032635/1.

REFERENCES

[1] Aaronson, S., and Wigderson, A. Algebrization: A new barrier in complexity theory. *ACM Transactions on Computation Theory 1*, 1 (2009), 2:1–2:54.
[2] Aharonov, D., Arad, I., and Vidick, T. Guest column: the quantum pcp conjecture. *Acm sigact news 44*, 2 (2013), 47–79.
[3] Aiello, W., and Håstad, J. Statistical zero-knowledge languages can be recognized in two rounds. *Journal of Computer and System Sciences 42*, 3 (1991), 327–345. Preliminary version appeared in FOCS '87.
[4] Alon, N. Combinatorial nullstellensatz. *Combinatorics, Probability and Computing 8* (1999), 7–29.
[5] Arora, S., and Barak, B. *Computational Complexity: A Modern Approach*, 1st ed. Cambridge University Press, New York, NY, USA, 2009.
[6] Arora, S., Lund, C., Motwani, R., Sudan, M., and Szegedy, M. Proof verification and hardness of approximation problems. In *Proceedings of the 33rd Annual Symposium on Foundations of Computer Science* (1992), pp. 14–23.
[7] Arora, S., and Sudan, M. Improved low-degree testing and its applications. *Combinatorica 23*, 3 (2003), 365–426. Preliminary version appeared in STOC '97.
[8] Ben-Or, M., Goldwasser, S., Kilian, J., and Wigderson, A. Multi-prover interactive proofs: how to remove intractability assumptions. In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing* (1988), STOC '88, pp. 113–131.
[9] Ben-Sasson, E., Chiesa, A., Forbes, M. A., Gabizon, A., Riabzev, M., and Spooner, N. Zero knowledge protocols from succinct constraint detection. In *Proceedings of the 15th Theory of Cryptography Conference* (2017), TCC '17, pp. 172–206.
[10] Bouland, A., Chen, L., Holden, D., Thaler, J., and Vasudevan, P. N. On the power of statistical zero knowledge. *SIAM J. Comput. 49*, 4 (2020).
[11] Chen, M., Chiesa, A., Gur, T., O'Connor, J., and Spooner, N. Proof-carrying data from arithmetized random oracles. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques* (2023), EUROCRYPT '23, pp. 379–404.
[12] Chiesa, A., Forbes, M. A., Gur, T., and Spooner, N. Spatial isolation implies zero knowledge even in a quantum world. *Journal of the ACM 69*, 2 (2022), 1–44.
[13] Chiesa, A., Forbes, M. A., and Spooner, N. A zero knowledge sumcheck and its applications, 2017.
[14] Dinur, I. The PCP theorem by gap amplification. *Journal of the ACM 54*, 3 (2007), 12.
[15] Dwork, C., Feige, U., Kilian, J., Naor, M., and Safra, S. Low communication 2-prover zero-knowledge proofs for NP. In *Proceedings of the 11th Annual International Cryptology Conference* (1992), CRYPTO '92, pp. 215–227.
[16] Fortnow, L. The complexity of perfect zero-knowledge (extended abstract). In *Proceedings of the 19th Annual ACM Symposium on Theory of Computing* (1987), STOC '87, pp. 204–209.
[17] Goldreich, O., Micali, S., and Wigderson, A. Proofs that yield nothing but their validity or all languages in NP have zero-knowledge proof systems. *Journal of the ACM 38*, 3 (1991), 691–729. Preliminary version appeared in FOCS '86.
[18] Goldwasser, S., Micali, S., and Rackoff, C. The knowledge complexity of interactive proof systems. *SIAM Journal on Computing 18*, 1 (1989), 186–208. Preliminary version appeared in STOC '85.
[19] Grilo, A. B., Slofstra, W., and Yuen, H. Perfect zero knowledge for quantum multiprover interactive proofs. In *60th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2019* (2019), IEEE Computer Society, pp. 611–635.
[20] Hazay, C., Venkitasubramaniam, M., and Weiss, M. Zk-pcps from leakage-resilient secret sharing. *J. Cryptol. 35*, 4 (2022), 23.
[21] Ishai, Y., Mahmoody, M., and Sahai, A. On efficient zero-knowledge PCPs. In *Proceedings of the 9th Theory of Cryptography Conference on Theory of Cryptography* (2012), TCC '12, pp. 151–168.
[22] Ishai, Y., and Weiss, M. Probabilistically checkable proofs of proximity with zero-knowledge. In *Proceedings of the 11th Theory of Cryptography Conference* (2014), TCC '14, pp. 121–145.
[23] Ishai, Y., Weiss, M., and Yang, G. Making the best of a leaky situation: Zero-knowledge PCPs from leakage-resilient circuits. In *Proceedings of the 13th Theory of Cryptography Conference* (2016), TCC '16-A, pp. 3–32.
[24] Juma, A., Kabanets, V., Rackoff, C., and Shpilka, A. The black-box query complexity of polynomial summation. *Computational Complexity 18*, 1 (2009), 59–79.
[25] Kalai, Y., and Raz, R. Interactive PCP. In *Proceedings of the 35th International Colloquium on Automata, Languages and Programming* (2008), ICALP '08, pp. 536–547.
[26] Kilian, J., Petrank, E., and Tardos, G. Probabilistically checkable proofs with zero knowledge. In *Proceedings of the 29th Annual ACM Symposium on Theory of Computing* (1997), STOC '97, pp. 496–505.
[27] Lapidot, D., and Shamir, A. A one-round, two-prover, zero-knowledge protocol for NP. *Combinatorica 15*, 2 (1995), 204–214.
[28] Lund, C., Fortnow, L., Karloff, H. J., and Nisan, N. Algebraic methods for interactive proof systems. *Journal of the ACM 39*, 4 (1992), 859–868.
[29] Thaler, J. Proofs, arguments, and zero-knowledge. *Foundations and Trends® in Privacy and Security 4*, 2–4 (2022), 117–660.
[30] Vadhan, S. P. *A study of statistical zero-knowledge proofs*. PhD thesis, Massachusetts Institute of Technology, 1999.
[31] Weiss, M. Shielding probabilistically checkable proofs: Zero-knowledge pcps from leakage resilience. *Entropy 24*, 7 (2022), 970.