Zero Knowledge Protocols from Succinct Constraint Detection

Eli Ben-Sasson[1], Alessandro Chiesa[2(✉)], Michael A. Forbes[3], Ariel Gabizon[4], Michael Riabzev[1], and Nicholas Spooner[2]

[1] Technion, Haifa, Israel
{eli,mriabzev}@cs.technion.ac.il
[2] UC Berkeley, Berkeley, USA
alexch@berkeley.edu, spooner@eecs.berkeley.edu
[3] University of Illinois Urbana-Champaign, Champaign, USA
miforbes@illinois.edu
[4] ZcashCo, Haifa, Israel
ariel@z.cash

Abstract. We study the problem of constructing proof systems that achieve both soundness and zero knowledge unconditionally (without relying on intractability assumptions). Known techniques for this goal are primarily *combinatorial*, despite the fact that constructions of interactive proofs (IPs) and probabilistically checkable proofs (PCPs) heavily rely on *algebraic* techniques to achieve their properties.

We present simple and natural modifications of well-known 'algebraic' IP and PCP protocols that achieve unconditional (perfect) zero knowledge in recently introduced models, overcoming limitations of known techniques.

- We modify the PCP of Ben-Sasson and Sudan [BS08] to obtain zero knowledge for **NEXP** in the model of Interactive Oracle Proofs [BCS16, RRR16], where the verifier, in each round, receives a PCP from the prover.
- We modify the IP of Lund et al. [LFKN92] to obtain zero knowledge for #**P** in the model of Interactive PCPs [KR08], where the verifier first receives a PCP from the prover and then interacts with him.

The simulators in our zero knowledge protocols rely on solving a problem that lies at the intersection of coding theory, linear algebra, and computational complexity, which we call the *succinct constraint detection* problem, and consists of detecting dual constraints with polynomial support size for codes of exponential block length. Our two results rely on solutions to this problem for fundamental classes of linear codes:

- An algorithm to detect constraints for Reed–Muller codes of exponential length. This algorithm exploits the Raz–Shpilka [RS05] deterministic polynomial identity testing algorithm, and shows, to our knowledge, a first connection of algebraic complexity theory with zero knowledge.

M.A. Forbes—Work conducted while at Stanford.

A. Gabizon—Work conducted while at Technion.

N. Spooner—Work conducted while at the University of Toronto.

Y. Kalai and L. Reyzin (Eds.): TCC 2017, Part II, LNCS 10678, pp. 172–206, 2017.
https://doi.org/10.1007/978-3-319-70503-3_6

– An algorithm to de tect constraints for PCPs of Proximity of Reed–Solomon codes [BS08] of exponential degree. This algorithm exploits the recursive structure of the PCPs of Proximity to show that small-support constraints are "locally" spanned by a small number of small-support constraints.

Keywords: Probabilistically checkable proofs · Interactive proofs · Sumcheck · Zero knowledge · Polynomial identity testing

1 Introduction

The study of interactive proofs (IPs) [BM88,GMR89] that unconditionally achieve zero knowledge [GMR89] has led to a rich theory, with connections well beyond zero knowledge. For example, the class of languages with statistical zero knowledge IPs, which we denote by **SZK-IP**, has complete problems that make no reference to either zero knowledge or interaction [SV03,GV99] and is closed under complement [Oka00,Vad99]. Despite the fact that all **PSPACE** languages have IPs [Sha92], **SZK-IP** is contained in $\mathbf{AM} \cap \mathbf{coAM}$, and thus **NP** is not in **SZK-IP** unless the polynomial hierarchy collapses [BHZ87]; one consequence is that Graph Non-Isomorphism is unlikely to be NP-complete. Moreover, constructing **SZK-IP** for a language is equivalent to constructing instance-dependent commitments for the language [IOS97,OV08], and has connections to other fundamental information-theoretic notions like randomized encodings [AR16,VV15] and secret-sharing schemes [VV15].

Unconditional zero knowledge in other models behaves very differently. Ben-Or et al. [BGKW88] introduced the model of multi-prover interactive proofs (MIPs) and showed that *all* such proofs can be made zero knowledge unconditionally. The analogous statement for IPs is equivalent to the existence of one-way functions, as shown by [GMR89,IY87,BGG+88] in one direction and by [Ost91,OW93] in the other (unless $\mathbf{BPP} = \mathbf{PSPACE}$, in which case the statement is trivial). Subsequent works not only established that all **NEXP** languages have MIPs [BFL91], but also led to formulating probabilistically checkable proofs (PCPs) and proving the celebrated PCP Theorem [FRS88,BFLS91, FGL+96,AS98,ALM+98], as well as constructing statistical zero knowledge PCPs [KPT97] and applying them to black-box cryptography [IMS12,IMSX15].

The theory of zero knowledge for these types of proofs, however, is not as rich as in the case of IPs. Most notably, known techniques to achieve zero knowledge MIPs or PCPs are limited, and come with caveats. Zero knowledge MIPs are obtained via complex generic transformations [BGKW88], assume the full power of the PCP Theorem [DFK+92], or support only languages in **NP** [LS95]. Zero knowledge PCPs are obtained via a construction that incurs polynomial blowups in proof length and requires the honest verifier to adaptively query the PCP [KPT97]. Alternative approaches are not known, despite attempts to find them. For example, [IWY16] apply PCPs to leakage-resilient circuits, obtaining PCPs for **NP** that do have a non-adaptive honest verifier but are only witness indistinguishable.

Even basic questions such as "are there zero-knowledge PCPs of quasilinear-size?" or "are there zero-knowledge PCPs with non-adaptive honest verifiers?" have remained frustratingly hard to answer, despite the fact the answers to these questions are well understood when removing the requirement of zero knowledge. This state of affairs begs the question of whether a richer theory about zero knowledge MIPs and PCPs could be established.

The current situation is that known techniques to achieve zero knowledge MIPs and PCPs are combinatorial, namely they make black-box use of an underlying MIP or PCP, despite the fact that most MIP and PCP constructions have a rich algebraic structure arising from the use of error correcting codes based on evaluations of low-degree polynomials. This separation is certainly an attractive feature, and perhaps even unsurprising: while error-correcting codes are designed to help recover information, zero knowledge proofs are designed to hide it.

Yet, a recent work by Ben-Sasson et al. [BCGV16] brings together linear error correcting codes and zero knowledge using an algebraic technique that we refer to as 'masking'. The paper introduces a "2-round PCP" for **NP** that unconditionally achieves zero knowledge and, nevertheless, has both quasilinear size and a non-adaptive honest verifier. Their work can be viewed not only as partial progress towards some of the open questions above, but also as studying the power of zero knowledge for a natural extension of PCPs ("multi-round PCPs" as discussed below) with its own motivations and applications [BCS16, RRR16, BCG+17].

The motivation of this work is to understand the power of algebraic tools, such as linear error correcting codes, for achieving zero knowledge unconditionally (without relying on intractability assumptions).

1.1 Results

We present new protocols that unconditionally achieve soundness and zero knowledge in recently suggested models that combine features of PCPs and IPs [KR08, BCS16, RRR16]. Our protocols consist of simple and natural modifications to well-known constructions: the PCP of Ben-Sasson and Sudan [BS08] and the IP for polynomial summation of Lund et al. [LFKN92]. By leveraging the linear codes used in these constructions, we reduce the problem of achieving zero knowledge to solving exponentially-large instances of a new linear-algebraic problem that we call *constraint detection*, which we believe to be of independent interest. We design efficient algorithms for solving this problem for notable linear code families, along the way exploiting connections to algebraic complexity theory and local views of linear codes. We now elaborate on the above by discussing each of our results.

Zero knowledge for non-deterministic exponential time. Two recent works [BCS16, RRR16] independently introduce and study the notion of an *interactive oracle proof* (IOP), which can be viewed as a "multi-round PCP". Informally, an IOP is an IP modified so that, whenever the prover sends to the verifier a message, the verifier does not have to read the message in full

but may probabilistically query it. Namely, in every round, the verifier sends the prover a message, and the prover replies with a PCP. IOPs enjoy better efficiency compared to PCPs [BCG+17], and have applications to constructing argument systems [BCS16] and IPs [RRR16].

The aforementioned work of [BCGV16] makes a simple modification to the PCP of Ben-Sasson and Sudan [BS08] and obtains a 2-round IOP for **NP** that is perfect zero knowledge, and yet has quasilinear size and a non-adaptive honest verifier. Our first result consists of extending this prior work to all languages in **NEXP**, positively answering an open question raised there. We do so by constructing, for each time T and query bound b, a suitable IOP for $\mathbf{NTIME}(T)$ that is zero knowledge against query bound b; the result for **NEXP** follows by setting b to be super-polynomial.

The foregoing notion of zero knowledge for IOPs directly extends that for PCPs, and requires showing the existence of an algorithm that simulates the view of any (malicious and adaptive) verifier interacting with the honest prover and making at most b queries across all oracles; here, 'view' consists of the answers to queries across all oracles.[1]

Theorem 1 (informal). *For every time bound T and query bound b, the complexity class* $\mathbf{NTIME}(T)$ *has 2-round Interactive Oracle Proofs that are perfect zero knowledge against b queries, and where the proof length is $\tilde{O}(T+\mathsf{b})$ and the (honest verifier's) query complexity is* $\mathrm{polylog}(T+\mathsf{b})$.

The prior work of [BCGV16] was "stuck" at **NP** because their simulator runs in $\mathrm{poly}(T+\mathsf{b})$ time so that T, b must be polynomially-bounded. In contrast, we achieve all of **NEXP** by constructing, essentially the same simple 2-round IOP, a simulator that runs in time $\mathrm{poly}(\tilde{q} + \log T + \log \mathsf{b})$, where $\tilde{q}$ is the *actual* number of queries made by the malicious verifier. This is an *exponential* improvement in simulation efficiency, and we obtain it by conceptualizing and solving a linear-algebraic problem about Reed–Solomon codes, and their proximity proofs, as discussed in Sect. 1.1.

In sum, our theorem gives new tradeoffs compared to [KPT97]'s result, which gives statistical zero knowledge PCPs for $\mathbf{NTIME}(T)$ with proof length $\mathrm{poly}(T, \mathsf{b})$ and an adaptive honest verifier. We obtain perfect zero knowledge for $\mathbf{NTIME}(T)$, with quasilinear proof length and a non-adaptive honest verifier, at the price of "2 rounds of PCPs".

Zero knowledge for counting problems. Kalai and Raz [KR08] introduce and study the notion of *interactive PCPs* (IPCPs), which "sits in between" IPs and IOPs: the prover first sends the verifier a PCP, and then the prover and verifier engage in a standard IP. IPCPs also enjoy better efficiency compared to PCPs or IPs alone [KR08].

[1] More precisely, while in a zero knowledge IP or MIP one is required to simulate the entire transcript of interaction (with one or multiple provers), in a zero knowledge IOP or PCP one is merely required to simulate answers to the oracle queries but not the entire oracle.

We show how a natural and simple modification of the sumcheck protocol of Lund et al. [LFKN92] achieves perfect zero knowledge in the IPCP model, even with a non-adaptive honest verifier. By running this protocol on the usual arithmetization of the counting problem associated to 3SAT, we obtain our second result, which is IPCPs for #**P** that are *perfect zero knowledge against unbounded queries*. This means that there exists a polynomial-time algorithm that simulates the view of any (malicious and adaptive) verifier making any polynomial number of queries to the PCP oracle. Here, 'view' consists of answers to oracle queries and the transcript of interaction with the prover. (In particular, this notion of zero knowledge is a 'hybrid' of corresponding notions for PCPs and IPs.)

Theorem 2 (informal). *The complexity class* #**P** *has Interactive PCPs that are perfect zero knowledge against unbounded queries. The PCP proof length is exponential, and the communication complexity of the interaction and the (honest verifier's) query complexity are polynomial.*

Our construction relies on a random self-reducibility property of the sumcheck protocol (see Sect. 2.2 for a summary) and its completeness and soundness properties are straightforward to establish. As in our previous result, the "magic" lies in the construction of the simulator, which must solve the same type of exponentially-large linear-algebraic problem, except that this time it is about Reed–Muller codes rather than Reed–Solomon codes. The algorithm that we give to solve this task relies on connections to the problem of polynomial identity testing in the area of algebraic complexity theory, as we discuss further below.

Goyal et al. [GIMS10] also study zero knowledge for IPCPs, and show how to obtain IPCPs for **NP** that (i) are statistical zero knowledge against unbounded queries, and yet (ii) each location of the (necessarily) super-polynomial size PCP is polynomial-time computable given the NP witness. They further prove that these two properties are not attainable by zero knowledge PCPs. Their construction consists of replacing the commitment scheme in the zero knowledge IP for 3-colorability of [GMW91] with an information-theoretic analogue in the IPCP model. Our Theorem 2 also achieves zero knowledge against unbounded queries, but targets the complexity class #**P** (rather than **NP**), for which there is no clear analogue of property (ii) above.

Information-theoretic commitments also underlie the construction of zero knowledge PCPs [KPT97]. One could apply the [KPT97] result for **NEXP** to obtain zero knowledge PCPs (thus also IPCPs) for #**P**, but this is an indirect and complex route (in particular, it relies on the PCP Theorem) that, moreover, yields an adaptive honest verifier. Our direct construction is simple and natural, and also yields a non-adaptive honest verifier.

We now discuss the common algebraic structure that allowed us to obtain both of the above results. We believe that further progress in understanding these types of algebraic techniques will lead to further progress in understanding the power of unconditional zero knowledge for IOPs and IPCPs, and perhaps also for MIPs and PCPs.

Succinct constraint detection for Reed–Muller and Reed–Solomon codes. The constructions underlying both of our theorems achieve zero knowledge by applying a simple modification to well-known protocols: the PCP of Ben-Sasson and Sudan [BS08] underlies our result for **NEXP** and the sumcheck protocol of Lund et al. [LFKN92] underlies our result for #**P**.

In both of these protocols the verifier has access (either via a polynomial-size representation or via a PCP oracle) to an exponentially-large word that allegedly belongs to a certain linear code, and the prover 'leaks' hard-to-compute information in the process of convincing the verifier that this word belongs to the linear code. We achieve zero knowledge via a modification that we call *masking*: the prover sends to the verifier a PCP containing a random codeword in this code, and then convinces the verifier that the *sum* of these two (the original codeword and this random codeword) is close to the linear code.[2] Intuitively, zero knowledge comes from the fact that the prover now argues about a random shift of the original word.

However, this idea raises a problem: how does the simulator 'sample' an exponentially-large random codeword in order to answer the verifier's queries to the PCP? Solving this problem crucially relies on solving a problem that lies at the intersection of coding theory, linear algebra, and computational complexity, which we call the *constraint detection problem*. We informally introduce it and state our results about it, and defer to Sect. 2.2 a more detailed discussion of its connection to zero knowledge.

Detecting constraints in codes. Constraint detection is the problem of determining which linear relations hold across all codewords of a linear code $C \subseteq \mathbb{F}^D$, when considering only a given subdomain $I \subseteq D$ of the code rather than all of the domain D. This problem can always be solved in time that is polynomial in $|D|$ (via Gaussian elimination); however, if there is an algorithm that solves this problem in time that is *polynomial in the subdomain's size* $|I|$, rather than the domain's size $|D|$, then we say that the code has *succinct* constraint detection; in particular, the domain could have *exponential* size and the algorithm would still run in polynomial time.

Definition 1 (informal). *We say that a linear code $C \subseteq \mathbb{F}^D$ has* **succinct constraint detection** *if there exists an algorithm that, given a subset $I \subseteq D$, runs in time* $\mathrm{poly}(\log|\mathbb{F}| + \log|D| + |I|)$ *and outputs $z \in \mathbb{F}^I$ such that $\sum_{i \in I} z(i)w(i) = 0$ for all $w \in C$, or "no" if no such z exists. (In particular, $|D|$ may be exponential.)*

We further discuss the problem of constraint detection in Sect. 2.1, and provide a formal treatment of it in Sect. 4.1. Beyond this introduction, we shall use (and achieve) a stronger definition of constraint detection: the algorithm is required to output a basis for the space of dual codewords in $C^\perp$ whose support lies in the subdomain I, i.e., a basis for the space $\{z \in D^I : \forall\, w \in C, \sum_{i \in I} z(i)w(i) = 0\}$.

[2] This is reminiscent of the use of a random secret share of 0 to achieve privacy in information-theoretic multi-party protocols [BGW88].

Note that in our discussion of succinct constraint detection we do not leverage the distance property of the code C, but we do leverage it in our eventual applications.

Our zero knowledge simulators' strategy includes sampling a "random PCP": a random codeword w in a linear code C with exponentially large domain size $|D|$ (see Sect. 2.2 for more on this). Explicitly sampling w requires time $\Omega(|D|)$, and so is inefficient. But a verifier makes only polynomially-many queries to w, so the simulator has to only simulate w when restricted to polynomial-size sets $I \subseteq D$, leaving open the possibility of doing so in time $\text{poly}(|I|)$. Achieving such a simulation time is an instance of (efficiently and perfectly) "implementing a huge random object" [GGN10] via a *stateful* algorithm [BW04]. We observe that if C has succinct constraint detection then this sampling problem for C has a solution: the simulator maintains the set $\{(i, a_i)\}_{i \in I}$ of past query-answer pairs; then, on a new verifier query $j \in D$, the simulator uses constraint detection to determine if w_j is linearly dependent on w_I, and answers accordingly (such linear dependencies characterize the required probability distribution, see Lemma 1).

Overall, our paper thus provides an application (namely, obtaining zero knowledge simulators) where the problem of efficient implementation of huge random objects arises naturally.

We now state our results about succinct constraint detection.

(1) Reed–Muller codes, and their partial sums. We prove that the family of linear codes comprised of evaluations of low-degree multivariate polynomials, along with their partial sums, has succinct constraint detection. This family is closely related to the *sumcheck protocol* [LFKN92], and indeed we use this result to obtain a PZK analogue of the sumcheck protocol (see Sect. 2.2), which yields Theorem 2 (see Sect. 2.3).

Recall that the family of Reed–Muller codes, denoted RM, is indexed by tuples $\mathbb{n} = (\mathbb{F}, m, d)$, where $\mathbb{F}$ is a finite field and m, d are positive integers, and the $\mathbb{n}$-th code consists of codewords $w \colon \mathbb{F}^m \to \mathbb{F}$ that are the evaluation of an m-variate polynomial Q of individual degree less than d over $\mathbb{F}$. We denote by ΣRM the family that extends RM with evaluations of all partial sums over certain subcubes of a hypercube:

Definition 2 (informal). *We denote by* ΣRM *the linear code family that is indexed by tuples* $\mathbb{n} = (\mathbb{F}, m, d, H)$*, where* H *is a subset of* $\mathbb{F}$*, and where the* $\mathbb{n}$*-th code consists of codewords* $(w_0, \ldots, w_m)$ *such that there exists an* m*-variate polynomial* Q *of individual degree less than* d *over* $\mathbb{F}$ *for which* $w_i \colon \mathbb{F}^{m-i} \to \mathbb{F}$ *is the evaluation of the* i*-th partial sum of* Q *over* H*, i.e.,* $w_i(\boldsymbol{\alpha}) = \sum_{\boldsymbol{\gamma} \in H^i} Q(\boldsymbol{\alpha}, \boldsymbol{\gamma})$ *for every* $\boldsymbol{\alpha} \in \mathbb{F}^{m-i}$*.*

The domain size for codes in ΣRM is $\Omega(|\mathbb{F}|^m)$, but our detector's running time is exponentially smaller.

Theorem 3 (informal statement of Theorem 5). *The family* ΣRM *has succinct constraint detection: there is a detector algorithm for* ΣRM *that runs in time* $\text{poly}(\log |\mathbb{F}| + m + d + |H| + |I|)$.

We provide intuition for the theorem's proof in Sect. 2.1 and provide the proof's details in Sect. 4.2; the proof leverages tools from algebraic complexity theory. (Our proof also shows that the family RM, which is a restriction of ΣRM, has succinct constraint detection.) Our theorem implies perfect and stateful implementation of a random low-degree multivariate polynomial and its partial sums over any hypercube; our proof extends an algorithm of [BW04], which solves this problem in the case of parity queries to boolean functions on subcubes of the boolean hypercube.

(2) Reed–Solomon codes, and their PCPPs. Second, we prove that the family of linear codes comprised of evaluations of low-degree univariate polynomials concatenated with corresponding BS proximity proofs [BS08] has succinct constraint detection. This family is closely related to quasilinear-size PCPs for **NEXP** [BS08], and indeed we use this result to obtain PZK proximity proofs for this family (see Sect. 2.2), from which we derive Theorem 1 (see Sect. 2.3).

Definition 3 (informal). *We denote by* BS-RS *the linear code family indexed by tuples* $\mathbb{n} = (\mathbb{F}, L, d)$*, where* $\mathbb{F}$ *is an extension field of* $\mathbb{F}_2$*,* L *is a linear subspace in* $\mathbb{F}$*, and* d *is a positive integer; the* $\mathbb{n}$*-th code consists of evaluations on* L *of univariate polynomials* Q *of degree less than* d*, concatenated with corresponding* [BS08] *proximity proofs.*

The domain size for codes in BS-RS is $\Omega(|L|)$, but our detector's running time is exponentially smaller.

Theorem 4 (informal statement of Theorem 6). *The family* BS-RS *has succinct constraint detection: there is a detector algorithm for* BS-RS *that runs in time* $\mathrm{poly}(\log|\mathbb{F}| + \dim(L) + |I|)$.

We provide intuition for the theorem's proof in Sect. 2.1 and provide the proof's details in Sect. 4.3; the proof leverages combinatorial properties of the recursive construction of BS proximity proofs.

2 Techniques

We informally discuss intuition behind our algorithms for detecting constraints (Sect. 2.1), their connection to zero knowledge (Sect. 2.2), and how we derive our results about #**P** and **NEXP** (Sect. 2.3). Throughout, we provide pointers to the technical sections that contain further details.

2.1 Detecting Constraints for Exponentially-Large Codes

As informally introduced in Sect. 1.1, the *constraint detection problem* corresponding to a linear code family $\mathscr{C} = \{C_{\mathbb{n}}\}_{\mathbb{n}}$ with domain $D(\cdot)$ and alphabet $\mathbb{F}(\cdot)$ is the following: given an index $\mathbb{n} \in \{0,1\}^*$ and subset $I \subseteq D(\mathbb{n})$, output a basis for the space $\{z \in D(\mathbb{n})^I : \forall\, w \in C_{\mathbb{n}}, \sum_{i \in I} z(i)w(i) = 0\}$. In other words,

for a given subdomain I, we wish to determine all linear relations that hold for codewords in $C_{\mathfrak{n}}$ restricted to the subdomain I.

If a generating matrix for $C_{\mathfrak{n}}$ can be found in polynomial time, this problem can be solved in $\text{poly}(|\mathfrak{n}| + |D(\mathfrak{n})|)$ time via Gaussian elimination (such an approach was implicitly taken by [BCGV16] to construct a perfect zero knowledge simulator for an IOP for **NP**). However, in our setting $|D(\mathfrak{n})|$ is *exponential* in $|\mathfrak{n}|$, so the straightforward solution is inefficient. With this in mind, we say that $\mathscr{C}$ has *succinct constraint detection* if there exists an algorithm that solves its constraint detection problem in $\text{poly}(|\mathfrak{n}| + |I|)$ time, even if $|D(\mathfrak{n})|$ is exponential in $|\mathfrak{n}|$.

The formal definition of succinct constraint detection is in Sect. 4.1. In the rest of this section we provide intuition for two of our theorems: succinct constraint detection for the family ΣRM and for the family BS-RS. As will become evident, the techniques that we use to prove the two theorems differ significantly. Perhaps this is because the two codes are quite different: ΣRM has a simple and well-understood algebraic structure, whereas BS-RS is constructed recursively using proof composition.

From algebraic complexity to detecting constraints for Reed–Muller codes and their partial sums. The purpose of this section is to provide intuition about the proof of Theorem 3, which states that the family ΣRM has succinct constraint detection. (Formal definitions, statements, and proofs are in Sect. 4.2.) We thus outline how to construct an algorithm that detects constraints for the family of linear codes comprised of evaluations of low-degree multivariate polynomials, along with their partial sums. Our construction generalizes the proof of [BW04], which solves the special case of parity queries to boolean functions on subcubes of the boolean hypercube by reducing this problem to a probabilistic identity testing problem that is solvable via an algorithm of [RS05].

Below, we temporarily ignore the partial sums, and focus on constructing an algorithm that detects constraints for the family of Reed–Muller codes RM, and at the end of the section we indicate how we can also handle partial sums.

Step 1: phrase as linear algebra problem. Consider a codeword $w\colon \mathbb{F}^m \to \mathbb{F}$ that is the evaluation of an m-variate polynomial Q of individual degree less than d over $\mathbb{F}$. Note that, for every $\boldsymbol{\alpha} \in \mathbb{F}^m$, $w(\boldsymbol{\alpha})$ equals the inner product of Q's coefficients with the vector $\phi_{\boldsymbol{\alpha}}$ that consists of the evaluation of all d^m monomials at $\boldsymbol{\alpha}$. One can argue that constraint detection for RM is equivalent to finding the nullspace of $\{\phi_{\boldsymbol{\alpha}}\}_{\boldsymbol{\alpha}\in I}$. However, "writing out" this $|I| \times d^m$ matrix and performing Gaussian elimination is too expensive, so we must solve this linear algebra problem *succinctly*.

Step 2: encode vectors as coefficients of polynomials. While each vector $\phi_{\boldsymbol{\alpha}}$ is long, it has a succinct description; in fact, we can construct an m-variate polynomial $\Phi_{\boldsymbol{\alpha}}$ whose coefficients (after expansion) are the entries of $\phi_{\boldsymbol{\alpha}}$, but has an arithmetic circuit of only size $O(md)$: namely, $\Phi_{\boldsymbol{\alpha}}(\boldsymbol{X}) := \prod_{i=1}^{m}(1 + \alpha_i X_i + \alpha_i^2 X_i^2 + \cdots + \alpha_i^{d-1} X_i^{d-1})$. Computing the nullspace of $\{\Phi_{\boldsymbol{\alpha}}\}_{\boldsymbol{\alpha}\in I}$ is thus equivalent to computing the nullspace of $\{\phi_{\boldsymbol{\alpha}}\}_{\boldsymbol{\alpha}\in I}$.

Step 3: computing the nullspace. Computing the nullspace of a set of polynomials is a problem in algebraic complexity theory, and is essentially equivalent to the Polynomial Identity Testing (PIT) problem, and so we leverage tools from that area.[3] While there are simple randomized algorithms to solve this problem (see for example [Kay10, Lemma 8] and [BW04]), these algorithms, due to a nonzero probability of error, suffice to achieve statistical zero knowledge but do not suffice to achieve perfect zero knowledge. To obtain perfect zero knowledge, we need a solution that has *no probability of error*. Derandomizing PIT for arbitrary algebraic circuits seems to be beyond current techniques (as it implies circuit lower bounds [KI04]), but derandomizations are currently known for some restricted circuit classes. The polynomials that we consider are special: they fall in the well-studied class of "sum of products of univariates", and for this case we can invoke the deterministic algorithm of [RS05] (see also [Kay10]). (It is interesting that derandomization techniques are ultimately used to obtain a qualitative improvement for an inherently probabilistic task, i.e., perfect sampling of verifier views.)

The above provides an outline for how to detect constraints for RM. The extension to ΣRM, which also includes partial sums, is achieved by considering a more general form of vectors $\phi_{\boldsymbol{\alpha}}$ as well as corresponding polynomials $\Phi_{\boldsymbol{\alpha}}$. These polynomials also have the special form required for our derandomization. See Sect. 4.2 for details.

From recursive code covers to detecting constraints for Reed–Solomon codes and their PCPPs. The purpose of this section is to provide intuition about the proof of Theorem 4, which states that the family BS-RS has succinct constraint detection. (Formal definitions, statements, and proofs are in Sect. 4.3.) We thus outline how to construct an algorithm that detects constraints for the family of linear codes comprised of evaluations of low-degree univariate polynomials concatenated with corresponding BS proximity proofs [BS08].

Our construction leverages the recursive structure of BS proximity proofs: we identify key combinatorial properties of the recursion that enable "local" constraint detection. To define and argue these properties, we introduce two notions that play a central role throughout the proof:

A *(local) view* of a linear code $C \subseteq \mathbb{F}^D$ is a pair $(\tilde{D}, \tilde{C})$ such that $\tilde{D} \subseteq D$ and $\tilde{C} = C|_{\tilde{D}} \subseteq \mathbb{F}^{\tilde{D}}$.
A *cover* of C is a set of local views $S = \{(\tilde{D}_j, \tilde{C}_j)\}_j$ of C such that $D = \cup_j \tilde{D}_j$.

Combinatorial properties of the recursive step. Given a finite field $\mathbb{F}$, domain $D \subseteq \mathbb{F}$, and degree d, let $C := \mathrm{RS}[\mathbb{F}, D, d]$ be the Reed–Solomon code

[3] PIT is the following problem: given a polynomial f expressed as an algebraic circuit, is f identically zero? This problem has well-known randomized algorithms [Zip79,Sch80], but deterministic algorithms for all circuits seem to be beyond current techniques [KI04]. PIT is a central problem in algebraic complexity theory, and suffices for solving a number of other algebraic problems. We refer the reader to [SY10] for a survey.

consisting of evaluations on D of univariate polynomials of degree less than d over $\mathbb{F}$; for concreteness, say that the domain size is $|D| = 2^n$ and the degree is $d = |D|/2 = 2^{n-1}$.

The first level of [BS08]'s recursion appends to each codeword $f \in C$ an auxiliary function $\pi_1(f)\colon D' \to \mathbb{F}$ with domain D' disjoint from D. Moreover, the mapping from f to $\pi_1(f)$ is linear over $\mathbb{F}$, so the set $C^1 := \{f \| \pi_1(f)\}_{f \in C}$, where $f\|\pi_1(f)\colon D \cup D' \to \mathbb{F}$ is the function that agrees with f on D and with $\pi_1(f)$ on D', is a linear code over $\mathbb{F}$. The code C^1 is the "first-level" code of a BS proximity proof for f.

The code C^1 has a naturally defined cover $S^1 = \{(\tilde{D}_j, \tilde{C}_j)\}_j$ such that each $\tilde{C}_j$ is a Reed–Solomon code $\mathrm{RS}[\mathbb{F}, \tilde{D}_j, d_j]$ with $2d_j \leq |\tilde{D}_j| = O(\sqrt{d})$, that is, with rate $1/2$ and block length $O(\sqrt{d})$. We prove several combinatorial properties of this cover:

- S^1 *is* 1 *-intersecting.* For all distinct j, j' in J, $|\tilde{D}_j \cap \tilde{D}_{j'}| \leq 1$ (namely, the subdomains are almost disjoint).
- S^1 *is* $O(\sqrt{d})$*-local.* Every partial assignment to $O(\sqrt{d})$ domains $\tilde{D}_j$ in the cover that is *locally consistent* with the cover can be extended to a *globally consistent* assignment, i.e., to a codeword of C^1. That is, there exists $\kappa = O(\sqrt{d})$ such that every partial assignment $h\colon \cup_{\ell=1}^{\kappa} \tilde{D}_{j_\ell} \to \mathbb{F}$ with $h|_{\tilde{D}_{j_\ell}} \in \tilde{C}_{j_\ell}$ (for each ℓ) equals the restriction to the subdomain $\cup_{\ell=1}^{\kappa} \tilde{D}_{j_\ell}$ of some codeword $f\|\pi_1(f)$ in C^1.
- S^1 *is* $O(\sqrt{d})$*-independent.* The ability to extend locally-consistent assignments to "globally-consistent" codewords of C^1 holds in a stronger sense: even when the aforementioned partial assignment h is extended *arbitrarily* to κ additional point-value pairs, this new partial assignment still equals the restriction of some codeword $f\|\pi_1(f)$ in C^1.

The locality property alone already suffices to imply that, given a subdomain $I \subseteq D \cup D'$ of size $|I| < \sqrt{d}$, we can solve the constraint detection problem on I by considering only those constraints that appear in views that intersect I. But C has exponential block length so a "quadratic speedup" does not yet imply succinct constraint detection. To obtain it, we also leverage the intersection and independence properties to reduce "locality" as follows.

Further recursive steps. So far we have only considered the first recursive step of a BS proximity proof; we show how to obtain covers with smaller locality (and thereby detect constraints with more efficiency) by considering additional recursive steps. Each code $\tilde{C}_j$ in the cover S^1 of C^1 is a Reed–Solomon code $\mathrm{RS}[\mathbb{F}, \tilde{D}_j, d_j]$ with $|\tilde{D}_j|, d_j = O(\sqrt{d})$, and the next recursive step appends to each codeword in $\tilde{C}_j$ a corresponding auxiliary function, yielding a new code C^2. In turn, C^2 has a cover S^2, and another recursive step yields a new code C^3, which has its own cover S^3, and so on. The crucial technical observation is that the intersection and independence properties, which hold recursively, enable us to deduce that C^i is 1-intersecting, $O(\sqrt[2^i]{d})$-local, and $O(\sqrt[2^i]{d})$-independent; in particular, for $r = \log\log d + O(1)$, S^r is 1-intersecting, $O(1)$-local, $O(1)$-independent.

Then, recalling that detecting constraints for local codes requires only the views in the cover that intersect I, our constraint detector works by choosing $i \in \{1, \ldots, r\}$ such that the cover S^i is $\text{poly}(|I|)$-local, finding in this cover a $\text{poly}(|I|)$-size set of $\text{poly}(|I|)$-size views that intersect I, and computing in $\text{poly}(|I|)$ time a basis for the dual of each of these views — thereby proving Theorem 4.

Remark 1. For the sake of those familiar with BS-RS we remark that the domain D' is the carefully chosen subset of $\mathbb{F} \times \mathbb{F}$ designated by that construction, the code C^1 is the code that evaluates bivariate polynomials of degree $O(\sqrt{d})$ on $D \cup D'$ (along the way mapping $D \subseteq \mathbb{F}$ to a subset of $\mathbb{F} \times \mathbb{F}$), the subdomains $\tilde{D}_j$ are the axis-parallel "rows" and "columns" used in that recursive construction, and the codes $\tilde{C}_j$ are Reed–Solomon codes of block length $O(\sqrt{d})$. The $O(\sqrt{d})$-locality and independence follow from basic properties of bivariate Reed–Muller codes; see the full version for more details.

Remark 2. It is interesting to compare the above result with *linear lower bounds on query complexity* for testing proximity to random low density parity check (LDPC) codes [BHR05, BGK+10]. Those results are proved by obtaining a basis for the dual code such that every small-support constraint is spanned by a small subset of that basis. The same can be observed to hold for BS-RS, even though this latter code is locally testable with *polylogarithmic query complexity* [BS08, Theorem 2.13]. The difference between the two cases is due to the fact that, for a random LDPC code, an assignment that satisfies all but a single basis-constraint is (with high probability) far from the code, whereas the recursive and 1-intersecting structure of BS-RS implies the existence of words that satisfy all but a single basis constraint, yet are negligibly close to being a codeword.

2.2 From Constraint Detection to Zero Knowledge via Masking

We provide intuition about the connection between constraint detection and zero knowledge (Sect. 2.2), and how we leverage this connection to achieve two intermediate results: (i) protocol that is zero knowledge in the Interactive PCP model (Sect. 2.2); and (ii) proximity proofs for Reed–Solomon codes that are zero knowledge in the Interactive Oracle Proof model (Sect. 2.2).

Local simulation of random codewords. Suppose that the prover and verifier both have oracle access to a codeword $w \in C$, for some linear code $C \subseteq \mathbb{F}^D$ with exponential-size domain D, and that they need to engage in some protocol that involves w. During the protocol, the prover may leak information about w that is hard to compute (e.g., requires exponentially-many queries to w), and so would violate zero knowledge (as we see below, this is the case for protocols such as sumcheck).

Rather than directly invoking the protocol, the prover first sends to the verifier a random codeword $r \in C$ (as an oracle since r has exponential size) and the verifier replies with a random field element $\rho \in \mathbb{F}$; then the prover and verifier invoke the protocol on the new codeword $w' := \rho w + r \in C$ rather than w.

Intuitively, running the protocol on w' now does not leak information about w, because w' is random in C (up to resolvable technicalities). This *random self-reducibility* makes sense for only some protocols, e.g., those where completeness is preserved for any choice of ρ and soundness is broken for only a small fraction of ρ; but this will indeed be the case for the settings described below.

The aforementioned *masking* technique was used by [BCGV16] for codes with polynomial-size domains, but we use it for codes with exponential-size domains, which requires exponentially more efficient simulation techniques. Indeed, to prove (perfect) zero knowledge, a simulator must be able to reproduce, exactly, the view obtained by any malicious verifier that queries entries of w', a uniformly random codeword in C; however, it is too expensive for the simulator to explicitly sample a random codeword and answer the verifier's queries according to it. Instead, the simulator must sample the "local view" that the verifier sees while querying w' at a *small* number of locations $I \subseteq D$.

But simulating local views of the form $w'|_I$ is reducible to detecting *constraints*, i.e., codewords in the dual code $C^\perp$ whose support is contained in I. Indeed, if no word in $C^\perp$ has support contained in I then $w'|_I$ is uniformly random; otherwise, each additional linearly independent constraint of $C^\perp$ with support contained in I further reduces the entropy of $w'|_I$ in a well-understood manner. (See Lemma 1 for a formal statement.) In sum, succinct constraint detection enables us to "implement" [GGN10, BW04] random codewords of C despite C having exponential size.

Note that in the above discussion we implicitly assumed that the set I is known in advance, i.e., that the verifier chooses its queries in advance. This, of course, need not be the case: a verifier may adaptively make queries based on answers to previous queries and, hence, the set I need not be known a priori. This turns out to not be a problem because, given a constraint detector, it is straightforward to compute the conditional distribution of the view $w'|_I$ given $w'|_J$ for a subset J of I. This is expressed precisely in Lemma 1.

We now discuss two concrete protocols for which the aforementioned random self-reducibility applies, and for which we also have constructed suitably-efficient constraint detectors.

Zero knowledge sumchecks. The celebrated sumcheck protocol [LFKN92] is *not* zero knowledge. In the sumcheck protocol, the prover and verifier have oracle access to a low-degree m-variate polynomial F over a field $\mathbb{F}$, and the prover wants to convince the verifier that $\sum_{\boldsymbol{\alpha} \in H^m} F(\boldsymbol{\alpha}) = 0$ for a given subset H of $\mathbb{F}$. During the protocol, the prover communicates partial sums of F, which are #**P**-hard to compute and, as such, violate zero knowledge.

We now explain how to use random self-reducibility to make the sumcheck protocol *(perfect) zero knowledge*, at the cost of moving from the Interactive Proof model to the Interactive PCP model.

IPCP sumcheck. Consider the following tweak to the classical sumcheck protocol: rather than invoking sumcheck on F directly, the prover first sends to the verifier (the evaluation of) a random low-degree polynomial R as an oracle; then, the prover sends the value $z := \sum_{\boldsymbol{\alpha} \in H^m} R(\boldsymbol{\alpha})$ and the verifier replies

with a random field element ρ; finally, the two invoke sumcheck on the claim "$\sum_{\boldsymbol{\alpha} \in H^m} Q(\boldsymbol{\alpha}) = z$" where $Q := \rho F + R$.

Completeness is clear because if $\sum_{\boldsymbol{\alpha} \in H^m} F(\boldsymbol{\alpha}) = 0$ and $\sum_{\boldsymbol{\alpha} \in H^m} R(\boldsymbol{\alpha}) = z$ then $\sum_{\boldsymbol{\alpha} \in H^m} (\rho F + R)(\boldsymbol{\alpha}) = z$; soundness is also clear because if $\sum_{\boldsymbol{\alpha} \in H^m} F(\boldsymbol{\alpha}) \neq 0$ then $\sum_{\boldsymbol{\alpha} \in H^m} (\rho F + R)(\boldsymbol{\alpha}) \neq z$ with high probability over ρ, regardless of the choice of R. (For simplicity, we ignore the fact that the verifier also needs to test that R has low degree.) We are thus left to show (perfect) zero knowledge, which turns out to be a much less straightforward argument.

The simulator. Before we explain how to argue zero knowledge, we first clarify what we mean by it: since the verifier has oracle access to F we cannot hope to 'hide' it; nevertheless, we can hope to argue that the verifier, by participating in the protocol, does not learn anything about F beyond what the verifier can directly learn by querying F (and the fact that F sums to zero on H^m). What we shall achieve is the following: an algorithm that simulates the verifier's view by making as many queries to F as the *total* number of verifier queries to either F or R.[4]

On the surface, zero knowledge seems easy to argue, because $\rho F + R$ seems random among low-degree m-variate polynomials. More precisely, consider the simulator that samples a random low-degree polynomial Q and uses it instead of $\rho F + R$ and answers the verifier queries as follows: (a) whenever the verifier queries $F(\boldsymbol{\alpha})$, respond by querying $F(\boldsymbol{\alpha})$ and returning the true value; (b) whenever the verifier queries $R(\boldsymbol{\alpha})$, respond by querying $F(\boldsymbol{\alpha})$ and returning $Q(\boldsymbol{\alpha}) - \rho F(\boldsymbol{\alpha})$. Observe that the number of queries to F made by the simulator equals the number of (mutually) distinct queries to F and R made by the verifier, as desired.

However, the above reasoning, while compelling, is insufficient. First, $\rho F + R$ is *not* random because a malicious verifier can choose ρ depending on queries to R. Second, even if $\rho F + R$ were random (e.g., the verifier does not query R before choosing ρ), the simulator must run in polynomial time, both producing correctly-distributed 'partial sums' of $\rho F + R$ and answering queries to R, but sampling Q alone requires exponential time. In this high level discussion we ignore the first problem (which nonetheless has to be tackled), and focus on the second.

At this point it should be clear from the discussion in Sect. 2.2 that the simulator does not have to sample Q explicitly, but only has to perfectly simulate local views of it by leveraging the fact that it can keep state across queries. And doing so requires solving the succinct constraint detection problem for a suitable code C. In this case, it suffices to consider the code $C = \Sigma\mathrm{RM}$, and our Theorem 3 guarantees the required constraint detector.

We refer the reader to the full version for further details.

[4] A subsequent work [CFS17] shows how to bootstrap this IPCP sumcheck protocol into a more complex one that has a stronger zero knowledge guarantee: the simulator can sample the verifier's view by making as many queries to F as the number of verifier queries (plus one). Nevertheless, the weaker zero knowledge guarantee that we achieve suffices for our purposes.

Zero knowledge proximity proofs for Reed–Solomon. Testing proximity of a codeword w to a given linear code C can be aided by a *proximity proof* [DR04,BGH+06], which is an auxiliary oracle π that facilitates testing (e.g., C is not locally testable). For example, testing proximity to the Reed–Solomon code, a crucial step towards achieving short PCPs, is aided via suitable proximity proofs [BS08].

From the perspective of zero knowledge, however, a proximity proof can be 'dangerous': a few locations of π can in principle leak a lot of information about the codeword w, and a malicious verifier could potentially learn a lot about w with only a few queries to w and π. The notion of zero knowledge for proximity proofs requires that this cannot happen: it requires the existence of an algorithm that simulates the verifier's view by making as many queries to w as the *total* number of verifier queries to either w or π [IW14]; intuitively, this means that any bit of the proximity proof π reveals no more information than one bit of w.

We demonstrate again the use of random self-reducibility and show a general transformation that, under certain conditions, maps a PCP of proximity (P, V) for a code C to a corresponding 2-round Interactive Oracle Proof of Proximity (IOPP) for C that is *(perfect) zero knowledge*.

IOP of proximity for C. Consider the following IOP of Proximity: the prover and verifier have oracle access to a codeword w, and the prover wants to convince the verifier that w is close to C; the prover first sends to the verifier a random codeword r in C, and the verifier replies with a random field element ρ; the prover then sends the proximity proof $\pi' := P(w')$ that attests that $w' := \rho w + r$ is close to C. Note that this is a 2-round IOP of Proximity for C, because completeness follows from the fact that C is linear, while soundness follows because if w is far from C, then so is $\rho w + r$ for every r with high probability over ρ. But is the zero knowledge property satisfied?

The simulator. Without going into details, analogously to Sect. 2.1, a simulator must be able to sample local views for random codewords from the code $L := \{ w \| P(w) \}_{w \in C}$, so the simulator's efficiency reduces to the efficiency of constraint detection for L. We indeed prove that if L has succinct constraint detection then the simulator works out. See the full version for further details.

The case of Reed–Solomon. The above machinery allows us to derive a zero knowledge IOP of Proximity for Reed–Solomon codes, thanks to our Theorem 4, which states that the family of linear codes comprised of evaluations of low-degree univariate polynomials concatenated with corresponding BS proximity proofs [BS08] has succinct constraint detection; see the full version for details. This is one of the building blocks of our construction of zero knowledge IOPs for **NEXP**, as described below in Sect. 2.3.

2.3 Achieving Zero Knowledge Beyond NP

We outline how to derive our results about zero knowledge for #**P** and **NEXP**.

Zero knowledge for counting problems. We provide intuition for the proof of Theorem 2, which states that the complexity class #**P** has Interactive PCPs that are perfect zero knowledge.

We first recall the classical (non zero knowledge) Interactive Proof for #**P** [LFKN92]. The language $\mathscr{L}_{\#3\mathrm{SAT}}$, which consists of pairs (ϕ, N) where ϕ is a 3-CNF boolean formula and N is the number of satisfying assignments of ϕ, is #**P**-complete, and thus it suffices to construct an IP for it. The IP for $\mathscr{L}_{\#3\mathrm{SAT}}$ works as follows: the prover and verifier both *arithmetize* ϕ to obtain a low-degree multivariate polynomial p_ϕ and invoke the (non zero knowledge) sumcheck protocol on the claim "$\sum_{\boldsymbol{\alpha}\in\{0,1\}^n} p_\phi(\boldsymbol{\alpha}) = N$", where arithmetic is over a large-enough prime field.

Returning to our goal, we obtain a perfect zero knowledge Interactive PCP by simply replacing the (non zero knowledge) IP sumcheck mentioned above with our perfect zero knowledge IPCP sumcheck, described in Sect. 2.2. In the full version we provide further details, including proving that the zero knowledge guarantees of our sumcheck protocol suffice for this case.

Zero knowledge for nondeterministic time. We provide intuition for the proof of Theorem 1, which implies that the complexity class **NEXP** has Interactive Oracle Proofs that are perfect zero knowledge. Very informally, the proof consists of combining two building blocks: (i) [BCGV16]'s reduction from **NEXP** to *randomizable* linear algebraic constraint satisfaction problems, and (ii) our construction of perfect zero knowledge IOPs of Proximity for Reed–Solomon codes, described in Sect. 2.2. Besides extending [BCGV16]'s result from **NP** to **NEXP**, our proof provides a conceptual simplification over [BCGV16] by clarifying how the above two building blocks work together towards the final result. We now discuss this.

Starting point: [BS08]**.** Many PCP constructions consist of two steps: (1) arithmetize the statement at hand (in our case, membership of an instance in some **NEXP**-complete language) by reducing it to a "PCP-friendly" problem that looks like a *linear-algebraic* constraint satisfaction problem (LACSP); (2) design a tester that probabilistically checks witnesses for this LACSP. In this paper, as in [BCGV16], we take [BS08]'s PCPs for **NEXP** as a starting point, where the first step reduces **NEXP** to a "univariate" LACSP whose witnesses are codewords in a Reed–Solomon code of exponential degree that satisfy certain properties, and whose second step relies on suitable *proximity proofs* [DR04,BGH+06] for that code. Thus, overall, the PCP consists of two oracles, one being the LACSP witness and the other being the corresponding BS proximity proof, and it is not hard to see that such a PCP is *not* zero knowledge, because both the LACSP witness and its proximity proof reveal hard-to-compute information.

Step 1: sanitize the proximity proof. We first address the problem that the BS proximity proof "leaks", by simply replacing it with our own perfect zero knowledge analogue. Namely, we replace it with our perfect zero knowledge 2-round IOP of Proximity for Reed–Solomon codes, described in Sect. 2.2. This modification ensures that there exists an algorithm that perfectly simulates the verifier's view by making as many queries to the LACSP witness as the *total* number of verifier queries to *either the LACSP witness or other oracles used to facilitate proximity testing*. At this point we have obtained a perfect zero

knowledge 2-round IOP of Proximity for **NEXP** (analogous to the notion of a zero knowledge PCP of Proximity [IW14]); this part is where, previously, [BCGV16] were restricted to **NP** because their simulator only handled Reed–Solomon codes with *polynomial* degree while our simulator is efficient even for such codes with *exponential* degree. But we are not done yet: to obtain our goal, we also need to address the problem that the LACSP witness itself "leaks" when the verifier queries it, which we discuss next.

Step 2: sanitize the witness. Intuitively, we need to inject randomness in the reduction from **NEXP** to LACSP because the prover ultimately sends an LACSP witness to the verifier as an oracle, which the verifier can query. This is precisely what [BCGV16]'s reduction from **NEXP** to *randomizable* LACSPs enables, and we thus use their reduction to complete our proof. Informally, given an a-priori query bound b on the verifier's queries, the reduction outputs a witness w with the property that one can efficiently sample *another* witness w' whose entries are b-wise independent. We can then simply use the IOP of Proximity from the previous step on this randomized witness. Moreover, since the efficiency of the verifier is polylogarithmic in b, we can set b to be super-polynomial (e.g., exponential) to preserve zero knowledge against any polynomial number of verifier queries.

The above discussion is only a sketch and we refer the reader to the full version for further details. One aspect that we did not discuss is that an LACSP witness actually consists of two sub-witnesses, where one is a "local" deterministic function of the other, which makes arguing zero knowledge somewhat more delicate.

2.4 Roadmap

Our results are structured as in the table below. For details, see the full version.

§4.2 **Theorem 3/5** detecting constraints for ΣRM	§4.3 **Theorem 4/6** detecting constraints for BS-RS
↓	↓
PZK IPCP for sumcheck	PZK IOP of Proximity for RS codes
↓	↓
Theorem 2 PZK IPCP for #**P**	**Theorem 1** PZK IOP for **NEXP**

3 Definitions

3.1 Basic Notations

Functions, distributions, fields. We use $f\colon D \to R$ to denote a function with domain D and range R; given a subset $\tilde{D}$ of D, we use $f|_{\tilde{D}}$ to denote the restriction of f to $\tilde{D}$. Given a distribution $\mathcal{D}$, we write $x \leftarrow \mathcal{D}$ to denote that x is sampled according to $\mathcal{D}$. We denote by $\mathbb{F}$ a finite field and by $\mathbb{F}_q$ the field of size q; we say $\mathbb{F}$ is a *binary field* if its characteristic is 2. Arithmetic operations over $\mathbb{F}_q$ cost polylog q but we shall consider these to have unit cost (and inspection shows that accounting for their actual polylogarithmic cost does not change any of the stated results).

Distances. A distance measure is a function $\Delta\colon \Sigma^n \times \Sigma^n \to [0,1]$ such that for all $x, y, z \in \Sigma^n$: (i) $\Delta(x,x) = 0$, (ii) $\Delta(x,y) = \Delta(y,x)$, and (iii) $\Delta(x,y) \leq \Delta(x,z) + \Delta(z,y)$. We extend Δ to distances to sets: given $x \in \Sigma^n$ and $S \subseteq \Sigma^n$, we define $\Delta(x,S) := \min_{y \in S} \Delta(x,y)$ (or 1 if S is empty). We say that a string x is ϵ-close to another string y if $\Delta(x,y) \leq \epsilon$, and ϵ-far from y if $\Delta(x,y) > \epsilon$; similar terminology applies for a string x and a set S. Unless noted otherwise, we use the *relative Hamming distance* over alphabet Σ (typically implicit): $\Delta(x,y) := |\{i : x_i \neq y_i\}|/n$.

Languages and relations. We denote by $\mathscr{R}$ a (binary ordered) relation consisting of pairs $(\mathtt{x}, \mathtt{w})$, where $\mathtt{x}$ is the *instance* and $\mathtt{w}$ is the *witness*. We denote by $\mathrm{Lan}(\mathscr{R})$ the language corresponding to $\mathscr{R}$, and by $\mathscr{R}|_{\mathtt{x}}$ the set of witnesses in $\mathscr{R}$ for $\mathtt{x}$ (if $\mathtt{x} \notin \mathrm{Lan}(\mathscr{R})$ then $\mathscr{R}|_{\mathtt{x}} := \emptyset$). As always, we assume that $|\mathtt{w}|$ is bounded by some computable function of $n := |\mathtt{x}|$; in fact, we are mainly interested in relations arising from nondeterministic languages: $\mathscr{R} \in \mathbf{NTIME}(T)$ if there exists a $T(n)$-time machine M such that $M(\mathtt{x}, \mathtt{w})$ outputs 1 if and only if $(\mathtt{x}, \mathtt{w}) \in \mathscr{R}$. Throughout, we assume that $T(n) \geq n$. We say that $\mathscr{R}$ has relative distance $\delta_{\mathscr{R}}\colon \mathbb{N} \to [0,1]$ if $\delta_{\mathscr{R}}(n)$ is the minimum relative distance among witnesses in $\mathscr{R}|_{\mathtt{x}}$ for all $\mathtt{x}$ of size n. Throughout, we assume that $\delta_{\mathscr{R}}$ is a constant.

Polynomials. We denote by $\mathbb{F}[X_1, \ldots, X_m]$ the ring of polynomials in m variables over $\mathbb{F}$. Given a polynomial P in $\mathbb{F}[X_1, \ldots, X_m]$, $\deg_{X_i} P[X_i]$ is the degree of P in the variable X_i. We denote by $\mathbb{F}^{<d}[X_1, \ldots, X_m]$ the subspace consisting of $P \in \mathbb{F}[X_1, \ldots, X_m]$ with $\deg_{X_i} P[X_i] < d$ for every $i \in \{1, \ldots, m\}$.

Random shifts. We later use a folklore claim about distance preservation for random shifts in linear spaces.

Claim. Let n be in $\mathbb{N}$, $\mathbb{F}$ a finite field, S an $\mathbb{F}$-linear space in $\mathbb{F}^n$, and $x, y \in \mathbb{F}^n$. If x is ϵ-far from S, then $\alpha x + y$ is $\epsilon/2$-far from S, with probability $1 - |\mathbb{F}|^{-1}$ over a random $\alpha \in \mathbb{F}$. (Distances are relative Hamming distances.)

3.2 Single-Prover Proof Systems

We use two types of proof systems that combine aspects of interactive proofs [Bab85, GMR89] and probabilistically checkable proofs [BFLS91, AS98, ALM+98]: **interactive PCPs** (IPCPs) [KR08] and **interactive oracle proofs** (IOPs) [BCS16, RRR16]. We first describe IPCPs (Sect. 3.2) and then IOPs (Sect. 3.2), which generalize the former.

Interactive probabilistically checkable proofs. An **IPCP** [KR08] is a PCP followed by an IP. Namely, the prover P and verifier V interact as follows: P sends to V a probabilistically checkable proof π; afterwards, P and V^{π} engage in an interactive proof. Thus, V may read a few bits of π but must read subsequent messages from P in full. An *IPCP system* for a relation $\mathscr{R}$ is thus a pair (P, V), where P, V are probabilistic interactive algorithms working as described, that satisfies naturally-defined notions of perfect completeness and soundness with a given error $\varepsilon(\cdot)$; see [KR08] for details.

We say that an IPCP has k rounds if this "PCP round" is followed by a $(\mathsf{k}-1)$-round interactive proof. (That is, we count the PCP round towards round complexity, unlike [KR08].) Beyond round complexity, we also measure how many bits the prover sends and how many the verifier reads: the *proof length* l is the length of π in bits plus the number of bits in all subsequent prover messages; the *query complexity* q is the number of bits of π read by the verifier plus the number of bits in all subsequent prover messages (since the verifier must read all of those bits).

In this work, we do not count the number of bits in the verifier messages, nor the number of random bits used by the verifier; both are bounded from above by the verifier's running time, which we do consider. Overall, we say that a relation $\mathscr{R}$ belongs to the complexity class $\mathbf{IPCP}[\mathsf{k},\mathsf{l},\mathsf{q},\varepsilon,\mathsf{tp},\mathsf{tv}]$ if there is an IPCP system for $\mathscr{R}$ in which: (1)the number of rounds is at most $\mathsf{k}(n)$; (2) the proof length is at most $\mathsf{l}(n)$; (3) the query complexity is at most $\mathsf{q}(n)$; (4) the soundness error is $\varepsilon(n)$; (5) the prover algorithm runs in time $\mathsf{tp}(n)$; (6) the verifier algorithm runs in time $\mathsf{tv}(n)$.

Interactive oracle proofs. An **IOP** [BCS16,RRR16] is a "multi-round PCP". That is, an IOP generalizes an interactive proof as follows: whenever the prover sends to the verifier a message, the verifier does not have to read the message in full but may probabilistically query it. In more detail, a k-round IOP comprises k rounds of interaction. In the i-th round of interaction: the verifier sends a message m_i to the prover; then the prover replies with a message π_i to the verifier, which the verifier can query in this and later rounds (via oracle queries). After the k rounds of interaction, the verifier either accepts or rejects.

An *IOP system* for a relation $\mathscr{R}$ with soundness error ε is thus a pair (P,V), where P,V are probabilistic interactive algorithms working as described, that satisfies the following properties. (See [BCS16] for more details.)

Completeness: For every instance-witness pair $(\mathbb{x},\mathbb{w})$ in the relation $\mathscr{R}$, $\Pr[\langle P(\mathbb{x},\mathbb{w}),V(\mathbb{x})\rangle=1]=1$.

Soundness: For every instance $\mathbb{x}$ not in $\mathscr{R}$'s language and unbounded malicious prover $\tilde{P}$, $\Pr[\langle \tilde{P},V(\mathbb{x})\rangle=1]\leq\varepsilon(n)$.

Beyond round complexity, we also measure how many bits the prover sends and how many the verifier reads: the *proof length* l is the total number of bits in all of the prover's messages, and the *query complexity* q is the total number of bits read by the verifier across all of the prover's messages. Considering all of these parameters, we say that a relation $\mathscr{R}$ belongs to the complexity class $\mathbf{IOP}[\mathsf{k},\mathsf{l},\mathsf{q},\varepsilon,\mathsf{tp},\mathsf{tv}]$ if there is an IOP system for $\mathscr{R}$ in which: (1) the number of rounds is at most $\mathsf{k}(n)$; (2) the proof length is at most $\mathsf{l}(n)$; (3) the query complexity is at most $\mathsf{q}(n)$; (4) the soundness error is $\varepsilon(n)$; (5) the prover algorithm runs in time $\mathsf{tp}(n)$; (6) the verifier algorithm runs in time $\mathsf{tv}(n)$.

IOP vs. IPCP. An IPCP (see Sect. 3.2) is a special case of an IOP because an IPCP verifier must read in full all of the prover's messages except the first one (while an IOP verifier may query any part of any prover message). The above complexity measures are consistent with those defined for IPCPs.

Restrictions and extensions. The definitions below are about IOPs, but IPCPs inherit all of these definitions because they are a special case of IOP.

Adaptivity of queries. An IOP system is *non-adaptive* if the verifier queries are non-adaptive, i.e., the queried locations depend only on the verifier's inputs.

Public coins. An IOP system is *public coin* if each verifier message m_i is chosen uniformly and independently at random, and all of the verifier queries happen after receiving the last prover message.

Proximity. An *IOP of proximity* extends the definition of an IOP in the same way that a PCP of proximity extends that of a PCP [DR04,BGH+06]. An *IOPP system* for a relation $\mathscr{R}$ with soundness error ε and proximity parameter δ is a pair (P, V) that satisfies the following properties.

Completeness: For every instance-witness pair $(\mathtt{x}, \mathtt{w})$ in the relation $\mathscr{R}$, $\Pr[\langle P(\mathtt{x}, \mathtt{w}), V^{\mathtt{w}}(\mathtt{x})\rangle = 1] = 1$.

Soundness: For every instance-witness pair $(\mathtt{x}, \mathtt{w})$ with $\Delta(\mathtt{w}, \mathscr{R}|_{\mathtt{x}}) \geq \delta(n)$ and unbounded malicious prover $\tilde{P}$, $\Pr[\langle \tilde{P}, V^{\mathtt{w}}(\mathtt{x})\rangle = 1] \leq \varepsilon(n)$.

Similarly to above, a relation $\mathscr{R}$ belongs to the complexity class $\mathbf{IOPP}[\mathsf{k}, \mathsf{l}, \mathsf{q}, \varepsilon, \delta, \mathsf{tp}, \mathsf{tv}]$ if there is an IOPP system for $\mathscr{R}$ with the corresponding parameters. Following [IW14], we call an IOPP *exact* if $\delta(n) = 0$.

Promise relations. A *promise relation* is a relation-language pair $(\mathscr{R}^{\mathsf{YES}}, \mathscr{L}^{\mathsf{NO}})$ with $\mathrm{Lan}(\mathscr{R}^{\mathsf{YES}}) \cap \mathscr{L}^{\mathsf{NO}} = \emptyset$. An IOP for a promise relation is the same as an IOP for the (standard) relation $\mathscr{R}^{\mathsf{YES}}$, except that soundness need only hold for $\mathtt{x} \in \mathscr{L}^{\mathsf{NO}}$. An IOPP for a promise relation is the same as an IOPP for the (standard) relation $\mathscr{R}^{\mathsf{YES}}$, except that soundness need only hold for $\mathtt{x} \in \mathrm{Lan}(\mathscr{R}^{\mathsf{YES}}) \cup \mathscr{L}^{\mathsf{NO}}$.

Prior constructions. In this paper we give new IPCP and IOP constructions that achieve perfect zero knowledge for various settings. Below we summarize known constructions in these two models.

IPCPs. Prior work obtains IPCPs with proof length that depends on the witness size rather than computation size [KR08,GKR08], and IPCPs with statistical zero knowledge [GIMS10] (see Sect. 3.3 for more details).

IOPs. Prior work obtains IOPs with perfect zero knowledge for **NP** [BCGV16], IOPs with small proof length and query complexity [BCG+17], and an amortization theorem for "unambiguous" IOPs [RRR16]. Also, [BCS16] show how to compile public-coin IOPs into non-interactive arguments in the random oracle model.

3.3 Zero Knowledge

We define the notion of zero knowledge for IOPs and IPCPs achieved by our constructions: *unconditional (perfect) zero knowledge via straightline simulators.*

This notion is quite strong not only because it unconditionally guarantees simulation of the verifier's view but also because straightline simulation implies desirable properties such as composability. We now provide some context and then give formal definitions.

At a high level, zero knowledge requires that the verifier's view can be efficiently simulated without the prover. Converting the informal statement into a mathematical one involves many choices, including choosing which verifier class to consider (e.g., the honest verifier? all polynomial-time verifiers?), the quality of the simulation (e.g., is it identically distributed to the view? statistically close to it? computationally close to it?), the simulator's dependence on the verifier (e.g., is it non-uniform? or is the simulator universal?), and others. The definitions below consider two variants: perfect simulation via universal simulators against either unbounded-query or bounded-query verifiers.

Moreover, in the case of universal simulators, one distinguishes between a non-blackbox use of the verifier, which means that the simulator takes the verifier's code as input, and a blackbox use of it, which means that the simulator only accesses the verifier via a restricted interface; we consider this latter case. Different models of proof systems call for different interfaces, which grant carefully-chosen "extra powers" to the simulator (in comparison to the prover) so to ensure that efficiency of the simulation does not imply the ability to efficiently decide the language. For example: in ZK IPs, the simulator may rewind the verifier; in ZK PCPs, the simulator may adaptively answer oracle queries. In ZK IPCPs and ZK IOPs (our setting), the natural definition would allow a blackbox simulator to rewind the verifier *and also* to adaptively answer oracle queries. The definitions below, however, consider only simulators that are straightline [FS89,DS98], that is they do not rewind the verifier, because our constructions achieve this stronger notion.

We are now ready to define the notion of unconditional (perfect) zero knowledge via straightline simulators. We first discuss the notion for IOPs, then for IOPs of proximity, and finally for IPCPs.

ZK for IOPs. We define zero knowledge (via straightline simulators) for IOPs. We begin by defining the view of an IOP verifier.

Definition 4. *Let A, B be algorithms and x, y strings. We denote by* $\mathrm{View}\,\langle B(y), A(x)\rangle$ *the* **view** *of $A(x)$ in an interactive oracle protocol with $B(y)$, i.e., the random variable $(x, r, a_1, \ldots, a_n)$ where x is A's input, r is A's randomness, and $a_1, \ldots, a_n$ are the answers to A's queries into B's messages.*

Straightline simulators in the context of IPs were used in [FS89], and later defined in [DS98]. The definition below considers this notion in the context of IOPs, where the simulator also has to answer oracle queries by the verifier. Note that since we consider the notion of unconditional (perfect) zero knowledge, the definition of straightline simulation needs to allow the efficient simulator to work even with inefficient verifiers [GIMS10].

Definition 5. *We say that 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 the concatenation of A's random tape and B's output. (Since A's random tape could be super-polynomially large, B cannot sample it for A and then output it; instead, we restrict B to not see it, and we prepend it to B's output.)*

Recall that an algorithm A is b-query if, on input $\mathbb{x}$, it makes at most $\mathsf{b}(|\mathbb{x}|)$ queries to any oracles it has access to. We are now ready to define zero knowledge IOPs.

Definition 6. *An IOP system (P, V) for a relation $\mathscr{R}$ is perfect zero knowledge (via straightline simulators) against unbounded queries (resp., against query bound b) if there exists a simulator algorithm S such that for every algorithm (resp., b-query algorithm) $\tilde{V}$ and instance-witness pair $(\mathbb{x}, \mathbb{w}) \in \mathscr{R}$, $S^{\tilde{V}}(\mathbb{x})$ and* View $\langle P(\mathbb{x}, \mathbb{w}), \tilde{V}(\mathbb{x})\rangle$ *are identically distributed. Moreover, S must run in time* $\mathrm{poly}(|\mathbb{x}| + \mathsf{q}_{\tilde{V}}(|\mathbb{x}|))$*, where $\mathsf{q}_{\tilde{V}}(\cdot)$ is $\tilde{V}$'s query complexity.*

For zero knowledge against arbitrary polynomial-time adversaries, it suffices for b to be superpolynomial. Note that S's running time need not be polynomial in b (in our constructions it is polylogarithmic in b); rather its running time may be polynomial in the input size $|\mathbb{x}|$ and the *actual* number of queries $\tilde{V}$ makes (as a random variable).

We say that a relation $\mathscr{R}$ belongs to the complexity class **PZK-IOP**$[\mathsf{k}, \mathsf{l}, \mathsf{q}, \varepsilon, \mathsf{tp}, \mathsf{tv}, \mathsf{b}]$ if there is an IOP system for $\mathscr{R}$, with the corresponding parameters, that is perfect zero knowledge with query bound b; also, it belongs to the complexity class **PZK-IOP**$[\mathsf{k}, \mathsf{l}, \mathsf{q}, \varepsilon, \mathsf{tp}, \mathsf{tv}, *]$ if the same is true with unbounded queries.

ZK for IOPs of proximity. We define zero knowledge (via straightline simulators) for IOPs of proximity. It is a straightforward extension of the corresponding notion for PCPs of proximity, introduced in [IW14].

Definition 7. *An IOPP system (P, V) for a relation $\mathscr{R}$ is perfect zero knowledge (via straightline simulators) against unbounded queries (resp., against query bound b) if there exists a simulator algorithm S such that for every algorithm (resp., b-query algorithm) $\tilde{V}$ and instance-witness pair $(\mathbb{x}, \mathbb{w}) \in \mathscr{R}$, the following two random variables are identically distributed:*

$$\left(S^{\tilde{V}, \mathbb{w}}(\mathbb{x}) \ , \ q_S\right) \quad \text{and} \quad \left(\text{View } \langle P(\mathbb{x}, \mathbb{w}), \tilde{V}^{\mathbb{w}}(\mathbb{x})\rangle \ , \ q_{\tilde{V}}\right),$$

where q_S is the number of queries to $\mathbb{w}$ made by S, and $q_{\tilde{V}}$ is the number of queries to $\mathbb{w}$ or to prover messages made by $\tilde{V}$. Moreover, S must run in time $\mathrm{poly}(|\mathbb{x}| + \mathsf{q}_{\tilde{V}}(|\mathbb{x}|))$*, where $\mathsf{q}_{\tilde{V}}(\cdot)$ is $\tilde{V}$'s query complexity.*

We say that a relation $\mathscr{R}$ belongs to the complexity class **PZK-IOPP**$[\mathsf{k}, \mathsf{l}, \mathsf{q}, \varepsilon, \delta, \mathsf{tp}, \mathsf{tv}, \mathsf{b}]$ if there is an IOPP system for $\mathscr{R}$, with the corresponding parameters, that is perfect zero knowledge with query bound b; also, it belongs to

the complexity class **PZK-IOPP**$[\mathsf{k}, \mathsf{l}, \mathsf{q}, \varepsilon, \delta, \mathsf{tp}, \mathsf{tv}, *]$ if the same is true with unbounded queries.

Remark 3. Analogously to [IW14], our definition of zero knowledge for IOPs of proximity requires that the number of queries to $\mathtt{w}$ by S equals the total number of queries (to $\mathtt{w}$ or prover messages) by $\tilde{V}$. Stronger notions are possible: "the number of queries to $\mathtt{w}$ by S equals the number of queries to $\mathtt{w}$ by $\tilde{V}$"; or, even more, "S and $\tilde{V}$ read the same locations of $\mathtt{w}$". The definition above is sufficient for the applications of IOPs of proximity that we consider.

ZK for IPCPs. The definition of perfect zero knowledge (via straightline simulators) for IPCPs follows directly from Definition 6 in Sect. 3.3 because IPCPs are a special case of IOPs. Ditto for IPCPs of proximity, whose perfect zero knowledge definition follows directly from Definition 7 in Sect. 3.3. (For comparison, [GIMS10] define statistical zero knowledge IPCPs, also with straightline simulators.)

3.4 Codes

An error correcting code C is a set of functions $w\colon D \to \Sigma$, where D, Σ are finite sets known as the domain and alphabet; we write $C \subseteq \Sigma^D$. The message length of C is $k := \log_{|\Sigma|} |C|$, its block length is $\ell := |D|$, its rate is $\rho := k/\ell$, its (minimum) distance is $d := \min\{\Delta(w, z) : w, z \in C, w \neq z\}$ when Δ is the (absolute) Hamming distance, and its (minimum) relative distance is $\tau := d/\ell$. At times we write $k(C), \ell(C), \rho(C), d(C), \tau(C)$ to make the code under consideration explicit. All the codes we consider are linear codes, discussed next.

Linearity. A code C is *linear* if Σ is a finite field and C is a Σ-linear space in Σ^D. The dual code of C is the set $C^\perp$ of functions $z\colon D \to \Sigma$ such that, for all $w\colon D \to \Sigma$, $\langle z, w\rangle := \sum_{i \in D} z(i)w(i) = 0$. We denote by $\dim(C)$ the dimension of C; it holds that $\dim(C) + \dim(C^\perp) = \ell$ and $\dim(C) = k$ (dimension equals message length).

Code families. A code family $\mathscr{C} = \{C_{\mathfrak{n}}\}_{\mathfrak{n} \in \{0,1\}^*}$ has domain $D(\cdot)$ and alphabet $\mathbb{F}(\cdot)$ if each code $C_{\mathfrak{n}}$ has domain $D(\mathfrak{n})$ and alphabet $\mathbb{F}(\mathfrak{n})$. Similarly, $\mathscr{C}$ has message length $k(\cdot)$, block length $\ell(\cdot)$, rate $\rho(\cdot)$, distance $d(\cdot)$, and relative distance $\tau(\cdot)$ if each code $C_{\mathfrak{n}}$ has message length $k(\mathfrak{n})$, block length $\ell(\mathfrak{n})$, rate $\rho(\mathfrak{n})$, distance $d(\mathfrak{n})$, and relative distance $\tau(\mathfrak{n})$. We also define $\rho(\mathscr{C}) := \inf_{\mathfrak{n} \in \mathbb{N}} \rho(\mathfrak{n})$ and $\tau(\mathscr{C}) := \inf_{\mathfrak{n} \in \mathbb{N}} \tau(\mathfrak{n})$.

Reed–Solomon codes. The Reed–Solomon (RS) code is the code consisting of evaluations of *univariate* low-degree polynomials: given a field $\mathbb{F}$, subset S of $\mathbb{F}$, and positive integer d with $d \leq |S|$, we denote by $\mathrm{RS}[\mathbb{F}, S, d]$ the linear code consisting of evaluations $w\colon S \to \mathbb{F}$ over S of polynomials in $\mathbb{F}^{<d}[X]$. The code's message length is $k = d$, block length is $\ell = |S|$, rate is $\rho = \frac{d}{|S|}$, and relative distance is $\tau = 1 - \frac{d-1}{|S|}$.

Reed–Muller codes. The Reed–Muller (RM) code is the code consisting of evaluations of *multivariate* low-degree polynomials: given a field $\mathbb{F}$, subset S of $\mathbb{F}$, and positive integers m, d with $d \leq |S|$, we denote by $\mathrm{RM}[\mathbb{F}, S, m, d]$ the linear code consisting of evaluations $w\colon S^m \to \mathbb{F}$ over S^m of polynomials in $\mathbb{F}^{<d}[X_1, \ldots, X_m]$ (i.e., we bound individual degrees rather than their sum). The code's message length is $k = d^m$, block length is $\ell = |S|^m$, rate is $\rho = (\frac{d}{|S|})^m$, and relative distance is $\tau = (1 - \frac{d-1}{|S|})^m$.

4 Succinct Constraint Detection

We introduce the notion of *succinct constraint detection* for linear codes. This notion plays a crucial role in constructing perfect zero knowledge simulators for super-polynomial complexity classes (such as #**P** and **NEXP**), but we believe that this naturally-defined notion is also of independent interest. Given a linear code $C \subseteq \mathbb{F}^D$ we refer to its dual code $C^\perp \subseteq \mathbb{F}^D$ as the *constraint space* of C. The *constraint detection problem* corresponding to a family of linear codes $\mathscr{C} = \{C_\mathfrak{n}\}_\mathfrak{n}$ with domain $D(\cdot)$ and alphabet $\mathbb{F}(\cdot)$ is the following:

$$\text{Given an index } \mathfrak{n} \text{ and subset } I \subseteq D(\mathfrak{n}), \text{ output a basis for } \{z \in D(\mathfrak{n})^I : \forall\, w \in C_\mathfrak{n}, \textstyle\sum_{i \in I} z(i)w(i) = 0\}.^5$$

If $|D(\mathfrak{n})|$ is polynomial in $|\mathfrak{n}|$ and a generating matrix for $C_\mathfrak{n}$ can be found in polynomial time, this problem can be solved in $\mathrm{poly}(|\mathfrak{n}| + |I|)$ time via Gaussian elimination; such an approach was implicitly taken by [BCGV16] to construct a perfect zero knowledge simulator for an IOP for **NP**. However, in our setting, $|D(\mathfrak{n})|$ is *exponential* in $|\mathfrak{n}|$ and $|I|$, and the aforementioned generic solution requires exponential time. With this in mind, we say $\mathscr{C}$ has *succinct constraint detection* if there exists an algorithm that solves the constraint detection problem in $\mathrm{poly}(|\mathfrak{n}| + |I|)$ time when $|D(\mathfrak{n})|$ is *exponential* in $|\mathfrak{n}|$. After defining succinct constraint detection in Sect. 4.1, we proceed as follows.

- In Sect. 4.2, we construct a succinct constraint detector for the family of linear codes comprised of evaluations of partial sums of low-degree polynomials. The construction of the detector exploits derandomization techniques from algebraic complexity theory. We leverage this result to construct a perfect zero knowledge simulator for an IPCP for #**P**; see the full version for details.
- In Sect. 4.3, we construct a succinct constraint detector for the family of evaluations of univariate polynomials concatenated with corresponding BS proximity proofs [BS08]. The construction of the detector exploits the recursive structure of these proximity proofs. We leverage this result to construct a perfect zero knowledge simulator for an IOP for **NEXP**; this simulator can

[5] In fact, the following weaker definition suffices for the applications in our paper: *given an index* $\mathfrak{n}$ *and subset* $I \subseteq D(\mathfrak{n})$, *output* $z \in \mathbb{F}(\mathfrak{n})^I$ *such that* $\sum_{i \in I} z(i)w(i) = 0$ *for all* $w \in C_\mathfrak{n}$, *or* 'independent' *if no such* z *exists*. We achieve the stronger definition, which is also easier to work with.

be interpreted as an analogue of [BCGV16]'s simulator that runs *exponentially faster* and thus enables us to "scale up" from **NP** to **NEXP**; see the full version for details.

Throughout this section we assume familiarity with terminology and notation about codes, introduced in Sect. 3.4. We assume for simplicity that $|\mathbb{n}|$, the number of bits used to represent $\mathbb{n}$, is at least $\log D(\mathbb{n}) + \log \mathbb{F}(\mathbb{n})$; if this does not hold, then one can replace $|\mathbb{n}|$ with $|\mathbb{n}| + \log D(\mathbb{n}) + \log \mathbb{F}(\mathbb{n})$ throughout the section.

Remark 4 (sparse representation). In this section we make statements about vectors v in $\mathbb{F}^D$ where the cardinality of the domain D may be super-polynomial. When such statements are computational in nature, we assume that v is not represented as a list of $|D|$ field elements (which requires $\Omega(|D| \log |\mathbb{F}|)$ bits) but, instead, assume that v is represented as a list of the elements in $\mathrm{supp}(v)$ (and each element comes with its index in D); this *sparse* representation only requires $\Omega(|\mathrm{supp}(v)| \cdot (\log |D| + \log |\mathbb{F}|))$ bits.

4.1 Definition of Succinct Constraint Detection

Formally define the notion of a *constraint detector*, and the notion of *succinct constraint detection.*

Definition 8. *Let $\mathscr{C} = \{C_{\mathbb{n}}\}_{\mathbb{n}}$ be a linear code family with domain $D(\cdot)$ and alphabet $\mathbb{F}(\cdot)$. A* **constraint detector** *for $\mathscr{C}$ is an algorithm that, on input an index $\mathbb{n}$ and subset $I \subseteq D(\mathbb{n})$, outputs a basis for the space*

$$\left\{ z \in D(\mathbb{n})^I : \forall\, w \in C_{\mathbb{n}}, \sum_{i \in I} z(i) w(i) \right\}.$$

We say that $\mathscr{C}$ has $T(\cdot,\cdot)$ **-time constraint detection** *if there exists a detector for $\mathscr{C}$ running in time $T(\mathbb{n}, \ell)$; we also say that $\mathscr{C}$ has* **succinct constraint detection** *if it has $\mathrm{poly}(|\mathbb{n}| + \ell)$-time constraint detection.*

A constraint detector induces a corresponding probabilistic algorithm for 'simulating' answers to queries to a random codeword; this is captured by the following lemma, the proof of which is in the full version. We shall use such probabilistic algorithms in the construction of perfect zero knowledge simulators.

Lemma 1. *Let $\mathscr{C} = \{C_{\mathbb{n}}\}_{\mathbb{n}}$ be a linear code family with domain $D(\cdot)$ and alphabet $\mathbb{F}(\cdot)$ that has $T(\cdot,\cdot)$-time constraint detection. Then there exists a probabilistic algorithm $\mathcal{A}$ such that, for every index $\mathbb{n}$, set of pairs $S = \{(\alpha_1, \beta_1), \ldots, (\alpha_\ell, \beta_\ell)\} \subseteq D(\mathbb{n}) \times \mathbb{F}(\mathbb{n})$, and pair $(\alpha, \beta) \in D(\mathbb{n}) \times \mathbb{F}(\mathbb{n})$,*

$$\Pr\Big[\mathcal{A}(\mathbb{n}, S, \alpha) = \beta\Big] = \Pr_{w \leftarrow C_{\mathbb{n}}}\left[w(\alpha) = \beta \;\middle|\; \begin{array}{c} w(\alpha_1) = \beta_1 \\ \vdots \\ w(\alpha_\ell) = \beta_\ell \end{array} \right].$$

Moreover $\mathcal{A}$ runs in time $T(\mathbb{n}, \ell) + \mathrm{poly}(\log |\mathbb{F}(\mathbb{n})| + \ell)$.

For the purposes of *constructing* a constraint detector, the sufficient condition given in Lemma 2 below is sometimes easier to work with. To state it we need to introduce two ways of restricting a code, and explain how these restrictions interact with taking duals; the interplay between these is delicate (see Remark 5).

Definition 9. *Given a linear code $C \subseteq \mathbb{F}^D$ and a subset $I \subseteq D$, we denote by (i) $C_{\subseteq I}$ the set consisting of the codewords $w \in C$ for which* $\mathrm{supp}(w) \subseteq I$*, and (ii) $C|_I$ the restriction to I of codewords $w \in C$.*

Note that $C_{\subseteq I}$ and $C|_I$ are *different notions*. Consider for example the 1-dimensional linear code $C = \{00, 11\}$ in $\mathbb{F}_2^{\{1,2\}}$ and the subset $I = \{1\}$: it holds that $C_{\subseteq I} = \{00\}$ and $C|_I = \{0, 1\}$. In particular, codewords in $C_{\subseteq I}$ are defined over D, while codewords in $C|_I$ are defined over I. Nevertheless, throughout this section, we sometimes compare vectors defined over different domains, with the implicit understanding that the comparison is conducted over the union of the relevant domains, by filling in zeros in the vectors' undefined coordinates. For example, we may write $C_{\subseteq I} \subseteq C|_I$ to mean that $\{00\} \subseteq \{00, 10\}$ (the set obtained from $\{0, 1\}$ after filling in the relevant zeros).

Claim. Let C be a linear code with domain D and alphabet $\mathbb{F}$. For every $I \subseteq D$,

$$(C|_I)^\perp = (C^\perp)_{\subseteq I},$$

that is,

$$\left\{ z \in D(\mathfrak{n})^I : \forall\, w \in C_{\mathfrak{n}}, \sum_{i \in I} z(i)w(i) \right\} = \left\{ z \in C_{\mathfrak{n}}^\perp : \mathrm{supp}(z) \subseteq I \right\}.$$

Proof. For the containment $(C^\perp)_{\subseteq I} \subseteq (C|_I)^\perp$: if $z \in C^\perp$ and $\mathrm{supp}(z) \subseteq I$ then z lies in the dual of $C|_I$ because it suffices to consider the subdomain I for determining duality. For the reverse containment $(C^\perp)_{\subseteq I} \supseteq (C|_I)^\perp$: if $z \in (C|_I)^\perp$ then $\mathrm{supp}(z) \subseteq I$ (by definition) so that $\langle z, w \rangle = \langle z, w|_I \rangle$ for every $w \in C$, and the latter inner product equals 0 because z is in the dual of $C|_I$; in sum z is dual to (all codewords in) C and its support is contained in I, so z belongs to $(C^\perp)_{\subseteq I}$, as claimed.

Observe that Claim 4.1 tells us the constraint detection is equivalent to determining a basis of $(C_{\mathfrak{n}}|_I)^\perp = (C_{\mathfrak{n}}^\perp)_{\subseteq I}$. The following lemma asserts that if, given a subset $I \subseteq D$, we can find a set of constraints W in $C^\perp$ that spans $(C^\perp)_{\subseteq I}$ then we can solve the constraint detection problem for C; see the full version for a proof.

Lemma 2. *Let $\mathscr{C} = \{C_{\mathfrak{n}}\}_{\mathfrak{n}}$ be a linear code family with domain $D(\cdot)$ and alphabet $\mathbb{F}(\cdot)$. If there exists an algorithm that, on input an index $\mathfrak{n}$ and subset $I \subseteq D(\mathfrak{n})$, outputs in* $\mathrm{poly}(|\mathfrak{n}| + |I|)$ *time a subset $W \subseteq \mathbb{F}(\mathfrak{n})^{D(\mathfrak{n})}$ (in sparse representation) with $(C_{\mathfrak{n}}^\perp)_{\subseteq I} \subseteq \mathrm{span}(W) \subseteq C_{\mathfrak{n}}^\perp$, then $\mathscr{C}$ has succinct constraint detection.*

Remark 5. The following operations do *not* commute: (i) expanding the domain via zero padding (for the purpose of comparing vectors over different domains), and (ii)taking the dual of the code. Consider for example the code $C = \{0\} \subseteq \mathbb{F}_2^{\{1\}}$: its dual code is $C^\perp = \{0, 1\}$ and, when expanded to $\mathbb{F}_2^{\{1,2\}}$, the dual code is expanded to $\{(0,0),(1,0)\}$; yet, when C is expanded to $\mathbb{F}_2^{\{1,2\}}$ it produces the code $\{(0,0)\}$ and its dual code is $\{(0,0),(1,0),(0,1),(1,1)\}$. To resolve ambiguities (when asserting an equality as in Claim 4.1), we adopt the convention that expansion is done *always last* (namely, as late as possible without having to compare vectors over different domains).

4.2 Partial Sums of Low-Degree Polynomials

We show that evaluations of partial sums of low-degree polynomials have succinct constraint detection (see Definition 8). In the following, $\mathbb{F}$ is a finite field, m, d are positive integers, and H is a subset of $\mathbb{F}$; also, $\mathbb{F}^{<d}[X_1, \ldots, X_m]$ denotes the subspace of $\mathbb{F}[X_1, \ldots, X_m]$ consisting of those polynomials with individual degrees less than d. Moreover, given $Q \in \mathbb{F}^{<d}[X_1, \ldots, X_m]$ and $\boldsymbol{\alpha} \in \mathbb{F}^{\leq m}$ (vectors over $\mathbb{F}$ of length at most m), we define $Q(\boldsymbol{\alpha}) := \sum_{\boldsymbol{\gamma} \in H^{m-|\boldsymbol{\alpha}|}} Q(\boldsymbol{\alpha}, \boldsymbol{\gamma})$, i.e., the answer to a query that specifies only a suffix of the variables is the sum of the values obtained by letting the remaining variables range over H. We begin by defining the code that we study, which extends the Reed–Muller code (see Sect. 3.4) with partial sums.

Definition 10. *We denote by* $\Sigma\mathrm{RM}[\mathbb{F}, m, d, H]$ *the linear code that comprises evaluations of partial sums of polynomials in* $\mathbb{F}^{<d}[X_1, \ldots, X_m]$*; more precisely,* $\Sigma\mathrm{RM}[\mathbb{F}, m, d, H] := \{w_Q\}_{Q \in \mathbb{F}^{<d}[X_1, \ldots, X_m]}$ *where* $w_Q \colon \mathbb{F}^{\leq m} \to \mathbb{F}$ *is the function defined by* $w_Q(\boldsymbol{\alpha}) := \sum_{\boldsymbol{\gamma} \in H^{m-|\boldsymbol{\alpha}|}} Q(\boldsymbol{\alpha}, \boldsymbol{\gamma})$ *for each* $\boldsymbol{\alpha} \in \mathbb{F}^{\leq m}$.[6] *We denote by* $\Sigma\mathrm{RM}$ *the linear code family indexed by tuples* $\mathfrak{n} = (\mathbb{F}, m, d, H)$ *and where the* $\mathfrak{n}$*-th code equals* $\Sigma\mathrm{RM}[\mathbb{F}, m, d, H]$*. (We represent indices* $\mathfrak{n}$ *so to ensure that* $|\mathfrak{n}| = \Theta(\log |\mathbb{F}| + m + d + |H|)$*.)*

We prove that the linear code family $\Sigma\mathrm{RM}$ has succinct constraint detection:

Theorem 5 (formal statement of 3). $\Sigma\mathrm{RM}$ *has* $\mathrm{poly}(\log |\mathbb{F}| + m + d + |H| + \ell)$*-time constraint detection.*

Combined with Lemma 1, the theorem above implies that there exists a probabilistic polynomial-time algorithm for answering queries to a codeword sampled at random from $\Sigma\mathrm{RM}$, as captured by the following corollary.

[6] Note that $\Sigma\mathrm{RM}[\mathbb{F}, m, d, H]$ is indeed linear: for every $w_{Q_1}, w_{Q_2} \in \Sigma\mathrm{RM}[\mathbb{F}, m, d, H]$, $a_1, a_2 \in \mathbb{F}$, and $\boldsymbol{\alpha} \in \mathbb{F}^{\leq m}$, it holds that $a_1 w_{Q_1}(\boldsymbol{\alpha}) + a_2 w_{Q_2}(\boldsymbol{\alpha}) = a_1 \sum_{\boldsymbol{\gamma} \in H^{m-|\boldsymbol{\alpha}|}} Q_1(\boldsymbol{\alpha}, \boldsymbol{\gamma}) + a_2 \sum_{\boldsymbol{\gamma} \in H^{m-|\boldsymbol{\alpha}|}} Q_2(\boldsymbol{\alpha}, \boldsymbol{\gamma}) = \sum_{\boldsymbol{\gamma} \in H^{m-|\boldsymbol{\alpha}|}} (a_1 Q_1 + a_2 Q_2)(\boldsymbol{\alpha}, \boldsymbol{\gamma}) = w_{a_1 Q_1 + a_2 Q_2}(\boldsymbol{\alpha})$. But $w_{a_1 Q_1 + a_2 Q_2} \in \Sigma\mathrm{RM}[\mathbb{F}, m, d, H]$, since $\mathbb{F}^{<d}[X_1, \ldots, X_m]$ is a linear space.

Corollary 1. *There exists a probabilistic algorithm $\mathcal{A}$ such that, for every finite field $\mathbb{F}$, positive integers m, d, subset H of $\mathbb{F}$, subset $S = \{(\alpha_1, \beta_1), \ldots, (\alpha_\ell, \beta_\ell)\} \subseteq \mathbb{F}^{\leq m} \times \mathbb{F}$, and $(\alpha, \beta) \in \mathbb{F}^{\leq m} \times \mathbb{F}$,*

$$\Pr\Big[\mathcal{A}(\mathbb{F}, m, d, H, S, \alpha) = \beta\Big] = \Pr_{R \leftarrow \mathbb{F}^{<d}[X_1, \ldots, X_m]}\left[R(\alpha) = \beta \;\middle|\; \begin{array}{c} R(\alpha_1) = \beta_1 \\ \vdots \\ R(\alpha_\ell) = \beta_\ell \end{array}\right].$$

Moreover $\mathcal{A}$ runs in time $\mathrm{poly}(\log|\mathbb{F}| + m + d + |H| + \ell)$.

We sketch the proof of Theorem 5, for the simpler case where the code is $\mathrm{RM}[\mathbb{F}, m, d, H]$ (i.e., without partial sums). We can view a polynomial $Q \in \mathbb{F}^{<d}[X_1, \ldots, X_m]$ as a vector over the monomial basis, with an entry for each possible monomial $X_1^{i_1} \ldots X_m^{i_m}$ (with $0 \leq i_1, \ldots, i_m < d$) containing the corresponding coefficient. The evaluation of Q at a point $\boldsymbol{\alpha} \in \mathbb{F}^m$ then equals the inner product of this vector with the vector $\phi_{\boldsymbol{\alpha}}$, in the same basis, whose entry for $X_1^{i_1} \ldots X_m^{i_m}$ is equal to $\alpha_1^{i_1} \ldots \alpha_m^{i_m}$. Given $\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_\ell$, we could use Gaussian elimination on $\phi_{\boldsymbol{\alpha}_1}, \ldots, \phi_{\boldsymbol{\alpha}_\ell}$ to check for linear dependencies, which would be equivalent to constraint detection for $\mathrm{RM}[\mathbb{F}, m, d, H]$.

However, we cannot afford to explicitly write down $\phi_{\boldsymbol{\alpha}}$, because it has d^m entries. Nevertheless, we can still implicitly check for linear dependencies, and we do so by reducing the problem, by building on and extending ideas of [BW04], to computing the nullspace of a certain set of polynomials, which can be solved via an algorithm of [RS05] (see also [Kay10]). The idea is to encode the entries of these vectors via a succinct description: a polynomial $\Phi_{\boldsymbol{\alpha}}$ whose coefficients (after expansion) are the entries of $\phi_{\boldsymbol{\alpha}}$. In our setting this polynomial has the particularly natural form:

$$\Phi_{\boldsymbol{\alpha}}(\boldsymbol{X}) := \prod_{i=1}^{m}(1 + \alpha_i X_i + \alpha_i^2 X_i^2 + \cdots + \alpha_i^{d-1} X_i^{d-1});$$

note that the coefficient of each monomial equals its corresponding entry in $\phi_{\boldsymbol{\alpha}}$. Given this representation we can use standard polynomial identity testing techniques to find linear dependencies between these polynomials, which corresponds to linear dependencies between the original vectors. Crucially, we cannot afford any mistake, even with exponentially small probability, when looking for linear dependencies for otherwise we would not achieve perfect simulation; this is why the techniques we leverage rely on derandomization. We now proceed with the full proof.

Proof (Proof of Theorem 5). We first introduce some notation. Define $[< d] := \{0, \ldots, d-1\}$. For vectors $\boldsymbol{\alpha} \in \mathbb{F}^m$ and $\boldsymbol{a} \in [< d]^m$, we define $\boldsymbol{\alpha}^{\boldsymbol{a}} := \prod_{i=1}^m \alpha_i^{a_i}$; similarly, for variables $\boldsymbol{X} = (X_1, \ldots, X_m)$, we define $\boldsymbol{X}^{\boldsymbol{a}} := \prod_{i=1}^m X_i^{a_i}$.

We identify $\Sigma\mathrm{RM}[\mathbb{F}, m, d, H]$ with $\mathbb{F}^{[<d]^m}$; a codeword w_Q then corresponds to a vector $\boldsymbol{Q}$ whose $\boldsymbol{a}$-th entry is the coefficient of the monomial $\boldsymbol{X}^{\boldsymbol{a}}$ in Q. For $\boldsymbol{\alpha} \in \mathbb{F}^{\leq m}$, let

$$\phi_{\boldsymbol{\alpha}} := \left(\boldsymbol{\alpha}^{\boldsymbol{a}} \sum_{\boldsymbol{\gamma} \in H^{m-|\boldsymbol{\alpha}|}} \boldsymbol{\gamma}^{\boldsymbol{b}} \right)_{\boldsymbol{a} \in [<d]^{|\boldsymbol{\alpha}|},\, \boldsymbol{b} \in [<d]^{m-|\boldsymbol{\alpha}|}}.$$

We can also view $\phi_{\boldsymbol{\alpha}}$ as a vector in $\mathbb{F}^{[<d]^m}$ by merging the indices, so that, for all $\boldsymbol{\alpha} \in \mathbb{F}^{\leq m}$ and $w_Q \in \Sigma\mathrm{RM}[\mathbb{F}, m, d, H]$,

$$\begin{aligned} w_Q(\boldsymbol{\alpha}) &= \sum_{\boldsymbol{\gamma} \in H^{m-|\boldsymbol{\alpha}|}} Q(\boldsymbol{\alpha}, \boldsymbol{\gamma}) = \sum_{\boldsymbol{\gamma} \in H^{m-|\boldsymbol{\alpha}|}} \sum_{\boldsymbol{a} \in [<d]^{|\boldsymbol{\alpha}|}} \sum_{\boldsymbol{b} \in [<d]^{m-|\boldsymbol{\alpha}|}} \boldsymbol{Q}_{\boldsymbol{a},\boldsymbol{b}} \cdot \boldsymbol{\alpha}^{\boldsymbol{a}} \boldsymbol{\gamma}^{\boldsymbol{b}} \\ &= \sum_{\boldsymbol{a} \in [<d]^{|\boldsymbol{\alpha}|}} \sum_{\boldsymbol{b} \in [<d]^{m-|\boldsymbol{\alpha}|}} \boldsymbol{Q}_{\boldsymbol{a},\boldsymbol{b}} \cdot \boldsymbol{\alpha}^{\boldsymbol{a}} \sum_{\boldsymbol{\gamma} \in H^{m-|\boldsymbol{\alpha}|}} \boldsymbol{\gamma}^{\boldsymbol{b}} = \langle \boldsymbol{Q}, \phi_{\boldsymbol{\alpha}} \rangle. \end{aligned}$$

Hence for every $\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_\ell, \boldsymbol{\alpha} \in \mathbb{F}^{\leq m}$ and $a_1, \ldots, a_\ell \in \mathbb{F}$, the following statements are equivalent (i)$w(\boldsymbol{\alpha}) = \sum_{i=1}^{\ell} a_i w(\boldsymbol{\alpha}_i)$ for all $w \in \Sigma\mathrm{RM}[\mathbb{F}, m, d, H]$; (ii)$\langle \boldsymbol{f}, \phi_{\boldsymbol{\alpha}} \rangle = \sum_{i=1}^{\ell} a_i \langle \boldsymbol{f}, \phi_{\boldsymbol{\alpha}_i} \rangle$ for all $\boldsymbol{f} \in \mathbb{F}^{[<d]^m}$ (iii)$\phi_{\boldsymbol{\alpha}} = \sum_{i=1}^{\ell} a_i \phi_{\boldsymbol{\alpha}_i}$. We deduce that constraint detection for $\Sigma\mathrm{RM}[\mathbb{F}, m, d, H]$ is equivalent to the problem of finding $a_1, \ldots, a_\ell \in \mathbb{F}$ such that $\phi_{\boldsymbol{\alpha}} = \sum_{i=1}^{\ell} a_i \phi_{\boldsymbol{\alpha}_i}$, or returning 'independent' if no such $a_1, \ldots, a_\ell$ exist.

However, the dimension of the latter vectors is d^m, which may be much larger than $\mathrm{poly}(\log |\mathbb{F}| + m + d + |H| + \ell)$, and so we cannot afford to "explicitly" solve the $\ell \times d^m$ linear system. Instead, we "succinctly" solve it, by taking advantage of the special structure of the vectors, as we now describe. For $\boldsymbol{\alpha} \in \mathbb{F}^m$, define the polynomial

$$\Phi_{\boldsymbol{\alpha}}(\boldsymbol{X}) := \prod_{i=1}^{m} (1 + \alpha_i X_i + \alpha_i^2 X_i^2 + \cdots + \alpha_i^{d-1} X_i^{d-1}).$$

Note that, while the above polynomial is computable via a small arithmetic circuit, its coefficients (once expanded over the monomial basis) correspond to the entries of the vector $\phi_{\boldsymbol{\alpha}}$. More generally, for $\boldsymbol{\alpha} \in \mathbb{F}^{\leq m}$, we define the polynomial

$$\begin{aligned} \Phi_{\boldsymbol{\alpha}}(\boldsymbol{X}) := &\left(\prod_{i=1}^{|\boldsymbol{\alpha}|} (1 + \alpha_i X_i + \cdots + \alpha_i^{d-1} X_i^{d-1}) \right) \\ &\left(\prod_{i=1}^{m-|\boldsymbol{\alpha}|} \sum_{\gamma \in H} (1 + \gamma X_{i+|\boldsymbol{\alpha}|} + \cdots + \gamma^{d-1} X_{i+|\boldsymbol{\alpha}|}^{d-1}) \right). \end{aligned}$$

Note that $\Phi_{\boldsymbol{\alpha}}$ is a product of univariate polynomials. To see that the above does indeed represent $\phi_{\boldsymbol{\alpha}}$, we rearrange the expression as follows:

$$\begin{aligned}\Phi_{\boldsymbol{\alpha}}(\boldsymbol{X}) &= \left(\prod_{i=1}^{|\boldsymbol{\alpha}|}(1+\alpha_i X_i+\cdots+\alpha_i^{d-1}X_i^{d-1})\right)\\ &\quad\left(\sum_{\boldsymbol{\gamma}\in H^{m-|\boldsymbol{\alpha}|}}\prod_{i=1}^{m-|\boldsymbol{\alpha}|}(1+\gamma_i X_{i+|\boldsymbol{\alpha}|}+\cdots+\gamma_i^{d-1}X_{i+|\boldsymbol{\alpha}|}^{d-1})\right)\\ &= \Phi_{\boldsymbol{\alpha}}(X_1,\ldots,X_{|\boldsymbol{\alpha}|})\left(\sum_{\boldsymbol{\gamma}\in H^{m-|\boldsymbol{\alpha}|}}\Phi_{\boldsymbol{\gamma}}(X_{|\boldsymbol{\alpha}|+1},\ldots,X_m)\right);\end{aligned}$$

indeed, the coefficient of $\boldsymbol{X}^{\boldsymbol{a},\boldsymbol{b}}$ for $\boldsymbol{a}\in[<d]^{|\boldsymbol{\alpha}|}$ and $\boldsymbol{b}\in[<d]^{m-|\boldsymbol{\alpha}|}$ is $\boldsymbol{\alpha}^{\boldsymbol{a}}\sum_{\boldsymbol{\gamma}\in H^{m-|\boldsymbol{\alpha}|}}\boldsymbol{\gamma}^{\boldsymbol{b}}$, as required.

Thus, to determine whether $\phi_\alpha\in\mathrm{span}(\phi_{\alpha_1},\ldots,\phi_{\alpha_\ell})$, it suffices to determine whether $\Phi_\alpha\in\mathrm{span}(\Phi_{\alpha_1},\ldots,\Phi_{\alpha_\ell})$. In fact, the linear dependencies are in correspondence: for $a_1,\ldots,a_\ell\in\mathbb{F}$, $\phi_\alpha=\sum_{i=1}^{\ell}a_i\phi_{\alpha_i}$ if and only if $\Phi_\alpha=\sum_{i=1}^{\ell}a_i\Phi_{\alpha_i}$. Crucially, each Φ_{α_i} is not only in $\mathbb{F}^{<d}[X_1,\ldots,X_m]$ but is a product of m univariate polynomials each represented via an $\mathbb{F}$-arithmetic circuit of size $\mathrm{poly}(|H|+d)$. We leverage this special structure and solve the above problem by relying on an algorithm of [RS05] that computes the nullspace for such polynomials (see also [Kay10]), as captured by the lemma below;[7] for completeness, we provide an elementary proof of the lema in the full version.

Lemma 3. *There exists a deterministic algorithm $\mathcal{D}$ such that, on input a vector of m-variate polynomials $\boldsymbol{Q}=(Q_1,\ldots,Q_\ell)$ over $\mathbb{F}$ where each polynomial has the form $Q_k(\boldsymbol{X})=\prod_{i=1}^{m}Q_{k,i}(X_i)$ and each $Q_{k,i}$ is univariate of degree less than d with $d\le|\mathbb{F}|$ and represented via an $\mathbb{F}$-arithmetic circuit of size s, outputs a basis for the linear space $\boldsymbol{Q}^{\perp}:=\{(a_1,\ldots,a_\ell)\in\mathbb{F}^\ell:\sum_{k=1}^{\ell}a_kQ_k\equiv 0\}$. Moreover, $\mathcal{D}$ runs in $\mathrm{poly}(\log|\mathbb{F}|+m+d+s+\ell)$ time.*

The above lemma immediately provides a way to construct a constraint detector for ΣRM: given as input an index $\mathfrak{n}=(\mathbb{F},m,d,H)$ and a subset $I\subseteq D(\mathfrak{n})$, we construct the arithmetic circuit Φ_α for each $\alpha\in I$, and then run the algorithm $\mathcal{D}$ on vector of circuits $(\Phi_\alpha)_{\alpha\in I}$, and directly output $\mathcal{D}$'s result. The lemma follows.

4.3 Univariate Polynomials with BS Proximity Proofs

We show that evaluations of univariate polynomials concatenated with corresponding BS proximity proofs [BS08] have succinct constraint detection (see

[7] One could use polynomial identity testing to solve the above problem in probabilistic polynomial time; see [Kay10, Lemma 8]. However, due to a nonzero probability of error, this suffices only to achieve statistical zero knowledge, but *does not suffice to achieve perfect zero knowledge.*

Definition 8). Recall that the Reed–Solomon code (see Sect. 3.4) is not locally testable, but one can test proximity to it with the aid of the quasilinear-size proximity proofs of Ben-Sasson and Sudan [BS08]. These latter apply when low-degree univariate polynomials are evaluated over *linear spaces*, so from now on we restrict our attention to Reed–Solomon codes of this form. More precisely, we consider Reed–Solomon codes $\mathrm{RS}[\mathbb{F}, L, d]$ where $\mathbb{F}$ is an extension field of a base field $\mathbb{K}$, L is a $\mathbb{K}$-linear subspace in $\mathbb{F}$, and $d = |L| \cdot |\mathbb{K}|^{-\mu}$ for some $\mu \in \mathbb{N}^+$. We then denote by $\text{BS-RS}[\mathbb{K}, \mathbb{F}, L, \mu, k]$ the code obtained by concatenating codewords in $\mathrm{RS}[\mathbb{F}, L, |L| \cdot |\mathbb{K}|^{-\mu}]$ with corresponding BS proximity proofs whose recursion terminates at "base dimension" $k \in \{1, \ldots, \dim(L)\}$ (for a formal definition of these, see the full version); typically $\mathbb{K}, \mu, k$ are fixed to certain constants (e.g., [BS08] fixes them to $\mathbb{F}_2, 3, 1$, respectively) but below we state the cost of constraint detection in full generality. The linear code family BS-RS is indexed by tuples $\mathtt{n} = (\mathbb{K}, \mathbb{F}, L, \mu, k)$ and the $\mathtt{n}$-th code is $\text{BS-RS}[\mathbb{K}, \mathbb{F}, L, \mu, k]$, and our result about BS-RS is the following:

Theorem 6 (formal statement of 4). BS-RS *has* $\mathrm{poly}(\log |\mathbb{F}| + \dim(L) + |\mathbb{K}|^{\mu} + \ell)$*-time constraint detection.*

The proof of the above theorem is technically involved, and we refer the reader to the full version for details.

The role of code covers. We are interested in succinct constraint detection: solving the constraint detection problem for certain code families with exponentially-large domains (such as BS-RS). We now build some intuition about how code covers can, in some cases, facilitate this.

Consider the simple case where the code $C \subseteq \mathbb{F}^D$ is a direct sum of many small codes: there exists $S = \{(\tilde{D}_j, \tilde{C}_j)\}_j$ such that $D = \cup_j \tilde{D}_j$ and $C = \oplus_j \tilde{C}_j$ where, for each j, $\tilde{C}_j$ is a linear code in $\mathbb{F}^{\tilde{D}_j}$ and the subdomain $\tilde{D}_j$ is small and disjoint from other subdomains. The detection problem for this case can be solved efficiently: use the generic approach of Gaussian elimination independently on each subdomain $\tilde{D}_j$.

Next consider a more general case where the subdomains are not necessarily disjoint: there exists $S = \{(\tilde{D}_j, \tilde{C}_j)\}_j$ as above but we do not require that the $\tilde{D}_j$ form a partition of D; we say that each $(\tilde{D}_j, \tilde{C}_j)$ is a *local view* of C because $\tilde{D}_j \subseteq D$ and $\tilde{C}_j = C|_{\tilde{D}_j}$, and we say that S is a *code cover* of C. Now suppose that for each j there exists an efficient constraint detector for $\tilde{C}_j$ (which is defined on $\tilde{D}_j$); in this case, the detection problem can be solved efficiently at least for those subsets I that are contained in $\tilde{D}_j$ for some j. Generalizing further, we see that we can efficiently solve constraint detection for a code C if there is a cover $S = \{(\tilde{D}_j, \tilde{C}_j)\}_j$ such that, given a subset $I \subseteq D$, (i) I is contained in some subdomain $\tilde{D}_j$, and (ii) constraint detection for $\tilde{C}_j$ can be solved efficiently.

We build on the above ideas to derive analogous statements for recursive code covers, which arise naturally in the case of BS-RS. But note that recursive constructions are common in the PCP literature, and we believe that our cover-based techniques are of independent interest as, e.g., they are applicable to *other* PCPs, including [BFLS91, AS98].

Acknowledgments. Work of E. Ben-Sasson, A. Gabizon, and M. Riabzev was supported by the Israel Science Foundation (grant 1501/14). Work of A. Chiesa and N. Spooner was partially supported in part by the UC Berkeley Center for Long-Term Cybersecurity. Work of M. A. Forbes was supported by the NSF, including NSF CCF-1617580, and the DARPA Safeware program; it was also partially completed when the author was at Princeton University, supported by the Princeton Center for Theoretical Computer Science.

References

[ALM+98] Arora, S., Lund, C., Motwani, R., Sudan, M., Szegedy, M.: Proof verification, the hardness of approximation problems. J. ACM **45**(3), 501–555 (1998). Preliminary version in FOCS 1992

[AR16] Applebaum, B., Raykov, P.: On the relationship between statistical zero-knowledge and statistical randomized encodings. In: Robshaw, M., Katz, J. (eds.) CRYPTO 2016. LNCS, vol. 9816, pp. 449–477. Springer, Heidelberg (2016). doi:10.1007/978-3-662-53015-3_16

[AS98] Arora, S., Safra, S.: Probabilistic checking of proofs: a new characterization of NP. J. ACM **45**(1), 70–122 (1998). Preliminary version in FOCS 1992

[Bab85] Babai, L.: Trading group theory for randomness. In: Proceedings of the 17th Annual ACM Symposium on Theory of Computing, STOC 1985, pp. 421–429 (1985)

[BCG+17] Ben-Sasson, E., Chiesa, A., Gabizon, A., Riabzev, M., Spooner, N.: Interactive oracle proofs with constant rate and query complexity. In: Proceedings of the 44th International Colloquium on Automata, Languages and Programming, ICALP 2017, pp. 40:1–40:15 (2017)

[BCGV16] Ben-Sasson, E., Chiesa, A., Gabizon, A., Virza, M.: Quasi-linear size zero knowledge from linear-algebraic PCPs. In: Kushilevitz, E., Malkin, T. (eds.) TCC 2016. LNCS, vol. 9563, pp. 33–64. Springer, Heidelberg (2016). doi:10.1007/978-3-662-49099-0_2

[BCS16] Ben-Sasson, E., Chiesa, A., Spooner, N.: Interactive oracle proofs. In: Hirt, M., Smith, A. (eds.) TCC 2016. LNCS, vol. 9986, pp. 31–60. Springer, Heidelberg (2016). doi:10.1007/978-3-662-53644-5_2

[BFL91] Babai, L., Fortnow, L., Lund, C.: Non-deterministic exponential time has two-prover interactive protocols. Comput. Complexity **1**, 3–40 (1991). Preliminary version appeared in FOCS 1990

[BFLS91] Babai, L., Fortnow, L., Levin, L.A., Szegedy, M.: Checking computations in polylogarithmic time. In: Proceedings of the 23rd Annual ACM Symposium on Theory of Computing, STOC 1991, pp. 21–32 (1991)

[BGG+88] Ben-Or, M., Goldreich, O., Goldwasser, S., Håstad, J., Kilian, J., Micali, S., Rogaway, P.: Everything provable is provable in zero-knowledge. In: Goldwasser, S. (ed.) CRYPTO 1988. LNCS, vol. 403, pp. 37–56. Springer, New York (1990). doi:10.1007/0-387-34799-2_4

[BGH+06] Ben-Sasson, E., Goldreich, O., Harsha, P., Sudan, M., Vadhan, S.P.: Robust PCPs of proximity, shorter PCPs, and applications to coding. SIAM J. Comput. **36**(4), 889–974 (2006)

[BGK+10] Ben-Sasson, E., Guruswami, V., Kaufman, T., Sudan, M., Viderman, M.: Locally testable codes require redundant testers. SIAM J. Comput. **39**(7), 3230–3247 (2010)

[BGKW88] Ben-Or, M., Goldwasser, S., Kilian, J., Wigderson, A.: Multi-prover interactive proofs: how to remove intractability assumptions. In: Proceedings of the 20th Annual ACM Symposium on Theory of Computing, STOC 1988, pp. 113–131 (1988)

[BGW88] Ben-Or, M., Goldwasser, S., Wigderson, A.: Completeness theorems for non-cryptographic fault-tolerant distributed computation. In: Proceedings of the 20th Annual ACM Symposium on Theory of Computing, STOC 1988, pp. 1–10 (1988)

[BHR05] Ben-Sasson, E., Harsha, P., Raskhodnikova, S.: Some 3CNF properties are hard to test. SIAM J. Comput. **35**(1), 1–21 (2005)

[BHZ87] Boppana, R.B., Håstad, J., Zachos, S.: Does co-NP have short interactive proofs? Inf. Process. Lett. **25**(2), 127–132 (1987)

[BM88] Babai, L., Moran, S.: Arthur-merlin games: a randomized proof system, and a hierarchy of complexity classes. J. Comput. Syst. Sci. **36**(2), 254–276 (1988)

[BS08] Ben-Sasson, E., Sudan, M.: Short PCPs with polylog query complexity. SIAM J. Comput. **38**(2), 551–607 (2008). Preliminary version appeared in STOC 2005

[BW04] Bogdanov, A., Wee, H.: A stateful implementation of a random function supporting parity queries over hypercubes. In: Jansen, K., Khanna, S., Rolim, J.D.P., Ron, D. (eds.) APPROX/RANDOM -2004. LNCS, vol. 3122, pp. 298–309. Springer, Heidelberg (2004). doi:10.1007/978-3-540-27821-4_27

[CFS17] Chiesa, A., Forbes, M.A., Spooner, N.: A zero knowledge sumcheck and its applications. Cryptology ePrint Archive, Report 2017/305 (2017)

[DFK+92] Dwork, C., Feige, U., Kilian, J., Naor, M., Safra, M.: Low communication 2-prover zero-knowledge proofs for NP. In: Brickell, E.F. (ed.) CRYPTO 1992. LNCS, vol. 740, pp. 215–227. Springer, Heidelberg (1993). doi:10.1007/3-540-48071-4_15

[DR04] Dinur, I., Reingold, O.: Assignment testers: towards a combinatorial proof of the PCP theorem. In: Proceedings of the 45th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2004, pp. 155–164 (2004)

[DS98] Dwork, C., Sahai, A.: Concurrent zero-knowledge: reducing the need for timing constraints. In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, pp. 442–457. Springer, Heidelberg (1998). doi:10.1007/BFb0055746

[FGL+96] Feige, U., Goldwasser, S., Lovász, L., Safra, S., Szegedy, M.: Interactive proofs, the hardness of approximating cliques. J. ACM **43**(2), 268–292 (1996). Preliminary version in FOCS 1991

[FRS88] Fortnow, L., Rompel, J., Sipser, M.: On the power of multi-prover interactive protocols. In: Theoretical Computer Science, pp. 156–161 (1988)

[FS89] Feige, U., Shamir, A.: Zero knowledge proofs of knowledge in two rounds. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 526–544. Springer, New York (1990). doi:10.1007/0-387-34805-0_46

[GGN10] Goldreich, O., Goldwasser, S., Nussboim, A.: On the implementation of huge random objects. SIAM J. Comput. **39**(7), 2761–2822 (2010). Preliminary version appeared in FOCS 2003

[GIMS10] Goyal, V., Ishai, Y., Mahmoody, M., Sahai, A.: Interactive locking, zero-knowledge PCPs, and unconditional cryptography. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 173–190. Springer, Heidelberg (2010). doi:10.1007/978-3-642-14623-7_10

[GKR08] Goldwasser, S., Kalai, Y.T., Rothblum, G.N.: Delegating computation: interactive proofs for Muggles. In: Proceedings of the 40th Annual ACM Symposium on Theory of Computing, STOC 2008, pp. 113–122 (2008)

[GMR89] Goldwasser, S., Micali, S., Rackoff, C.: The knowledge complexity of interactive proof systems. SIAM J. Comput. **18**(1), 186–208 (1989). Preliminary version appeared in STOC 1985

[GMW91] Goldreich, O., Micali, S., Wigderson, A.: Proofs that yield nothing but their validity or all languages in NP have zero-knowledge proof systems. J. ACM **38**(3), 691–729 (1991). Preliminary version appeared in FOCS 1986

[GV99] Goldreich, O., Vadhan, S.P.: Comparing entropies in statistical zero knowledge with applications to the structure of SZK. In: Proceedings of the 14th Annual IEEE Conference on Computational Complexity, CCC 1999, p. 54 (1999)

[IMS12] Ishai, Y., Mahmoody, M., Sahai, A.: On efficient zero-knowledge PCPs. In: Cramer, R. (ed.) TCC 2012. LNCS, vol. 7194, pp. 151–168. Springer, Heidelberg (2012). doi:10.1007/978-3-642-28914-9_9

[IMSX15] Ishai, Y., Mahmoody, M., Sahai, A., Xiao, D.: On zero-knowledge PCPs: limitations, simplifications, and applications (2015). http://www.cs.virginia.edu/~mohammad/files/papers/ZKPCPs-Full.pdf

[IOS97] Itoh, T., Ohta, Y., Shizuya, H.: A language-dependent cryptographic primitive. J. Cryptol. **10**(1), 37–50 (1997)

[IW14] Ishai, Y., Weiss, M.: Probabilistically checkable proofs of proximity with zero-knowledge. In: Lindell, Y. (ed.) TCC 2014. LNCS, vol. 8349, pp. 121–145. Springer, Heidelberg (2014). doi:10.1007/978-3-642-54242-8_6

[IWY16] Ishai, Y., Weiss, M., Yang, G.: Making the best of a leaky situation: zero-knowledge PCPs from leakage-resilient circuits. In: Kushilevitz, E., Malkin, T. (eds.) TCC 2016. LNCS, vol. 9563, pp. 3–32. Springer, Heidelberg (2016). doi:10.1007/978-3-662-49099-0_1

[IY87] Impagliazzo, R., Yung, M.: Direct minimum-knowledge computations (extended abstract). In: Pomerance, C. (ed.) CRYPTO 1987. LNCS, vol. 293, pp. 40–51. Springer, Heidelberg (1988). doi:10.1007/3-540-48184-2_4

[Kay10] Kayal, N.: Algorithms for arithmetic circuits (2010). ECCC TR10-073

[KI04] Kabanets, V., Impagliazzo, R.: Derandomizing polynomial identity tests means proving circuit lower bounds. Comput. Complexity **13**(1–2), 1–46 (2004)

[KPT97] Kilian, J., Petrank, E., Tardos, G.: Probabilistically checkable proofs with zero knowledge. In: Proceedings of the 29th Annual ACM Symposium on Theory of Computing, STOC 1997, pp. 496–505 (1997)

[KR08] Kalai, Y.T., Raz, R.: Interactive PCP. In: Aceto, L., Damgård, I., Goldberg, L.A., Halldórsson, M.M., Ingólfsdóttir, A., Walukiewicz, I. (eds.) ICALP 2008. LNCS, vol. 5126, pp. 536–547. Springer, Heidelberg (2008). doi:10.1007/978-3-540-70583-3_44

[LFKN92] Lund, C., Fortnow, L., Karloff, H.J., Nisan, N.: Algebraic methods for interactive proof systems. J. ACM **39**(4), 859–868 (1992)

[LS95] Lapidot, D., Shamir, A.: A one-round, two-prover, zero-knowledge protocol for NP. Combinatorica **15**(2), 204–214 (1995)

[Oka00] Okamoto, T.: On relationships between statistical zero-knowledge proofs. J. Comput. Syst. Sci. **60**(1), 47–108 (2000)

[Ost91] Ostrovsky, R.: One-way functions, hard on average problems, and statistical zero-knowledge proofs. In: Proceedings of the 6th Annual Structure in Complexity Theory Conference, CoCo 1991, pp. 133–138 (1991)

[OV08] Ong, S.J., Vadhan, S.: An equivalence between zero knowledge and commitments. In: Canetti, R. (ed.) TCC 2008. LNCS, vol. 4948, pp. 482–500. Springer, Heidelberg (2008). doi:10.1007/978-3-540-78524-8_27

[OW93] Ostrovsky, R., Wigderson, A.: One-way functions are essential for non-trivial zero-knowledge. In: Proceedings of the 2nd Israel Symposium on Theory of Computing Systems, ISTCS 1993, pp. 3–17 (1993)

[RRR16] Reingold, O., Rothblum, R., Rothblum, G.: Constant-round interactive proofs for delegating computation. In: Proceedings of the 48th ACM Symposium on the Theory of Computing, STOC 2016, pp. 49–62 (2016)

[RS05] Raz, R., Shpilka, A.: Deterministic polynomial identity testing in non-commutative models. Comput. Complexity **14**(1), 1–19 (2005). Preliminary version appeared in CCC 2004

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

[Sha92] Shamir, A.: IP = PSPACE. J. ACM **39**(4), 869–877 (1992)

[SV03] Sahai, A., Vadhan, S.P.: A complete problem for statistical zero knowledge. J. ACM **50**(2), 196–249 (2003)

[SY10] Shpilka, A., Yehudayoff, A.: Arithmetic circuits: a survey of recent results and open questions. Found. Trends Theoret. Comput. Sci. **5**(3–4), 207–388 (2010)

[Vad99] Vadhan, S.P.: A Study of statistical zero-knowledge proofs. Ph.D. thesis, MIT, August 1999

[VV15] Vaikuntanathan, V., Vasudevan, P.N.: Secret sharing and statistical zero knowledge. In: Iwata, T., Cheon, J.H. (eds.) ASIACRYPT 2015. LNCS, vol. 9452, pp. 656–680. Springer, Heidelberg (2015). doi:10.1007/978-3-662-48797-6_27

[Zip79] Zippel, R.: Probabilistic algorithms for sparse polynomials. In: Ng, E.W. (ed.) Symbolic and Algebraic Computation. LNCS, vol. 72, pp. 216–226. Springer, Heidelberg (1979). doi:10.1007/3-540-09519-5_73