Linear-Size Constant-Query IOPs for Delegating Computation

Eli Ben-Sasson
eli@starkware.co
StarkWare

Alessandro Chiesa
alexch@berkeley.edu
UC Berkeley

Lior Goldberg
lior@starkware.co
StarkWare

Tom Gur
tom.gur@warwick.ac.uk
University of Warwick

Michael Riabzev
michael@starkware.co
StarkWare

Nicholas Spooner
nick.spooner@berkeley.edu
UC Berkeley

Abstract

We study the problem of delegating computations via interactive proofs that can be probabilistically checked. Known as *interactive oracle proofs* (IOPs), these proofs extend probabilistically checkable proofs (PCPs) to multi-round protocols, and have received much attention due to their application to constructing cryptographic proofs (such as succinct non-interactive arguments).

We prove that a rich class of NEXP-complete problems, which includes machine computations over large fields and succinctly-described arithmetic circuits, has constant-query IOPs with $O(T)$-size proofs and $\mathrm{polylog}(T)$-time verification for T-size computations. This is the first construction that simultaneously achieves linear-size proofs and fast verification, regardless of query complexity.

An important metric when using IOPs to delegate computations is the cost of producing the proof. The highly-optimized proof length in our construction enables a very efficient prover, with arithmetic complexity $O(T \log T)$. Hence this construction is also the first to simultaneously achieve prover complexity $O(T \log T)$ and verifier complexity $\mathrm{polylog}(T)$.

Keywords: interactive oracle proofs; probabilistically checkable proofs; delegation of computation

Contents

1 Introduction

Checking computations faster than they can be run is a central goal in the theory of computation. The study of proof protocols that enable fast verification has produced powerful tools for complexity theory and cryptography, and has even led to applications to real-world problems such as delegation of computation. For applications, it is crucial that the underlying complexity-theoretic objects are efficient.

An influential line of work began with *probabilistically checkable proofs* (PCPs) [BFLS91]. These are non-interactive proofs for membership in a language, which admit fast probabilistic verification based on local queries to the proof. While the most prominent application of PCPs is to hardness of approximation [FGLSS96], seminal works of Kilian [Kil92] and Micali [Mic00] showed that PCPs can also be used to obtain computationally-sound schemes for delegation of computation that are asymptotically efficient.

The application of PCPs to delegation of computation singles out particular design objectives, distinct from those that arise from hardness of approximation. The relevant complexity measures for PCPs in the context of delegation are: *query complexity*, *verifier time*, and *prover time*. The latter two are self-explanatory, since the proof must be produced and validated; the former arises because, in existing delegation schemes based on PCPs, communication complexity depends linearly on the query complexity of the underlying PCP. Note that the running time of the prover is not typically considered in the context of PCPs, because one considers only the existence of a valid PCP string and not how it is constructed. For delegation schemes, on the other hand, the time required to generate the proof is often a barrier to practical use.

An ideal PCP for delegation would have *constant* query complexity, *(poly)logarithmic* verifier time, and *linear* prover time. This naturally implies that the *proof length* must also be linear, since it is a lower bound on the prover's running time in most applications. State-of-the-art PCPs achieve constant query complexity and polylogarithmic verifier time, but only *quasilinear* ($O(N \log^c N)$) proof length [Mie09]. While the proof length is asymptotically close to optimal, c is a fairly large constant, and moreover the construction uses gap amplification techniques that are not believed to be concretely efficient. The *only* construction of PCPs with linear proof length has query complexity $O(N^\epsilon)$ and a verifier that runs in non-uniform polynomial time [BKKMS13]; the running time of the prover is not specified. Clearly, these parameters are a long way from those desired of PCPs for fast verification of computation.

In light of these apparent barriers, Ben-Sasson et al. [BCS16] have demonstrated how to obtain computationally-sound delegation schemes from *interactive oracle proofs* (IOPs). This is a natural generalization of PCPs independently introduced by [BCS16; RRR16] (also generalizing the "interactive PCP" model studied in [KR08]). An IOP is an interactive protocol consisting of multiple rounds, where in each round the verifier sends a challenge and the prover responds with a PCP oracle to which the verifier can make a small number of queries. The proof length of an IOP is the total length of all oracles sent by the prover, and the query complexity is the total number of queries made to these oracles. The study of IOPs explores the tradeoff between a new efficiency measure, round complexity, and other efficiency measures. Viewed in this way, a PCP is an IOP with optimal round complexity.

The work of [BCS16] justifies why exploiting the tradeoff between round complexity and other efficiency measures is potentially advantageous for constructing computationally-sound delegation schemes. In particular, if we could obtain IOPs with constant round complexity that otherwise match the parameters of an ideal PCP (constant query complexity, polylogarithmic verifier time, linear prover time), then we would obtain delegation schemes that have the same asymptotic efficiency as those derived from an ideal PCP. Thus, for the purposes of verifiable delegation schemes, it suffices to construct such "ideal" IOPs.

A recent line of works has leveraged this tradeoff to establish a number of results that we do not know how to achieve via PCPs alone [BCGV16; BCGRS17; Ben+17; BBHR18b; BBHR18a; BCRSVW19; RR19].

Two of these constructions are particularly relevant for us: [BCGRS17] achieves IOPs for Boolean circuit satisfiability (CSAT) with linear proof length, unspecified polynomial prover time, and constant query and round complexity,[1] and [BCRSVW19] achieves logarithmic-round IOPs for arithmetic circuit satisfiability with $\tilde{O}(N \log N)$ prover time,[2] linear proof length, and logarithmic query complexity.

However, these IOPs do *not* have sublinear verifier time (see Section 1.2). The state-of-the-art for IOPs with polylogarithmic verifier time is [BBHR18a], which achieves $O(N \log N)$ proof length, $\tilde{O}(N \log^2 N)$ prover time, and logarithmic query and round complexity. Our goal in this paper is to make progress towards constructing ideal IOPs by giving a construction that simultaneously achieves the state of the art in all of these metrics: linear proof length, $\tilde{O}(N \log N)$ prover time, constant query and round complexity, and polylogarithmic verifier time.

1.1 Our results

In this work we construct IOPs for algebraic computations over large fields that are "almost" ideal; namely, we achieve linear proof length, $O(N \log N)$ (*strictly quasilinear*) prover arithmetic complexity, constant query and round complexity, and *polylogarithmic* verifier time. Our new IOP protocols match the state-of-the-art proof length and prover complexity of [BCRSVW19], while at the same time achieving an *exponential improvement* in verifier time for a rich class of computations. We focus on arithmetic complexity as the natural notion of efficiency for IOPs for algebraic problems.[3]

The arithmetic complexity of our prover is tightly connected to the proof length. That is, the proof consists of a constant number of Reed–Solomon codewords of size $O(N)$, and the running time of the prover is dominated by the time required to produce these encodings. In particular, if there were a linear-time encoding procedure for the Reed–Solomon code, our prover would run in linear time, and thereby achieve optimal prover efficiency without any other changes to the scheme itself.

Small fields. All of our results are stated over large fields. Computations over small fields (e.g. $\mathbb{F}_2$) can be handled by moving to an extension field, which introduces an additional logarithmic factor in the proof length and prover time (the same is true of [BBHR18a; BCRSVW19]). Even with this additional logarithmic factor, our construction matches the state of the art for prover complexity (but not proof length) for succinct *boolean* circuit satisfiability, while improving verifier time to polylogarithmic.

1.1.1 Delegating bounded-space algebraic computation

Rank-one constraint satisfiability (R1CS) is a natural generalization of arithmetic circuits that is widely used across theoretical and applied constructions of proof systems (see [Bow+18]). An R1CS instance is specified by matrices A, B, C over a finite field $\mathbb{F}$, and is satisfied by a vector w if $Aw \circ Bw = Cw$, where $\circ$ is the element-wise (Hadamard) product. Arithmetic circuits reduce in linear time to R1CS instances.

Many problems of interest, however, involve R1CS instances where the matrices A, B, C have some structure. For example, many applications consider computations that involve checking many Merkle authentication paths — in this case a hash function is invoked many times, within the same path and across different paths. It would be valuable for the verifier to run in time that is related to a *succinct representation* of such instances, rather than to the (much larger) explicit representation that "ignores" the structure. In light

[1] Subsequent to this work, [RR19] showed how to achieve such IOPs for CSAT where the proof length is $(1+\epsilon)N$ for any $\epsilon > 0$. Their verifier runs in time $\tilde{O}(N)$.

[2] Here the $\tilde{O}$ notation hides $\mathrm{poly}(\log \log N)$ factors.

[3] In terms of bit operations, the prover runs in time $O(N' \log^2 N' \, \mathrm{poly}(\log \log N')) = \tilde{O}(N \log N)$, where $N' = N / \log |\mathbb{F}| = \Theta(N / \log N)$ is the size of the instance measured in field elements.

of this motivation, we introduce a notion of *succinctly-represented R1CS instances* that capture a rich class of bounded-space algebraic computations. (Later in the paper we refer to these as *algebraic automata*.)

Definition 1.1 (informal)**.** *A* **succinct R1CS** *instance is specified by matrices $A = [A_0|A_1], B = [B_0|B_1], C = [C_0|C_1] \in \mathbb{F}^{k \times 2k}$ over $\mathbb{F}$, and a time bound T, and is satisfied by a vector $z \in \mathbb{F}^{kT}$ if*

$$\begin{pmatrix} A_0 & A_1 & & & \\ & A_0 & A_1 & & \\ & & \ddots & \ddots & \\ & & & A_0 & A_1 \end{pmatrix} z \quad \circ \quad \begin{pmatrix} B_0 & B_1 & & & \\ & B_0 & B_1 & & \\ & & \ddots & \ddots & \\ & & & B_0 & B_1 \end{pmatrix} z \quad = \quad \begin{pmatrix} C_0 & C_1 & & & \\ & C_0 & C_1 & & \\ & & \ddots & \ddots & \\ & & & C_0 & C_1 \end{pmatrix} z$$

The relation Succinct-R1CS *consists of pairs $(\mathbb{x}, z)$ s.t. $\mathbb{x}$ is an instance of succinct R1CS satisfied by z.*

The size of an *instance* is $O(k^2 + \log T)$, but the size of the described computation is kT. Note that Succinct-R1CS is a PSPACE-complete relation, while the (regular) R1CS relation is merely NP-complete.

To obtain some intuition about the definition, consider the problem of repeated application of an arithmetic circuit $\mathcal{C} \colon \mathbb{F}^n \to \mathbb{F}^n$. Suppose that we want to check that there exists z such that $\mathcal{C}^T(z) = 0^n$, where $\mathcal{C}^T = \mathcal{C}(\mathcal{C}(\cdots \mathcal{C}(\cdot)))$ is the circuit that applies $\mathcal{C}$ iteratively T times. The circuit $\mathcal{C}^T$ has size $\Omega(|\mathcal{C}| \cdot T)$, and if the verifier were to "unroll" the circuit then it would pay this cost in time. However, the R1CS instance corresponding to $\mathcal{C}^T$ is of the above form, with $k = O(|\mathcal{C}|)$, where (roughly) the matrices A_0, B_0, C_0 represent the gates of $\mathcal{C}$ and A_1, B_1, C_1 represent the wires between adjacent copies of $\mathcal{C}$. (The condition that the output of $\mathcal{C}^T$ is zero is encoded separately as a "boundary constraint"; see Definition 8.1 for details.)

Our first result gives a constant-round IOP for satisfiability of succinct R1CS where the verifier runs in time $\mathrm{poly}(k, \log T)$, the prover has arithmetic complexity $O(kT \log kT)$, and the proof length is $O(kT \log |\mathbb{F}|)$ (linear in the computation transcript). In the theorem statement below we take $k = O(1)$ for simplicity.

Theorem 1 (informal)**.** *There is a universal constant $\epsilon_0 \in (0, 1)$ such that, for any computation time bound $T(n)$ and large smooth field family $\mathbb{F}(n)$, there is a 4-round IOP for succinct R1CS over $\mathbb{F}(n)$, with proof length $O(T(n) \log |\mathbb{F}(n)|)$, 4 queries, and soundness error ϵ_0. The prover uses $O(T(n) \log T(n))$ field operations and the verifier uses $\mathrm{poly}(n, \log T(n))$ field operations.*

As in prior work (e.g., [BS08]), "large smooth field" refers to a field of size $\Omega(N)$, whose additive or multiplicative group has a nice decomposition (see Definitions 10.2 and 10.3). For example, ensembles of large enough binary fields have this property, as well as prime fields with smooth multiplicative groups.

1.1.2 Delegating unbounded-space algebraic computation

While algebraic automata capture a useful class of computations, they are restricted to space-bounded computation (recall Succinct-R1CS $\in$ PSPACE). In particular, they do not capture general NEXP computations unless PSPACE = NEXP. Our second result shows that a NEXP-complete algebraic problem, namely a succinct version of the *arithmetic circuit satisfiability* problem over large fields, has efficient IOPs.

Succinct SAT. In more detail, first recall the NP-complete problem of *arithmetic* circuit satisfiability, denoted ASAT, in which the goal is to decide whether an arithmetic circuit $C \colon \mathbb{F}^k \to \mathbb{F}$ over a finite field $\mathbb{F}$ has a satisfying assignment or not. Then, to capture NEXP, we follow Papadimitriou and Yannakakis [PY86] and consider a *succinct* version of the arithmetic circuit satisfiability problem, denoted Succinct-ASAT.[4] In

[4] Informally, Papadimitriou and Yannakakis [PY86] showed that if a language $\mathcal{L}$ is NP-complete under a local reduction, then its succinct version Succinct-$\mathcal{L}$ is NEXP-complete. A reduction is *local* if each bit of the reduction's output can be computed from a poly-logarithmic number of bits of the input, in poly-logarithmic time (which is typically the case for NP-completeness reductions).

this variant of the problem, we consider the satisfiability of circuits of size N that are represented by circuit descriptors of size $O(\log N)$. These descriptors are themselves circuits that, given a gate index g, return the type of gate g (addition or multiplication), its left input gate, and its right input gate.

Definition 1.2 (informal)**.** *The relation* Succinct-ASAT *consists of pairs* $\big((\mathbb{F}, m, H, I, o, D), w\big)$*, where* $\mathbb{F}$ *is a finite field,* $m \in \mathbb{N}$*,* H *is a succinctly-represented subset of* $\mathbb{F}^m$ *representing the gates of the circuit,* I *is a subset of* H *representing the input gates,* $o \in H$ *is the output gate,* $D\colon H \to (\{+, \times\} \times H \times H) \cup \{\mathbb{F}\}$ *is a circuit descriptor of an arithmetic circuit* C*, and the witness* $w\colon I \to \mathbb{F}$ *is such that* $C(w) = 0$*.*

Our second result is an IOP for Succinct-ASAT that has linear proof length and constant query complexity. We stress that the verifier in this IOP runs in *(poly)logarithmic* time in the size N of the circuit.[5]

Theorem 2 (informal)**.** *There is a universal constant* $\epsilon_0 \in (0, 1)$ *such that there is a* 5*-round IOP for* Succinct-ASAT *over large smooth fields with proof length* $O(N)$*,* 5 *queries, and soundness error* ϵ_0*. The prover uses* $O(N \log N)$ *field operations and the verifier uses* $\mathrm{poly}(|D|, \log N)$ *field operations.*

As in prior work (e.g., [BS08]), "large smooth field" refers to a field of size $\Omega(N)$, whose additive or multiplicative group has a nice decomposition (see Definition 10.2). For example, ensembles of large enough binary fields have this property. A *round* consists of a verifier message followed by a prover (oracle) message.

The single logarithmic factor in the prover's arithmetic complexity solely comes from the use of a constant number of Fast Fourier transforms over domains of size $\Theta(N)$. This prover time matches the best prover times of protocols for *non-succinct* arithmetic circuit satisfiability (which is merely NP-complete).

We remark that standard query reduction techniques do *not* preserve linear proof length here. Nevertheless, they can be modified in a straightforward way to preserve it, at the cost of an additional round of interaction. In particular, the number of queries in Theorem 2 can be reduced to 2 at the cost of an additional round.

Algebraic machines. We prove Theorem 2 by designing an IOP for the *algebraic machine* relation. This is a natural algebraic analogue of the bounded accepting problem for nondeterminstic random-access machines, where the transition function is an arithmetic (rather than Boolean) circuit. There is a simple linear-size reduction from Succinct-ASAT to the satisfiability problem for algebraic machines: the machine holds the values of the wires in its memory, and checks that each gate is correctly evaluated.

Theorem 3 (informal)**.** *There is a universal constant* $\epsilon_0 \in (0, 1)$ *such that, for any computation time bound* $T(n)$ *and large smooth field* $\mathbb{F}(n)$*, there is a* 5*-round IOP for the satisfiability problem of* $T(n)$*-time algebraic machines over* $\mathbb{F}(n)$*, with proof length* $O(T(n) \log |\mathbb{F}(n)|)$*,* 5 *queries, and soundness error* ϵ_0*. The prover uses* $O(T(n) \log T(n))$ *field operations and the verifier uses* $\mathrm{poly}(n, \log T(n))$ *field operations.*

For simplicity, we have stated Theorem 3 for machines whose description is a constant number of field elements, or $\Theta(\log |\mathbb{F}|)$ bits. The proof length is *linear* in the size of the computation trace, which is $N := \Theta(T \log |\mathbb{F}|)$ bits. We stress that the number of queries is 5, *regardless* of the machine.

On the power of machines. In the linear-length regime, the choice of computational model supported by a proof protocol is important, because reductions between problems typically introduce logarithmic factors. For example, it is not known how to reduce a random-access machine, or even a Turing machine, to a circuit of linear size. Indeed, the linear-size sublinear-query PCP of [BKKMS13] only works for circuit but not machine computations. We thus view Theorem 3 as particularly appealing, because it achieves linear length for a powerful model of computation, algebraic machines, which facilitates linear-size reductions from many

[5] This is because the verifier runs in time polynomial in the size of the circuit descriptor D, which in turn is logarithmic in the size of the circuit C that it describes.

other problems. Notably, while Theorem 3 implies Theorem 2, *we do not know whether the converse holds*. We view the identification of a model which is both highly expressive and amenable to efficient probabilistic checking using IOPs as a contribution of this work.

	rounds	**circuit type**	**prover time**	**verifier time**	**proof length**	**queries**
[Mie09]	1	succinct boolean	$N \operatorname{polylog}(N)$ *	$\operatorname{polylog}(N)$	$N \operatorname{polylog}(N)$	$O(1)$
[BKKMS13]	1	boolean	$\operatorname{poly}(N)$ †	$\operatorname{poly}(N)$ †	$O_\epsilon(N)$	N^ϵ
[BCGRS17]	3	boolean	$\operatorname{poly}(N)$	$\operatorname{poly}(N)$	$O(N)$	$O(1)$
[BBHR18a]	$O(\log N)$	succinct arithmetic ◊	$\widetilde{O}(N \log^2 N)$ ‡	$\operatorname{polylog}(N)$	$O(N \log N)$	$O(\log N)$
[BCRSVW19]	$O(\log N)$	arithmetic ◊	$\widetilde{O}(N \log N)$ ‡	$\operatorname{poly}(N)$	$O(N)$	$O(\log N)$
this work	5	succinct arithmetic ◊	$\widetilde{O}(N \log N)$ ‡	$\operatorname{polylog}(N)$	$O(N)$	5

Table 1: Comparison of PCP/IOP constructions for circuit satisfiability problems for a (fixed) constant soundness. Here N is the size of the circuit *in bits*, which means that, for arithmetic circuits, N implicitly includes a factor of $\log |\mathbb{F}|$. For succinct problems, the circuit size N is exponential in the size of its description.
*: [Mie09] shows a $\operatorname{poly}(N)$ bound; this tighter bound is due to [BCGT13b].
◊: The size of the underlying field must grow as $\Omega(N)$ to achieve the stated efficiency. Problems over smaller fields (e.g. boolean circuits) incur a multiplicative cost of $\log N$ in prover time and proof length.
†: The specified time is for *non-uniform* computation (each input size receives $\operatorname{poly}(N)$ advice bits).
‡: The notation $\widetilde{O}$ hides $\operatorname{poly}(\log \log N)$ factors, which arise because here we consider the *bit complexity* of the prover (rather than the arithmetic complexity).

1.2 Limitations of prior work

There are relatively few works that explicitly deal with prover complexity for PCP and IOP constructions. We present a comparison of the relevant parameters for each construction in Table 1. Since we are concerned with logarithmic factors, it is not sufficient to specify only a complexity class (NP or NEXP) for each one. Instead, for each proof system we give a canonical expressive language for which the given parameters are achieved. In particular, the first three proof systems are for boolean circuit problems, and the latter three are for arithmetic circuit problems. For purposes of comparison, all of the parameters for both boolean and arithmetic constructions are presented in terms of bit complexity.

Our construction also achieves asymptotically optimal proof length and query complexity, which are more well-studied metrics. There are two natural approaches that one could take to achieve such a result: (1) start from a construction with constant query complexity and reduce proof length; or (2) start from a construction with linear proof length and reduce query complexity. We summarize prior works that have followed these approaches, and highlight the limitations that arise in each case.

Approach (1). The first approach has been studied extensively [BFLS91; PS94; HS00; BSVW03; GS06; BGHSV06], leading to PCPs for NEXP with proof length $N \operatorname{polylog}(N)$ and query complexity $O(1)$ [BS08; BGHSV05; Din07; Mie09]. Later works have reduced the logarithmic factors in the proof length [BCGT13b; BCGT13a], but attempts to achieve linear length have failed. Recent work has obtained IOPs with proof length $O(N \log N)$ but at the cost of increasing query complexity from $O(1)$ to $O(\log N)$ [Ben+17; BBHR18a].

Approach (2). The second approach has received much less attention. Insisting on linear proof length significantly restricts the available techniques because many tools introduce logarithmic factors in proof

length. For example, one cannot rely on arithmetization via multivariate polynomials and standard low-degree tests, nor rely on algebraic embeddings via de Bruijn graphs for routing; in addition, query-reduction techniques for interactive PCPs [KR08] do not apply to the linear proof length regime. The state-of-the-art in linear-length PCPs is due to [BKKMS13], and the construction is based on a non-uniform family of algebraic geometry (AG) codes (every input size needs a polynomial-size advice string). In more detail, [BKKMS13] proves that for every $\epsilon \in (0, 1)$ there is a (non-uniform) PCP for the NP-complete problem CSAT (Boolean circuit satisfiability) with proof length $2^{O(1/\epsilon)}N$ and query complexity N^{ϵ}, much more than our goal of $O(1)$.

By leveraging interaction, [BCGRS17] obtains IOPs for CSAT with proof length $O(N)$ and query complexity $O(1)$. This is a natural starting point for our goal of achieving polylogarithmic-time verification, because we are "only" left to extend this result from CSAT to its succinct analogue, Succinct-CSAT. Unfortunately, the construction in [BCGRS17] uses AG codes and such an extension would, in particular, require obtaining a succinct representation of a dense asymptotically good family of AG codes over a small field, which is out of reach of current techniques. More generally, we do not know of any suitable code over small fields, which currently seems to prevent us from obtaining linear-size IOPs for Succinct-CSAT.

We now consider arithmetic circuit satisfiability defined over fields $\mathbb{F}$ that are large (of size $\Omega(N)$). In this regime, [BCRSVW19] obtains IOPs for ASAT with proof length $O(N)$ and query complexity $O(\log N)$. The arithmetization, following [BS08], is based on the Reed–Solomon code and uses the algebraic structure of large smooth fields. Testing is done via FRI [BBHR18b], a recent IOP of proximity for the Reed–Solomon code with linear proof length and logarithmic query complexity. The construction in [BCRSVW19], which we will build upon, falls short of our goal on two fronts: verifier time is linear in the size of the circuit rather than polylogarithmic, and query complexity is logarithmic rather than constant.

Comparison with [Ben+17; BBHR18a]. The techniques that we use in this work to achieve linear proof length could be applied to the protocols of [Ben+17; BBHR18a], which would yield linear-size proofs there. In this modified protocol, however, in order to verify time-T computations the verifier must read $O(\log T)$ field elements from the proof (see e.g. [BBHR18a, Lemma B.9]); in our protocol, the number of field elements the verifier reads is a universal constant. This is because the query complexity of these protocols depends linearly on the size of the description of the transition function of the machine (e.g., for Succinct-ASAT, this would be the size of the circuit descriptor).

1.3 Open questions

We highlight four problems left open by our work.

Optimal arithmetic complexity. The prover in our construction has strictly quasilinear arithmetic complexity and produces a proof of linear size. A natural question is whether the arithmetic complexity of the prover can be reduced to linear. To do so with our construction would require a breakthrough in encoding algorithms for the Reed–Solomon code. A promising direction may be to build IOPs based on codes with linear-time encoding procedures [Spi96; GI05; BCGGHJ17].

All fields. The question of whether it is possible to simultaneously achieve linear-length proofs and polylogarithmic-time verifier for Succinct-ASAT over *any* field $\mathbb{F}$ remains open. Progress on this question motivates the search for arithmetization-friendly families of good codes beyond the Reed–Solomon code. For example, the case of $\mathbb{F} = \mathbb{F}_2$, which corresponds to boolean circuits, motivates the search for succinctly-represented families of good algebraic-geometry codes over constant-size fields with fast encoding algorithms.

Zero knowledge. Zero knowledge, while not a goal of this work, is a desirable property, because zero knowledge PCP/IOPs lead to zero knowledge succinct arguments [IMSX15; BCS16]. Straightforward modifications to the protocol, similar to [BCRSVW19], achieve a notion of zero knowledge wherein the

simulator runs in time polynomial in the size of the computation being checked, which is meaningful for nondeterministic problems since it does not have access to the witness.

There is a stronger notion of zero knowledge for succinct languages where the simulator runs in *polylogarithmic* time (in time polynomial in the size of the instance). This gap was precisely the subject of a work on designing succinct simulators for certain tests [BCFGRS17]. Whether low-degree tests with the parameters we require have succinct simulators remains an intriguing problem that we leave to future research.

Round complexity. Our protocol has 5 rounds. Round complexity can be reduced to 4 at the cost of increased (constant) query complexity. Reducing round complexity beyond this while preserving linear proof length and polylogarithmic time verification, or finding evidence against this possibility, remains open.

2 Technical overview

We discuss the main ideas behind our results. Recall that Succinct-ASAT is the satisfiability problem for succinctly-represented arithmetic circuits over the field $\mathbb{F}$; this problem is NEXP-complete for any field $\mathbb{F}$ [PY86]. Our goal is to construct an IOP for Succinct-ASAT, over a large field $\mathbb{F}$, with proof length that is linear in the size of the circuit N and strictly quasilinear ($O(N \log N)$) prover arithmetic complexity; crucially, the running time of the verifier is *polylogarithmic* in the size of the circuit (more precisely, polynomial in the size of the circuit descriptor). Additionally, we strive to optimize the query and round complexity of this IOP. We stress that no prior work achieves non-trivial linear-length PCPs or IOPs wherein the verifier runs in polylogarithmic time in the size of the circuit.

The rest of this section is organized as follows. In Section 2.1 we discuss our starting point, which is a construction of [BCRSVW19]. In Section 2.2 we discuss our approach to overcoming the limitations of prior work by describing a new protocol for checking succinctly-represented linear relations; this achieves an *exponential* improvement over the prior state of the art. In Section 2.4 we discuss how to overcome the challenges that arise when attempting to build on this exponential improvement to checking the computation of algebraic automata, and then of algebraic machines (which capture Succinct-ASAT as a special case). In Section 2.5 we describe a modular framework, which we call *oracle reductions*, in which we prove our results.

2.1 Our starting point

The starting point of our work is [BCRSVW19], which obtains IOPs for ASAT (arithmetic circuit satisfiability) with proof length $O(N)$ and query complexity $O(\log N)$, and in which the prover uses $O(N \log N)$ field operations and the verifier uses $O(N)$. More precisely, this prior work obtains IOPs for an NP-complete problem that has a linear-time reduction from ASAT. The problem is denoted R1CS and captures the satisfiability of *rank-1 constraint systems*: given matrices A, B, C over a finite field $\mathbb{F}$, the problem asks whether there exists a witness vector w, where some entries are fixed to known values, for which the following *R1CS equation* holds: $(Aw) \circ (Bw) = Cw$, where "$\circ$" denotes entry-wise product.

Our goal in this paper is to achieve an IOP for Succinct-ASAT, the succinct analogue of ASAT. This entails an *exponential* improvement in the running time of the verifier, from linear in the circuit size to polylogarithmic in the circuit size. Moreover, we need to achieve this improvement with proof length $O(N)$ and query complexity $O(1)$ (and a prover that uses $O(N \log N)$ field operations). At a high level, our strategy towards this goal is to prove that there exists a class of succinctly-representable R1CS instances such that: (i) there exists a linear time reduction from Succinct-ASAT to R1CS that produces an instance in this class; and (ii) design an IOP protocol with the aforementioned parameters for R1CS instances in this class.

The ideas behind our results are better understood if we first briefly recall the IOP of [BCRSVW19]. The prover sends to the verifier four oracles $\pi_w, \pi_A, \pi_B, \pi_C$ that are purported encodings of w, Aw, Bw, Cw. The verifier must now check two sub-problems: (a) if π_w encodes w then π_A, π_B, π_C respectively encode Aw, Bw, Cw; and (b) if π_A, π_B, π_C encode w_A, w_B, w_C then $w_A \circ w_B = w_C$.

As usual, there is a tension in selecting the encoding used to obtain the oracles $\pi_w, \pi_A, \pi_B, \pi_C$. One needs an encoding that allows for non-trivial checking protocols, e.g., where the verifier makes a small number of queries. On the other hand, the encoding must have constant rate so that proof length can be linear.

The encoding used relies on univariate polynomials: denote by $\mathrm{RS}[L, \rho] \subseteq \mathbb{F}^L$ the Reed-Solomon code over a subset L of a field $\mathbb{F}$ with rate parameter $\rho \in (0, 1]$ (that is, the set of all functions $f\colon L \to \mathbb{F}$ of degree less than $\rho|L|$). Also, denote by $\hat{f}$ the unique univariate polynomial of degree less than $\rho|L|$ whose evaluation on L equals f. Given a subset $H \subseteq \mathbb{F}$ (the *domain* of the encoding), a Reed–Solomon codeword f *encodes*

$x \in \mathbb{F}^H$ if $\hat{f}(a) = x_a$ for all $a \in H$; for each x, there is a unique encoding f_x of x of minimal rate. We can now restate the aforementioned sub-problems in terms of the Reed–Solomon code.

- **Lincheck:** given a subset $H \subseteq \mathbb{F}$, Reed–Solomon codewords $f, g \in \mathrm{RS}[L, \rho]$ that encode $x, y \in \mathbb{F}^H$ respectively, and a matrix $M \in \mathbb{F}^{H \times H}$, check that $Mx = y$.
- **Rowcheck:** given a subset $H \subseteq \mathbb{F}$ and Reed–Solomon codewords $f, g, h \in \mathrm{RS}[L, \rho]$ that encode $x, y, z \in \mathbb{F}^H$ respectively, check that $x \circ y = z$.

The IOP in [BCRSVW19] is obtained by combining sub-protocols for these sub-problems, a *lincheck protocol* and a *rowcheck protocol*. The latter is a simple reduction from the rowcheck problem to testing membership in the Reed–Solomon code, and is implied by standard PCP tools. The former, however, is a novel (and non-trivial) reduction from the lincheck problem to testing membership in the Reed–Solomon code.

While on the one hand the verifier in the rowcheck protocol runs in time that is polylogarithmic in $|H|$ (which is good) the verifier in the lincheck protocol runs in time that is linear in $|H|$ (which is much too slow). In other words, if we simply invoked the IOP in [BCRSVW19] on the circuit described by an instance of Succinct-ASAT, the verifier would run in time that is linear in the circuit size, which is exponentially worse than our goal of polylogarithmic time. This state of affairs in the starting point of our work.

Next, in Section 2.2, we discuss how to obtain a succinct lincheck protocol that, for suitable linear relations, is exponentially more efficient. After that, in Section 2.4, we discuss how to build on our succinct lincheck protocol to reduce Succinct-ASAT to testing membership in the Reed–Solomon code, while achieving verifier time that is *polylogarithmic* in circuit size.

Throughout, we present our contributions as *oracle reductions* from some computational task to testing membership in the Reed–Solomon code. Loosely speaking, these are are reductions in the setting of the IOP model (and therefore, in particular, allow interaction in which the prover sends PCP oracles). This abstraction allows us to decouple IOP protocol-design from the low-degree test that we invoke at the end of the protocol. See Section 2.5 for details.

2.2 Checking succinctly-represented linear relations

Following the above discussion, we now temporarily restrict our attention to devising a lincheck protocol, which reduces checking linear relations defined by matrices $M \in \mathbb{F}^{H \times H}$ to testing membership in the Reed–Solomon code, in which the verifier runs in time that is *polylogarithmic* in $|H|$. This is not possible in general, however, because the verifier needs to at least *read the description* of the matrix M. We shall have to consider matrices M that have a special structure that can be exploited to obtain an exponential improvement in verifier time. This improvement is a core technical contribution of this paper, and we refer to the resulting reduction as the *succinct lincheck protocol*. We start by describing the ideas behind the (non-succinct) lincheck protocol of [BCRSVW19].

Definition 2.1 (informal)**.** *In the lincheck problem, we are given a subset $H \subseteq \mathbb{F}$, Reed–Solomon codewords $f, g \in \mathrm{RS}[L, \rho]$ encoding vectors $x, y \in \mathbb{F}^H$, and a matrix $M \in \mathbb{F}^{H \times H}$. The goal is to check that $x = My$.*

A simple probabilistic test for the claim "$x = My$" is to check that $\langle r, x - My \rangle = 0$ for a random $r \in \mathbb{F}^H$. Indeed, if $x \neq My$, then $\Pr_{r \in \mathbb{F}^H}[\langle r, x - My \rangle = 0] = 1/|\mathbb{F}|$. However, this approach would require the verifier to sample, and send to the prover, $|H|$ random field elements (too many).

A natural derandomization is to choose $\vec{r}$ using a small-bias generator over $\mathbb{F}$, rather than uniformly at random. A small-bias generator G over $\mathbb{F}$ is a function with the property that for any nonzero $z \in \mathbb{F}^H$, it holds with high probability over $\rho \in \{0,1\}^\ell$ that $\langle z, G(\rho) \rangle \neq 0$. Now the verifier needs to send only ℓ bits to the prover, which can be much smaller than $|H| \log |\mathbb{F}|$.

A natural choice (used also, e.g., in [BFLS91, §5.2]) is the powering construction of [AGHP92], which requires sending a *single* random field element ($\ell = \log |\mathbb{F}|$), and incurs only a modest increase in soundness error. In this construction, we define a vector $\vec{r}(X) \in \mathbb{F}[X]^H$ of linearly independent polynomials in X, given by $(1, X, X^2, \ldots, X^{|H|-1})$. The small-bias generator is then $G(\alpha) := \vec{r}(\alpha)$ for $\alpha \in \mathbb{F}$. If z is nonzero then $h(X) := \langle \vec{r}(X), z \rangle$ is a nonzero polynomial and so $\Pr_{\alpha \in \mathbb{F}}[\langle G(\alpha), z \rangle = 0] \leq \deg(h)/|\mathbb{F}|$. The verifier now merely has to sample and send $\alpha \in \mathbb{F}$, and the prover must then prove the claim "$h(\alpha) = 0$" to the verifier. Rearranging, this is the same as testing that $\langle \vec{r}(\alpha), x \rangle - \langle \vec{r}(\alpha)M, y \rangle = 0$. The problem is thus reduced to checking inner products of known vectors with oracles.

In the setting of Reed–Solomon codewords, if f_u is an encoding of u and f_v is an encoding of v, then $f_u \cdot f_v$ is an encoding of $u \circ v$, the pointwise product of u and v. Hence, to check that $\langle u, v \rangle = c$, it suffices to check that the low-degree polynomial $f_u \cdot f_v$ sums to c on H, since $\langle u, v \rangle = \sum_{h \in H} f_u(h) f_v(h)$. This can be achieved by running the *univariate sumcheck protocol* ([BCRSVW19]) on the codeword $f_u \cdot f_v$. This protocol requires the verifier to efficiently determine the value of $f_u \cdot f_v$ at a given point in L.

The inefficiency. The foregoing discussion tells us that, to solve the lincheck problem, the verifier must determine the value of the Reed–Solomon encodings of $\vec{r}(\alpha) \circ x$ and $\vec{r}(\alpha)M \circ y$ at a given point in L. The encodings of the vectors x and y are provided (as f and g). Hence it suffices for the verifier to evaluate *low-degree extensions* of $\vec{r}(\alpha)$ and $\vec{r}(\alpha)M$ at a given point, and then perform a field multiplication.

This last step is the computational bottleneck of the protocol. In [BCRSVW19], the verifier evaluates the low-degree extensions of $\vec{r}(\alpha)$ and $M^\top \vec{r}(\alpha)$ via Lagrange interpolation, which requires time $\Omega(|H|)$. To make our verifier efficient, we must evaluate both low-degree extensions in time $\mathrm{polylog}(|H|)$. In particular, this requires that M be succinctly represented, since computing the low-degree extension of $\vec{r}(\alpha)M$ in general requires time linear in the number of nonzero entries in M, which is at least $|H|$.

The lincheck protocol in [BCRSVW19] chooses the linearly independent polynomials $\vec{r}(X)$ to be the *standard* (or *coefficient*) basis $(1, X, \ldots, X^{|H|-1})$. For this basis, however, we do not know how to efficiently evaluate the low-degree extension of $\vec{r}(\alpha)$. This problem must be addressed *regardless* of the matrix M.

A new basis and succinct matrices. We leverage certain algebraic properties to overcome the above problem. There is another natural choice of basis for polynomials, the *Lagrange basis* $(L_{H,h}(X))_{h \in H}$, where $L_{H,h}$ is the unique polynomial of degree less than $|H|$ with $L_{H,h}(h) = 1$ and $L_{H,\alpha}(\gamma) = 0$ for all $\gamma \in H \setminus \{h\}$. We observe that the low-degree extension of $\vec{r}(\alpha) = (L_{H,h}(\alpha))_{h \in H} \in \mathbb{F}^H$ has a simple form that allows one to evaluate it in time $\mathrm{polylog}(|H|)$ provided that H is an additive or multiplicative subgroup of $\mathbb{F}$. In other words, the Lagrange basis yields a small-bias generator over $\mathbb{F}$ whose low-degree extension can be computed efficiently.

It remains to find a useful class of succinctly-represented matrices M for which one can efficiently evaluate a low-degree extension of $\vec{r}(\alpha)M \in \mathbb{F}^H$. The foregoing discussion suggests a natural condition: *if* we can efficiently compute a low-degree extension of a vector $v \in \mathbb{F}^H$ *then* we should also be able to efficiently compute a low-degree extension of the vector vM. If this holds for all vectors v, we say that the matrix M is (algebraically) *succinct*. For example, the identity matrix satisfies this definition (trivially), and so does the matrix with 1s on the superdiagonal for appropriate choices of $\mathbb{F}$ and H (see Section 8.1).

In sum, if we choose the Lagrange basis in the lincheck protocol and the linear relation is specified by a succinct matrix, then, with some work, we obtain a lincheck protocol where the verifier runs in time $\mathrm{polylog}(|H|)$. To check satisfiability of succinctly-represented arithmetic circuits, however, we need to handle a more general class of matrices, described next.

Succinct lincheck for semisuccinct matrices. We will relax the condition on a matrix M in a way that captures the matrices that arise when checking succinctly-described arithmetic circuits, while still allowing us to obtain a lincheck protocol in which the verifier runs in time $\mathrm{polylog}(|H|)$.

We show that the matrices that we consider are *semisuccinct*, namely, they can be decomposed into a "large" part that is succinct and a "small" part that has no special structure.[6] This structure should appear familiar, because it is analogous to how a succinctly-described circuit consists of a small arbitrary component (the circuit descriptor) that is repeatedly used in a structured way to define the large circuit. Another analogy is how in an automaton or machine computation a small, and arbitrary, transition function is repeatedly applied across a large computation.

Specifically, by "decompose" we mean that the matrix $M \in \mathbb{F}^{H \times H}$ can be written as the *Kronecker product* of a succinct matrix $A \in \mathbb{F}^{H_1 \times H_1}$ and a small matrix $B \in \mathbb{F}^{H_2 \times H_2}$; we write $M = A \otimes B$. (Succinctly representing a large operator like M via the tensor product of simpler operators should be a natural idea to readers familiar with quantum information.) In order for the product to be well-defined, we must supply a bijection $\Phi \colon H \to H_1 \times H_2$. If this bijection satisfies certain algebraic properties, which preserve the succinctness of the matrix A, we call it a *bivariate embedding*.

We obtain a succinct lincheck protocol for semisuccinct matrices.

Lemma 2.2 (informal)**.** *There is a linear-length oracle reduction from the lincheck problem for semisuccinct matrices to testing membership in the Reed–Solomon code, where the verifier runs in polylogarithmic time.*

Next, we discuss how to obtain a reduction from algebraic automata (succinct R1CS) to testing membership in the Reed–Solomon code, where the verifier runs in time that is *polylogarithmic* in the circuit size, by building on our succinct lincheck protocol for semisuccinct matrices.

2.3 Checking bounded-space computations in polylogarithmic time

An instance of the *algebraic automaton* relation is specified by three $k \times 2k$ matrices (A, B, C) over $\mathbb{F}$, and a time bound T. A witness $f \colon [T] \to \mathbb{F}^k$ is valid if

$$\forall\, t \in [T-1] \quad Af(t, t+1) \circ Bf(t, t+1) = Cf(t, t+1) \ , \tag{1}$$

where $f(t, t+1) \coloneqq f(t) \| f(t+1)$ is the concatenation of the consecutive states $f(t) \in \mathbb{F}^k$ and $f(t+1) \in \mathbb{F}^k$.

We use the term "algebraic automata" since one can think of A, B, C as specifying the transition relation of a computational device with k algebraic registers, and f as an *execution trace* specifying an accepting computation of the device. The relation is PSPACE-complete: it is in NPSPACE because it can be checked by a polynomial-space Turing machine with one-directional access to an exponential-size witness, and recall that NPSPACE = PSPACE; also, it is PSPACE-hard because given an arithmetic circuit specifying the transition relation of a polynomial-space machine, we can find an equisatisfiable R1CS instance in linear time.

If we view the execution trace f as a vector $f = f(1) \| \cdots \| f(T) \in \mathbb{F}^{Tk}$, then we can rewrite the condition in Eq. (1) via the following (possibly exponentially large) R1CS equation:

$$\begin{pmatrix} A_0 & A_1 & & & \\ & A_0 & A_1 & & \\ & & \ddots & \ddots & \\ & & & A_0 & A_1 \end{pmatrix} f \quad \circ \quad \begin{pmatrix} B_0 & B_1 & & & \\ & B_0 & B_1 & & \\ & & \ddots & \ddots & \\ & & & B_0 & B_1 \end{pmatrix} f \quad = \quad \begin{pmatrix} C_0 & C_1 & & & \\ & C_0 & C_1 & & \\ & & \ddots & \ddots & \\ & & & C_0 & C_1 \end{pmatrix} f$$

where $A_0, A_1 \in \mathbb{F}^{k \times k}$ are the first half and second half of A respectively; and likewise for B and C. We thus see that algebraic automata are equivalent to Succinct-R1CS.

[6] We actually need to handle matrices that are the *sum* of semisuccinct matrices, but we ignore this in this high-level discussion.

The matrices in the above R1CS equation have a rigid block structure that we refer to as a *staircase*. Given the discussions in Sections 2.1 and 2.2, in order to achieve polylogarithmic verifier time, *it suffices to show that staircase matrices are semisuccinct* (or, at least, the sum of few semisuccinct matrices).

So let $S(M_0, M_1)$ be the staircase matrix of two given $k \times k$ matrices M_0, M_1 over $\mathbb{F}$. Namely, $S(M_0, M_1)$ is the $Tk \times Tk$ matrix with M_0-blocks on the diagonal and M_1-blocks on the superdiagonal. Observe that:

1. we can write the matrix with M_0-blocks on the diagonal as $I \otimes M_0$, where I is the $T \times T$ identity matrix;
2. we can write the matrix with M_1-blocks on the superdiagonal as $I^{\rightarrow} \otimes M_1$, where $I^{\rightarrow}$ is the $T \times T$ matrix with 1s on the superdiagonal.

Under an appropriate mapping from $[Tk]$ into a subset of $\mathbb{F}$, we prove that both of these matrices are semisuccinct. This tells us that $S(M_0, M_1)$ is the sum of two semisuccinct matrices:

$$S(M_0, M_1) := \begin{pmatrix} M_0 & M_1 & & & \\ & M_0 & M_1 & & \\ & & \ddots & \ddots & \\ & & & M_0 & M_1 \\ & & & & M_0 \end{pmatrix} = I \otimes M_0 + I^{\rightarrow} \otimes M_1 \in \mathbb{F}^{Tk} \times \mathbb{F}^{Tk} \ .$$

(Note that the above is not exactly the matrix structure we want, because of the extra M_0 block; we handle this technicality separately.) We obtain the following lemma.

Lemma 2.3 (informal)**.** *There is a linear-length oracle reduction from the algebraic automaton relation to testing membership in the Reed–Solomon code, where the verifier runs in time* $\mathrm{poly}(k, \log T)$.

2.4 Checking succinct satisfiability in polylogarithmic time

We outline our construction of a linear-length IOP for succinct circuit satisfiability (Succinct-ASAT) over a large smooth field $\mathbb{F}$. Informally, our strategy is as follows. First, we note that there is a simple linear-size reduction (see Section 10.2) from Succinct-ASAT to satisfiability of an algebraic automaton augmented with a permutation constraint; we refer to the latter as an *algebraic machine*. Then, we prove that checking such an instance can be reduced to checking an algebraic automaton *without* the permutation constraint, which can be achieved as described in the previous section. In the remainder of this section we discuss the main technical challenges that arise in this approach.

An instance of the *algebraic (R1CS) machine* relation is specified by two algebraic (R1CS) automata $(\mathcal{A}, \mathcal{A}')$. A witness (f, π), where $f \colon [T] \to \mathbb{F}^k$ is an execution trace and $\pi \colon [T] \to [T]$ is a permutation, is valid if: (1) f is a valid witness for the automaton $\mathcal{A}$, and (2) $f \circ \pi$ is a valid witness for the automaton $\mathcal{A}'$. The algebraic machine relation is NEXP-complete, as Succinct-ASAT reduces to it in linear time (see Section 10.2).

Execution traces for machines. Before we discuss how we reduce from the algebraic machine relation, we briefly explain why the above relation is a natural problem to consider, and in particular why it has anything to do with (random-access) machines. Recall that a random-access machine is specified by a list of instructions, each of which is an arithmetic operation, a control-flow operation, or a read/write to memory. One way to represent the execution trace for the machine is to record the state of the entire memory at each time step; for a time-T space-S computation, such an execution trace has size $\Theta(TS)$ (much more than linear!). Yet, the machine can access only a *single* memory location at each time step. Thus, instead of writing down the state of the entire memory at each time step, we could hope to only write the state of the accessed address — this would reduce the size of the trace to $\Theta(T \log S)$. The problem then is that it is no longer possible to check consistency of memory via local constraints because the same address can be accessed at any time.

Classical techniques of Gurevich and Shelah [GS89] tell us that one can efficiently represent an execution trace for a machine via *two* execution traces that are *permutations of one another*. Informally, sorting the execution trace by time enables us to check the transition relation of the machine; and sorting the execution trace by the accessed addressed (and then by time) enables us to locally check that memory is consistent. (One must ensure that, for each location, if we write some value to memory and then read the same address, we retrieve that same value.) The transition relation and memory consistency can each be expressed individually as automata. This view of machines immediately gives rise to the algebraic machine relation above.

Checking the algebraic machine relation. We have discussed how to check automata in Section 2.3, so it remains to check that the traces are permutations of one another. Historically this has been achieved in the PCP literature using algebraic embeddings of routing networks; e.g., see [BCGT13a]. The problem is that this increases the size of the representation of the execution trace by at least a logarithmic factor. We instead use an interactive permutation test from the program checking literature [Lip89; BK95]. The test is based on the observation that $u \in \mathbb{F}^T$ is a permutation of $v \in \mathbb{F}^T$ if and only if the multi-sets given by their elements are equal, which is true if and only if the polynomials $p_u(X) = \prod_{i=1}^{T}(X - u_i)$ and $p_v(X) = \prod_{i=1}^{T}(X - v_i)$ are equal. Thus it suffices to evaluate each polynomial at a random point and check equality.

These polynomials require time $\Theta(T)$ to evaluate, which in our setting is exponential. Therefore the prover must assist the verifier with the evaluation. We show that evaluating this polynomial can be expressed as an algebraic automaton, and can therefore be checked again using the protocol from Section 2.3.

The reader may believe that by now we have reduced checking an algebraic machine to checking three instances of algebraic automata. Recall, however, that the algebraic automaton relation is PSPACE-complete, whereas the algebraic machine relation is NEXP-complete. What happened? The answer lies in the randomness used in the permutation automaton. In order to check that u is a permutation of v, the prover must first commit to u and v before the verifier chooses his evaluation point α, and then the prover sends the trace of the automaton that evaluates $p_u(\alpha), p_v(\alpha)$. This trace depends on the choice of α, and so we use interaction. This is captured by the *interactive automaton relation* (Definition 8.1), which is NEXP-complete; it can be checked in essentially the same way as the automaton relation described in Section 2.3.

We hence obtain the following lemma.

Lemma 2.4 (informal)**.** *There is a linear-length oracle reduction from the algebraic machine relation (hence,* Succinct-ASAT*) to testing membership in the Reed–Solomon code, where the verifier runs in time* $\mathrm{poly}(k, \log T)$.

2.5 Oracle reductions

Many results in this paper describe IOPs that reduce a computational problem to membership in (a subcode of) the Reed–Solomon code. We find it useful to capture this class of reductions via a precise definition. This lets us prove general lemmas about such reductions, and obtain our protocols in a modular fashion.

We thus formulate a new notion that we call *interactive oracle reductions* (in short, *oracle reductions*). Informally, an oracle reduction is a protocol that reduces from a computational problem to testing membership in a code (in this paper, the code is the interleaved Reed–Solomon code). This is a well-understood idea in constructions of PCPs and IOPs. Our contribution is to provide a formal framework for this technique.

We illustrate the notion of oracle reductions via an example. Consider the problem of testing proximity to the *vanishing Reed–Solomon code*, which plays an important role in a PCP of Ben-Sasson and Sudan [BS08] and several other PCPs/IOPs. Informally, the goal is to test whether a univariate polynomial f, provided as an oracle, is zero everywhere on a subset H of $\mathbb{F}$.

We describe an oracle reduction that maps the foregoing problem to the problem of testing membership in the Reed–Solomon code of the related polynomial $g := f/\mathbb{Z}_H$. Observe that f is divisible by $\mathbb{Z}_H$ if and only if f is zero everywhere in H, and so g is in the Reed–Solomon code if and only if f satisfies the desired property. But what exactly *is* g? In the oracle reduction framework, we refer to g as a *virtual oracle*: an oracle whose value at any given point in its domain can be determined efficiently by making a small number of queries to concrete oracles. In this case, so long as the domain L we choose for g does not intersect H, a verifier can evaluate g at any point $\alpha \in L$ with only a single query to f. To test that g is low degree, the verifier can invoke any low-degree test on g, and simulate queries to the virtual oracle g via queries to f.

The two main parameters in an oracle reduction are the *proof length*, which is simply the total length of the oracles sent by the prover, and the *locality*, which is the number of queries one would have to make to the concrete oracles to answer a single query to any virtual oracle (in this paper, locality always equals the number of rounds). Using the perspective of oracle reductions, our main theorems (Theorems 1 and 2) follows by combining two main sub-components: (1) a linear-length 3-local oracle reduction from the Succinct-ASAT problem to proximity testing to the Reed–Solomon code (discussed in Section 2.4); and (2) a linear-length 3-query IOP for testing proximity to the Reed–Solomon code from [BCGRS17] (see Lemma 10.1).

3 Roadmap

Fig. 1 below provides a diagram of the results proved in this paper. The remaining sections in this paper are organized as follows. In Section 4 we recall useful notions and definitions. In Section 5 we define oracle reductions, and prove how to create IOP protocols from RS oracle reductions and RS proximity tests. In Section 6 we define and construct trace embeddings. In Section 7 we describe our succinct lincheck protocol. In Section 8 we describe an oracle reduction from R1CS automata to testing proximity to the Reed–Solomon code, proving Theorem 1. In Section 9 we describe an oracle reduction from R1CS machines to testing proximity to the Reed–Solomon code. In Section 10 we prove Theorem 2 and Theorem 3.

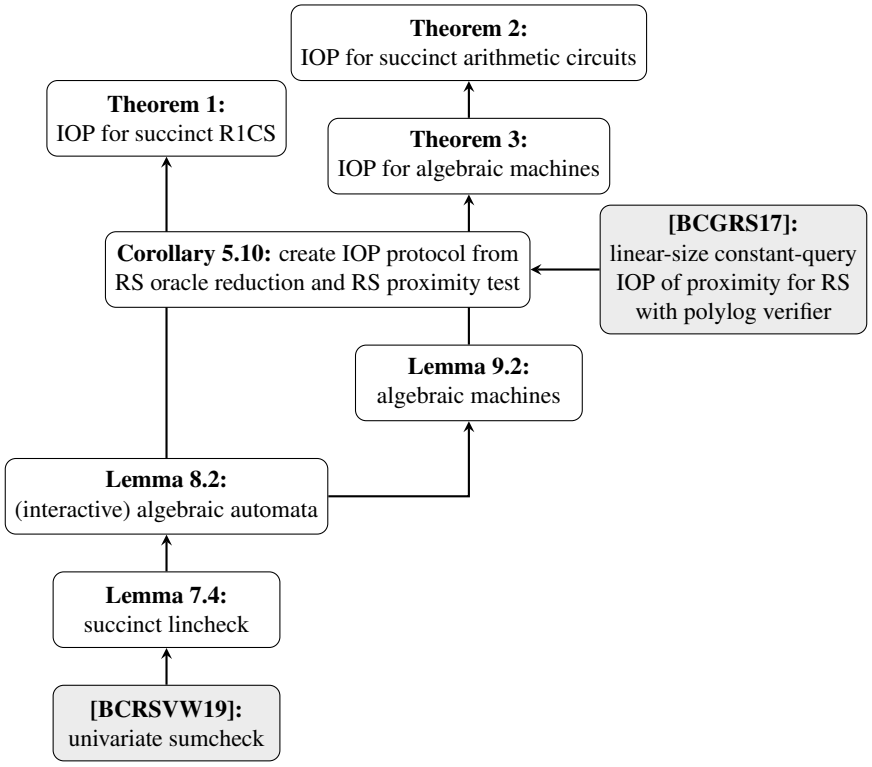

Figure 1: Diagram of the results in this paper.

4 Preliminaries

Given a relation $\mathcal{R} \subseteq S \times T$, we denote by $\mathcal{L}(\mathcal{R}) \subseteq S$ the set of $s \in S$ such that there exists $t \in T$ with $(s,t) \in \mathcal{R}$; for $s \in S$, we denote by $\mathcal{R}|_s \subseteq T$ the set $\{t \in T : (s,t) \in \mathcal{R}\}$. Given a set S and strings $v, w \in S^n$ for some $n \in \mathbb{N}$, the *fractional Hamming distance* $\Delta(v,w) \in [0,1]$ is $\Delta(v,w) \coloneqq \frac{1}{n}|\{i : v_i \neq w_i\}|$. We denote the concatenation of two vectors u_1, u_2 by $u_1 \| u_2$, and the concatenation of two matrices A, B by $[A|B]$. All fields $\mathbb{F}$ in this paper are finite, and we denote the finite field of size q by $\mathbb{F}_q$. We say that H is a *subgroup* in $\mathbb{F}$ if it is either a subgroup of $(\mathbb{F}, +)$ (an additive subgroup) or of $(\mathbb{F} \setminus \{0\}, \times)$ (a multiplicative subgroup); we say that H is a *coset* in $\mathbb{F}$ if it is a coset of a subgroup in $\mathbb{F}$ (possibly the subgroup itself).

4.1 Codes and polynomials

The Reed–Solomon code. Given a subset S of a field $\mathbb{F}$ and *rate* $\rho \in (0,1]$, we denote by $\mathrm{RS}[S,\rho] \subseteq \mathbb{F}^S$ all evaluations over S of univariate polynomials of degree less than $\rho|S|$. Namely, a word $c \in \mathbb{F}^S$ is in $\mathrm{RS}[S,\rho]$ if there is a polynomial of degree less than $\rho|S|$ that, for every $a \in S$, evaluates to c_a at a. We denote by $\mathrm{RS}[S,(\rho_1,\ldots,\rho_n)] \coloneqq \prod_{i=1}^n \mathrm{RS}[S,\rho_i]$ the interleaving of Reed–Solomon codes with rates $\rho_1,\ldots,\rho_n$.

Representations of polynomials. We frequently move from univariate polynomials over $\mathbb{F}$ to their evaluations on subsets of $\mathbb{F}$, and back. We use plain letters like f, g, h, π to denote *evaluations* of polynomials, and "hatted letters" $\hat{f}, \hat{g}, \hat{h}, \hat{\pi}$ to denote corresponding polynomials. This bijection is well-defined only if the size of the evaluation domain is larger than the degree. If $f \in \mathrm{RS}[S,\rho]$ for $S \subseteq \mathbb{F}$ and $\rho \in (0,1]$, then $\hat{f}$ is the unique polynomial of degree less than $\rho|S|$ whose evaluation on S equals f. Likewise, if $\hat{f} \in \mathbb{F}[X]$ with $\deg(\hat{f}) < \rho|S|$, then $f_S \coloneqq \hat{f}|_S \in \mathrm{RS}[S,\rho]$ (but we drop the subscript when the subset is clear from context).

Vanishing polynomials. Let $\mathbb{F}$ be a finite field, and $S \subseteq \mathbb{F}$. We denote by $\mathbb{Z}_S$ the unique non-zero monic polynomial of degree at most $|S|$ that is zero everywhere on S; $\mathbb{Z}_S$ is called the *vanishing polynomial* of S. In this work we use efficiency properties of vanishing polynomials for sets S that have group structure.

If S is a multiplicative subgroup of $\mathbb{F}$, then $\mathbb{Z}_S(X) = X^{|S|} - 1$, and so $\mathbb{Z}_S(X)$ can be evaluated at any $\alpha \in \mathbb{F}$ in $O(\log|S|)$ field operations. More generally, if S is a γ-coset of a multiplicative subgroup S_0 (namely, $S = \gamma S_0$) then $\mathbb{Z}_S(X) = \gamma^{|S|}\mathbb{Z}_{S_0}(X/\gamma) = X^{|S|} - \gamma^{|S|}$.

If S is an (affine) subspace of $\mathbb{F}$, then $\mathbb{Z}_S$ is called an *(affine) subspace polynomial*. In this case, there exist coefficients $c_0,\ldots,c_k \in \mathbb{F}$, where $k \coloneqq \dim(S)$, such that $\mathbb{Z}_S(X) = X^{p^k} + \sum_{i=1}^k c_i X^{p^{i-1}} + c_0$ (if S is linear then $c_0 = 0$). Hence, $\mathbb{Z}_S(X)$ can be evaluated at any $\alpha \in \mathbb{F}$ in $O(k \log p) = O(\log|S|)$ operations. Such polynomials are called *linearized* because they are $\mathbb{F}_p$-affine maps: if $S = S_0 + \gamma$ for a subspace $S_0 \subseteq \mathbb{F}$ and shift $\gamma \in \mathbb{F}$, then $\mathbb{Z}_S(X) = \mathbb{Z}_{S_0}(X-\gamma) = \mathbb{Z}_{S_0}(X) - \mathbb{Z}_{S_0}(\gamma)$, and $\mathbb{Z}_{S_0}$ is an $\mathbb{F}_p$-linear map. The coefficients $c_0,\ldots,c_k$ can be derived from a description of S (any basis of S_0 and the shift γ) in $O(k^2 \log p)$ field operations (see [LN97, Chapter 3.4] and [BCGT13a, Remark C.8]).

Lagrange polynomials. For $\mathbb{F}$ a finite field, $S \subseteq \mathbb{F}$, $a \in S$, we denote by $L_{S,a}$ the unique polynomial of degree less than $|S|$ such that $L_{S,a}(a) = 1$ and $L_{S,a}(b) = 0$ for all $b \in S \setminus \{a\}$. Note that

$$L_{S,a}(X) = \frac{\prod_{b \in S\setminus\{a\}}(X-b)}{\prod_{b \in S\setminus\{a\}}(a-b)} = \frac{L'_S(X)}{L'_S(a)} \ ,$$

where $L'_S(X)$ is the polynomial $\mathbb{Z}_S(X)/(X-a)$. For additive and multiplicative subgroups S and $a \in S$, we can evaluate $L_{S,a}(X)$ at any $\alpha \in \mathbb{F}$ in $\mathrm{polylog}(|S|)$ field operations. This is because an arithmetic circuit for L'_S can be efficiently derived from an arithmetic circuit for $\mathbb{Z}_S$ [SY10].

4.2 Interactive oracle proofs

The information-theoretic protocols in this paper are *Interactive Oracle Proofs* (IOPs) [BCS16; RRR16], which combine aspects of Interactive Proofs [Bab85; GMR89] and Probabilistically Checkable Proofs [BFLS91; AS98; ALMSS98], and also generalize the notion of Interactive PCPs [KR08].

A k-round public-coin IOP has k rounds of interaction. In the i-th round of interaction, the verifier sends a uniformly random message m_i to the prover; then the prover replies with a message π_i to the verifier. After k rounds of interaction, the verifier makes some queries to the oracles it received and either accepts or rejects.

An *IOP system* for a relation $\mathcal{R}$ with round complexity k and soundness error ε is a pair (P, V), where P, V are probabilistic algorithms, that satisfies the following properties. (See [BCS16; RRR16] for details.)

Completeness: For every instance-witness pair $(\mathtt{x}, \mathtt{w})$ in the relation $\mathcal{R}$, $(P(\mathtt{x}, \mathtt{w}), V(\mathtt{x}))$ is a $\mathsf{k}(n)$-round interactive oracle protocol with accepting probability 1.

Soundness: For every instance $\mathtt{x} \notin \mathcal{L}(\mathcal{R})$ and unbounded malicious prover $\tilde{P}$, $(\tilde{P}, V(\mathtt{x}))$ is a $\mathsf{k}(n)$-round interactive oracle protocol with accepting probability at most $\varepsilon(n)$.

Like the IP model, a fundamental measure of efficiency is the round complexity k. Like the PCP model, two additional fundamental measures of efficiency are the *proof length* p, which is the total number of alphabet symbols in all of the prover's messages, and the *query complexity* q, which is the total number of locations queried by the verifier across all of the prover's messages.

We say that an IOP system is *non-adaptive* if the verifier queries are non-adaptive, namely, the queried locations depend only on the verifier's inputs and its randomness. All of our IOP systems will be non-adaptive.

4.2.1 IOPs of proximity

An IOP *of Proximity* extends an IOP the same way that PCPs of Proximity extend PCPs. An *IOPP system* for a relation $\mathcal{R}$ with round complexity k, 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 $\mathcal{R}$, $(P(\mathtt{x}, \mathtt{w}), V^{\mathtt{w}}(\mathtt{x}))$ is a $\mathsf{k}(n)$-round interactive oracle protocol with accepting probability 1.

Soundness: For every instance-witness pair $(\mathtt{x}, \mathtt{w})$ with $\Delta(\mathtt{w}, \mathcal{R}|_{\mathtt{x}}) \geq \delta(n)$ and unbounded malicious prover $\tilde{P}$, $(\tilde{P}, V^{\mathtt{w}}(\mathtt{x}))$ is a $\mathsf{k}(n)$-round interactive oracle protocol with accepting probability at most $\varepsilon(n)$.

Efficiency measures for IOPPs are as for IOPs, except that we also count queries to the witness: if V makes at most $\mathsf{q}_{\mathtt{w}}$ queries to $\mathtt{w}$ and at most q_{π} queries to prover messages, the query complexity is $\mathsf{q} := \mathsf{q}_{\mathtt{w}} + \mathsf{q}_{\pi}$.

5 Oracle reductions

We define *interactive oracle reductions* (henceforth just *oracle reductions*), which, informally, are reductions from computational problems to the problem of testing membership of collections of oracles in a code.

The main result in this section is Lemma 5.4 (and an implication of it, Corollary 5.10), which enables the construction of IOPs by modularly combining oracle reductions and proximity tests. The ideas underlying oracle reductions are not new. Essentially all known constructions of PCPs/IPCPs/IOPs consist of two parts: (1) an encoding, typically via an algebraic code, that endows the witness with robust structure (often known as *arithmetization*); and (2) a procedure that locally tests this encoding (often known as *low-degree testing*).

Oracle reductions provide a formal method of constructing proof systems according to this framework. We use them to express results in Sections 7 to 9, which significantly simplifies exposition. Additionally, expressing our results as oracle reductions enables us to consider the efficiency of the oracle reduction itself as a separate goal from the efficiency of the low-degree test. In particular, future improvements in low-degree testing will lead immediately to improvements in our protocols.

This section has two parts: in Section 5.1 we define oracle reductions; then in Section 5.2, we introduce a special case of oracle reductions where the target code is the Reed–Solomon (RS) code. For this special case we give additional lemmas: we show that it suffices to prove a weaker soundness property, because it generically implies standard soundness; also, we show that all such oracle reductions admit a useful optimization which reduces the number of low-degree tests needed to a single one.

5.1 Definitions

Informally, an oracle reduction is an interactive public-coin protocol between a prover and a verifier that reduces membership in a language to a promise problem on oracles sent by the prover during the protocol.

In more detail, an oracle reduction from a language $\mathcal{L} \subseteq X$ to a relation $\mathcal{R}' \subseteq X' \times \Sigma^s$ is an interactive protocol between a prover and a verifier that both receive an instance $\mathbb{x} \in X$, where in each round the verifier sends a message and the prover replies with an oracle (or several oracles), as in the IOP model. Unlike in an IOP, the verifier *does not make any queries*. Instead, after the interaction the verifier outputs a list of claims of the form "$(\mathbb{x}, \mathbf{\Pi}) \in \mathcal{R}'$", which may depend on the verifier's randomness, where $\mathbb{x}' \in X'$ and $\mathbf{\Pi}$ is a deterministic oracle algorithm that specifies a string in Σ^s as follows: the i-th entry in Σ^s is computed as $\mathbf{\Pi}^{\pi_1,\ldots,\pi_r}(i)$, where π_j is the oracle sent by the prover in the j-th round. The reduction has the property that if $\mathbb{x} \in \mathcal{L}$ then all claims output by the verifier are true, and if instead $\mathbb{x} \notin \mathcal{L}$ then (with high probability over the verifier's randomness) at least one claim is false.

We refer to each oracle algorithm $\mathbf{\Pi}^{(j)}$ as a *virtual oracle* because $\mathbf{\Pi}^{(j)}$ represents an oracle that is derived from oracles sent by the prover. We are interested in virtual oracles $\mathbf{\Pi}^{(j)}$ where, for each i, the number of queries $\mathbf{\Pi}^{\pi_1,\ldots,\pi_r}(i)$ makes to the oracles is small. For simplicity, we also assume that the algorithms are non-adaptive in that the queried locations are independent of the answers to the queries.

A crucial property is that virtual oracles with small locality compose well, which allows us to compose oracle reductions. For this we need an oracle reduction *of proximity* (Definition 5.3), which we can view as an oracle reduction from a relation $\mathcal{R} \subseteq X \times \Sigma^s$ to another relation $\mathcal{R}' \subseteq X' \times \Sigma^{s'}$. Then we can construct an oracle reduction from $\mathcal{L}$ to $\mathcal{R}'$ by composing an oracle reduction A from $\mathcal{L}$ to $\mathcal{R}'$ with an oracle reduction of proximity B from $\mathcal{R}'$ to $\mathcal{R}''$. Such a reduction may output virtual oracles of the form $\mathbf{\Pi}_B^{\mathbf{\Pi}_A}$ where $\mathbf{\Pi}_B$ is a virtual oracle output by B and $\mathbf{\Pi}_A$ is a virtual oracle output by A. This can be expressed as a standard virtual oracle with access to the prover messages, and if $\mathbf{\Pi}_A$ and $\mathbf{\Pi}_B$ have small locality then so does $\mathbf{\Pi}_B^{\mathbf{\Pi}_A}$.

We now formalize the foregoing discussion, starting with the notion of a virtual oracle. Since the virtual oracles in this work are non-adaptive, we specify them via query ("pre-processing") and answer ("post-

processing") algorithms. The query algorithm receives an index $i \in [s]$ and computes a list of locations to be queried across oracles. The answer algorithm receives the same index i, and answers to the queries, and computes the value of the virtual oracle at location i. In other words, the answer algorithm computes the value of the virtual oracle at the desired location from the values of the real oracles at the queried locations.

Definition 5.1. *A* **virtual oracle $\mathbf{\Pi}$** *of length s over an alphabet Σ is a pair of deterministic polynomial-time algorithms (Q, A). Given any oracles $\pi_1, \ldots, \pi_r$ of appropriate sizes, these algorithms define an oracle $\mathbf{\Pi} \in \Sigma^s$ given by $\mathbf{\Pi}[\pi_1, \ldots, \pi_r](i) \coloneqq A(i, (\pi_j[k])_{(j,k)\in Q(i)})$ for $i \in [s]$. $\mathbf{\Pi}$ is ℓ-**local** if $\max_{i\in[s]} |Q(i)| \leq \ell$.*

Observe that the definition of a virtual oracle given above is equivalent to saying that $\mathbf{\Pi}$ is an algorithm with non-adaptive query access to $\pi_1, \ldots, \pi_r$. Where convenient we will use this perspective.

We now define the notion of an oracle reduction. Since in this work we primarily deal with relations, rather than languages, we define our reductions accordingly.

Definition 5.2. *An* **oracle reduction** *from a relation $\mathcal{R}$ to a relation $\mathcal{R}'$ with base alphabet Σ is an interactive protocol between a prover P and verifier V that works as follows. The prover P takes as input an instance-witness pair $(\mathbb{x}, \mathbb{w})$ and the verifier V takes as input the instance $\mathbb{x}$. In each round, V sends a message $m_i \in \{0,1\}^*$, and P replies with an oracle $\pi_i \in \Sigma_i^*$ over an alphabet $\Sigma_i = \Sigma^{s_i}$; let $\pi_1, \ldots, \pi_r$ be all oracles sent.[7] After the interaction, V outputs a list of instances $(\mathbb{x}^{(1)}, \ldots, \mathbb{x}^{(m)})$ and a list of virtual oracles $(\mathbf{\Pi}^{(1)}, \ldots, \mathbf{\Pi}^{(m)})$ over alphabets $\Sigma'_1, \ldots, \Sigma'_m$ respectively, where $\Sigma'_i = \Sigma^{s'_i}$.*

We say that the oracle reduction has **soundness error** ϵ *and* **distance** δ *if the following conditions hold.*

- Completeness: *If $(\mathbb{x}, \mathbb{w}) \in \mathcal{R}$ then, with probability 1 over the verifier's randomness, for every $j \in [m]$ it holds that $\left(\mathbb{x}^{(j)}, \mathbf{\Pi}^{(j)}[\pi_1, \ldots, \pi_r]\right) \in \mathcal{R}'$ where $(\mathbb{x}^{(j)}, \mathbf{\Pi}^{(j)})_{j\in[m]} \leftarrow (P(\mathbb{x}, \mathbb{w}), V(\mathbb{x}))$.*

- Soundness:[8] *If $\mathbb{x} \notin \mathcal{L}(\mathcal{R})$ then for any prover $\tilde{P}$, with probability $1-\epsilon$ over the verifier's randomness, there exists $j \in [m]$ such that $\Delta(\mathbf{\Pi}^{(j)}[\pi_1, \ldots, \pi_r], \mathcal{R}'|_{\mathbb{x}^{(j)}}) > \delta$ where $(\mathbb{x}^{(j)}, \mathbf{\Pi}^{(j)})_{j\in[m]} \leftarrow (P(\mathbb{x}, \mathbb{w}), V(\mathbb{x}))$.*

An oracle reduction is *public coin* if all of the verifier's messages consist of uniform randomness. All of the oracle reductions we present in this paper are public coin. Note that we can always choose the base alphabet Σ to be $\{0,1\}$, but it will be convenient for us to use a larger base alphabet.

This above definition can be viewed as extending the notion of *linear-algebraic CSPs* [BCGV16], and indeed Lemma 5.4 below gives a construction similar to the "canonical" PCP described in that work.

It will be useful to *compose* oracle reductions. As in the PCP setting, for this we will require an object with a stronger *proximity soundness* property.

Definition 5.3. *An* **oracle reduction of proximity** *is as in Definition 5.2 except that, for a given proximity parameter $\delta_0 \in (0,1)$, the soundness condition is replaced by the following one.*

- Proximity soundness: *If $(\mathbb{x}, \mathbb{w})$ is such that $\Delta(\mathbb{w}, \mathcal{R}|_{\mathbb{x}}) > \delta_0$ then for any prover $\tilde{P}$, with probability $1-\epsilon$ over the verifier's randomness, there exists $j \in [m]$ such that $\Delta(\mathbf{\Pi}^{(j)}[\pi_1, \ldots, \pi_r], \mathcal{R}'|_{\mathbb{x}^{(j)}}) > \delta(\delta_0)$ where $(\mathbb{x}^{(j)}, \mathbf{\Pi}^{(j)})_{j\in[m]} \leftarrow (P(\mathbb{x}, \mathbb{w}), V(\mathbb{x}))$.*

In the PCPP literature the foregoing soundness property is usually known as *robust soundness*, and the condition is expressed in terms of expected distance. The definition given here is more convenient for us.

Efficiency measures. There are several efficiency measures that we study for an oracle reduction.

[7] Sometimes it is convenient to allow the prover to reply with multiple oracles $\pi_{i,1}, \pi_{i,2}, \ldots$; all discussions extend to this case.

[8] This is analogous to the "interactive soundness error" ε_{i} in [BCRSVW19].

- An oracle reduction has r *rounds* if the interactive protocol realizing it has r rounds.
- An oracle reduction has m *virtual oracles* and *locality* ℓ if the verifier outputs at most m virtual oracles $\{\mathbf{\Pi}^{(j)} = (Q_j, A_j)\}_{j\in[m]}$, and it holds that $\max_{i\in[s]} |\cup_{j=1}^{m} Q_j(i)| \leq \ell$. Note that the answer to a single query may consist of multiple symbols over the base alphabet Σ, but we count the query only once.
- An oracle reduction has *length* $s = \sum_{i=1}^{r} s_i|\pi_i|$ over the base alphabet. Its length in *bits* is $s \log |\Sigma|$.

Other efficiency measures include the running time of the prover and of the verifier.

Oracle reductions combine naturally with proofs of proximity to produce IOPs. The following lemma is straightforward, and we state it without proof.

Lemma 5.4. *Suppose that there exist:*

(i) an r-round oracle reduction from $\mathcal{R}$ to $\mathcal{R}'$ over base alphabet Σ with soundness error ϵ, distance δ, length s, locality ℓ, and m virtual oracles;

(ii) an r'-round IOPP for $\mathcal{R}'$ over alphabet Σ with soundness error ϵ', proximity parameter $\delta' \leq \delta$, length s', and query complexity $(\mathsf{q}_\mathsf{w}, \mathsf{q}_\pi)$.

Then there exists an $(r + mr')$-round IOP for $\mathcal{R}$ with soundness error $\epsilon + \epsilon'$, length $s + s' \cdot m$ over Σ, and query complexity $(\mathsf{q}_\mathsf{w} \cdot \ell + \mathsf{q}_\pi) \cdot m$.

5.2 Reed–Solomon oracle reductions

In this work we focus on a special class of oracle reductions, in which we reduce to membership in the Reed–Solomon code, and where the virtual oracles have a special form. These reductions coincide with "RS-encoded IOPs" [BCRSVW19, Definition 4.6], which we recast in the language of virtual oracles.

We first define the notion of a *rational constraint*, a special type of virtual oracle that is "compatible" with the (interleaved) Reed–Solomon code.

Definition 5.5. *A* **rational constraint** *is a virtual oracle $\mathbf{\Pi} = (Q, A)$ over a finite field $\mathbb{F}$ where $Q(\alpha) = ((1, \alpha), \ldots, (r, \alpha))$ and $A(\alpha, \beta_1, \ldots, \beta_r) = N(\alpha, \beta_1, \ldots, \beta_r)/D(\alpha)$, for two arithmetic circuits (without division gates) $N \colon \mathbb{F}^{\sum_i s_i} \to \mathbb{F}$ and $D \colon \mathbb{F} \to \mathbb{F}$.*

A Reed–Solomon (RS) oracle reduction is a reduction from some relation to membership in the Reed–Solomon code, where additionally every oracle is a rational constraint.

Definition 5.6. *A* **Reed–Solomon (RS) oracle reduction** *over a domain $L \subseteq \mathbb{F}$ is an oracle reduction, over the base alphabet $\mathbb{F}$, from a relation $\mathcal{R}$ to the interleaved Reed–Solomon relation*

$$\mathcal{R}^*_{\mathrm{RS}} \coloneqq \left\{ (\vec{\rho}, f) \text{ s.t. } \vec{\rho} \in (0,1]^*,\ f \colon L \to \mathbb{F} \text{ is a codeword in } \mathrm{RS}\left[L, \vec{\rho}\right] \right\}$$

where every virtual oracle output by the verifier is a rational constraint, except for a special instance $(\vec{\rho}_0, \mathbf{\Pi}_0)$, which the verifier must output. $\mathbf{\Pi}_0$, over alphabet $\mathbb{F}^{\sum_i s_i}$, is given by $\mathbf{\Pi}_0(\alpha) = (\pi_1(\alpha), \ldots, \pi_r(\alpha))$ (i.e., it is a stacking of the oracles sent by the prover).

In this work we will assume throughout that L comes a family of subgroups (of a family of fields) such that there is an encoding algorithm for the Reed–Solomon code on domain L with arithmetic complexity $O(|L| \log |L|)$.

Note that $\mathbf{\Pi}_0$ is not a rational constraint because its alphabet is not $\mathbb{F}$. Later we will also refer to the *non-interleaved* Reed–Solomon relation $\mathcal{R}_{\mathrm{RS}} := \{(\rho, f) : \rho \in (0, 1], f \in \mathrm{RS}\left[L, \rho\right]\}$.

RS oracle reductions have a useful property: if the soundness condition holds for $\delta = 0$, then the soundness condition also holds for a distance $\delta > 0$ related to the *maximum rate* of the reduction. Informally,

the maximum rate is the maximum over the (prescribed) rates of codewords sent by the prover and those induced by the verifier's rational constraints. To define it, we need notation for the *degree* and *rate* of a circuit.

Definition 5.7. *The* **degree** *of an arithmetic circuit $C\colon \mathbb{F}^{1+\ell} \to \mathbb{F}$ on input degrees $d_1, \ldots, d_\ell \in \mathbb{N}$, denoted $\deg(C; d_1, \ldots, d_\ell)$, is the smallest integer e such that for all $p_i \in \mathbb{F}^{\leq d_i}[X]$ there exists a polynomial $q \in \mathbb{F}^{\leq e}[X]$ such that $C(X, p_1(X), \ldots, p_\ell(X)) \equiv q(X)$. Given domain $L \subseteq \mathbb{F}$ and rates $\vec{\rho} \in (0,1]^\ell$, the* **rate** *of C is $\mathrm{rate}(C; \vec{\rho}) := \deg(C; \rho_1|L|, \ldots, \rho_\ell|L|)/|L|$. (The domain L will be clear from context.) Note that if $\ell = 0$ then this notion of degree coincides with the usual one (namely, $\deg(C)$ is the degree of the polynomial described by C), and $\mathrm{rate}(C) := \deg(C)/|L|$.*

An oracle reduction has ***maximum rate*** ρ^* if, for every rational constraint $(\sigma, \mathbf{\Pi})$ output by the verifier, $\max(\mathrm{rate}(N; \vec{\rho}_0), \sigma + \mathrm{rate}(D)) \leq \rho^*$. This expression is motivated by the proof of the following lemma; see [BCRSVW19, Proof of Theorem 9.1] for details.

Lemma 5.8. *Suppose that an RS oracle reduction with maximum rate ρ^* satisfies the following* **weak soundness** *condition: if $\mathbb{x} \notin \mathcal{L}(\mathcal{R})$ then for any prover $\tilde{P}$, with probability $1 - \epsilon$ over the verifier's randomness, there exists $j \in [m]$ such that $(\rho^{(j)}, \mathbf{\Pi}^{(j)}[\pi_1, \ldots, \pi_n]) \notin \mathcal{R}_{\mathrm{RS}}$. Then the reduction satisfies the standard soundness condition (see Definition 5.2) with soundness error ϵ and distance $\delta := \frac{1}{2}(1 - \rho^*)$.*

This means that for the oracle reductions in this paper *we need only establish weak soundness*. Also, one can see that RS oracle reductions have locality r (the number of rounds), since $|Q(\alpha)| = r$ for all $\alpha \in L$.

The following lemma shows that, for RS oracle reductions, it suffices to run the proximity test on a single virtual oracle. This reduces the query complexity and proof length when we apply Lemma 5.4.

Lemma 5.9. *Suppose that there exists an r-round RS oracle reduction from $\mathcal{R}$ over domain L, m virtual oracles, soundness error ϵ, maximum rate ρ^*, and distance δ. Then there is an r-round oracle reduction from $\mathcal{R}$ to the* non-interleaved *Reed–Solomon relation $\mathcal{R}_{\mathrm{RS}}$ with locality r,* **one** *virtual oracle, soundness error $\epsilon + |L|/|\mathbb{F}|$, maximum rate ρ^*, and distance $\min(\delta, (1 - \rho^*)/3, (1 - 2\rho^*)/2)$.*

Proof. Implicit in [BCRSVW19, Proof of Theorem 9.1], where it follows from [BKS18]. □

Combining Lemmas 5.4, 5.8 and 5.9 yields the following useful corollary. We shall invoke it, in Section 10, on the two main building blocks obtained in this paper in order to prove our main result.

Corollary 5.10. *Suppose that there exist:*

(i) an r-round RS oracle reduction from $\mathcal{R}$ over domain L, m virtual oracles, length s and rate ρ^* that satisfies the weak soundness condition with soundness error ϵ;

(ii) an r'-round IOP of proximity for $\mathcal{R}_{\mathrm{RS}}$ with soundness error ϵ', proximity parameter $\delta' < \min((1 - \rho^*)/3, (1 - 2\rho^*)/2)$, length p and query complexity $(\mathsf{q}_{\mathbb{w}}, \mathsf{q}_\pi)$.

Then there exists an $(r + r')$-round IOP for $\mathcal{R}$ with soundness error $\epsilon + \epsilon' + \frac{|L|}{|\mathbb{F}|}$, length $s + \mathsf{p}$ and query complexity $\mathsf{q}_{\mathbb{w}} \cdot r + \mathsf{q}_\pi$.

6 Trace embeddings

The IOPs that we construct rely on the algebraic structure of univariate polynomials. For example, all prover messages are (allegedly) Reed–Solomon codewords. Since our proof systems argue the validity of execution traces, which are two-dimensional "tall and skinny" tables representing the state of a small number of registers in each step of a long computation, we need to embed such traces into univariate polynomials, in a way that allows us to efficiently reconstruct the two-dimensional structure and traverse it. Ad-hoc methods for obtaining such embeddings are ubiquitous in the PCP literature, and typically rely on the algebraic structure of the underlying field; see, e.g., [PS94; HS00; BS08; BGHSV05; BCGT13b; CZ15] for examples.

We introduce *trace embeddings*, a notion that abstracts the foregoing methods and encapsulates the algebraic structure required to instantiate them. Throughout, by "efficient" we mean a function that can be evaluated in time that is polylogarithmic in the size of its domain (which is the usual notion of efficiency given an appropriate encoding of the input). Informally, a trace embedding has two components:

1. A *bivariate embedding*, which encodes a table $A \in \mathbb{F}^{N \times n}$ into a function $f_A \colon H \to \mathbb{F}$, with $H \subseteq \mathbb{F}$ of size Nn, whose (univariate) low-degree extension admits efficient extraction of A's row/column structure.

   In more detail, a bivariate embedding (Definition 6.3) is a method of embedding two-dimensional data, indexed by coordinates $(h_1, h_2) \in H_1 \times H_2$ (with $H_1, H_2 \subseteq \mathbb{F}$), into one-dimensional data, indexed by a single coordinate $h \in H$ (with $H \subseteq \mathbb{F}$), such that (h_1, h_2) can be efficiently derived from h. Also, the embedding's projections to its two coordinates have efficiently-computable low-degree extensions.

2. A *successor ordering*, which induces a total order on the domain and allows for efficiently moving from each element to its successor. In more detail, we equip H_1 with an ordering $\gamma \colon [|H_1|] \to H_1$ given by a "first" element $\mathbf{1}_{H_1} \in H_1$ and an efficiently-computable low-degree polynomial N. The ordering is $\gamma(1) \coloneqq \mathbf{1}_{H_1}$ and $\gamma(i+1) \coloneqq N(\gamma(i))$ for every $i \in \{1, \ldots, |H_1| - 1\}$.

Note the asymmetry in the definition: we require H_1 to have a successor ordering, whereas we require nothing of H_2. This is because in our application H_1 will be large (roughly the length of the computation) whereas H_2 will be small (roughly the number of registers in the computation).

For convenience, we first present the formal definition of trace embeddings below, and discuss the components it uses, as well as their properties, in the following subsections.

Definition 6.1. *Let $\mathbb{F}$ be a finite field. A* **trace embedding** *is a tuple $\mathcal{T} = (\Phi, \mathcal{O}, \gamma)$ where $\Phi \colon H \to H_1 \times H_2$ is an efficient bivariate embedding in $\mathbb{F}$ (see Section 6.1), $\mathcal{O}$ an efficient successor ordering on H_1 (see Section 6.2), and $\gamma \colon [|H_2|] \to H_2$ an ordering on H_2. Additionally, we require that the vanishing polynomial $\mathbb{Z}_{H_1}$ and its (standard formal) derivative can be evaluated anywhere on $\mathbb{F}$ in $\mathrm{polylog}(|H|)$ field operations.*

Letting $N \coloneqq |H_1|$ and $n \coloneqq |H_2|$, a trace embedding $\mathcal{T}$ induces a bijection $\mathcal{T} \colon [N] \times [n] \to H$ defined as $\mathcal{T}(i,j) \coloneqq \Phi^{-1}(\gamma_{\mathcal{O}}(i), \gamma(j))$ for every $i \in [N]$ and $j \in [n]$, and $\gamma_{\mathcal{O}}$ being the ordering induced by $\mathcal{O}$. In light of this, we sometimes simply write $\mathcal{T} \colon [N] \times [n] \to H$ to refer to a trace embedding (instead of writing the tuple that defines it), which means that we consider *any* trace embedding that induces such a bijection.

In the remainder of this section, we give the formal definitions of bivariate embeddings and successor orderings, and prove the following lemma about the existence and efficient realizability of trace embeddings.

Lemma 6.2. *The field $\mathbb{F}$ of size p^e, for prime p, has trace embeddings of all sizes (N, n) for which either:*

(i) Nn divides $p^e - 1$, and N, n are coprime; or
(ii) $N = p^i$ and $n = p^j$ for all i, j such that $i + j < e$.

Moreover, there is a polynomial-time algorithm that, on input a description of $\mathbb{F}$, integers N and 1^n, outputs a description of a trace embedding of size $N \times n$, if N, n satisfy one of the above conditions (and $\perp$ otherwise).

6.1 Bivariate embeddings

We define the notion of an (efficient) bivariate embedding, and show that it can be instantiated via the additive or multiplicative subgroup structure of a finite field. Then in Section 6.1.1, we state some simple linear-algebraic implications of bivariate embeddings that are used frequently in subsequent sections.

Definition 6.3. *Let $\mathbb{F}$ be a finite field. A (low-degree)* **bivariate embedding** *in $\mathbb{F}$ is a tuple $(\Phi_1, \Phi_2, H, H_1, H_2)$ where H, H_1, H_2 are subsets of $\mathbb{F}$, $\Phi_1 \in \mathbb{F}[X]$ is a polynomial of degree $|H_2|$, $\Phi_2 \in \mathbb{F}[X]$ is a polynomial of degree $|H_1|$, such that the function $\Phi\colon H \to H_1 \times H_2$ defined as $\Phi(h) \coloneqq (\Phi_1(h), \Phi_2(h))$ is a bijection.*

A bivariate embedding $\Phi\colon H \to H_1 \times H_2$ is **efficient** *if Φ_1 and Φ_2 can be evaluated at any $\alpha \in \mathbb{F}$ in $\mathrm{polylog}(|H|)$ field operations.*

For notational convenience we refer to a bivariate embedding $(\Phi_1, \Phi_2, H, H_1, H_2)$ with the notation $\Phi\colon H \to H_1 \times H_2$, defined as in Definition 6.3, leaving the polynomials Φ_1, Φ_2 implicit.

The degrees of Φ_1, Φ_2 will impact the efficiency of our proof system, and so we aim to minimize them. In particular, the degree choices in the definition are *minimal*: if Φ_1 has degree $d > 0$ then $\mathbb{Z}_{H_1}(\Phi_1(X))$ has degree $|H_1| \cdot d$, is not the zero polynomial, and is zero everywhere on H, so $d \geq |H|/|H_1| = |H_2|$; a similar statement holds for Φ_2. Since we achieve these degree bounds in the constructions below, we make these choices part of the above definition. In particular, $\mathbb{Z}_{H_1}(\Phi_1(X))$ and $\mathbb{Z}_{H_2}(\Phi_2(X))$ are each multiples of $\mathbb{Z}_H$.

While for any $H, H_1, H_2 \subseteq \mathbb{F}$ with $|H| = |H_1| \cdot |H_2|$ there exist (many) bijections $H \to H_1 \times H_2$, the aforementioned degree constraints severely limit our choices for these sets. All known constructions use the group structure of H. It remains an intriguing open question to determine whether other constructions exist.

Efficiency is an even stronger requirement. Since the "truth table" of Φ is of size $|H|$, it must be that H has some inherent product structure that we can exploit. In our protocols, H will be an additive or multiplicative subgroup of $\mathbb{F}$ and H_1, H_2 will be isomorphic to subgroups of H whose product is H.

We now construct efficient bivariate embeddings, over multiplicative and additive subgroups of the field.

Lemma 6.4. *Let $\mathbb{F}$ be a finite field, and let H, H_1, H_2 be* **multiplicative** *subgroups of $\mathbb{F}$ such that $|H| = |H_1| \cdot |H_2|$ and $\gcd(|H_1|, |H_2|) = 1$. Then, there exists an efficient bivariate embedding $\Phi\colon H \to H_1 \times H_2$.*

Proof. Map each $h \in H$ to the pair $\Phi(h) \coloneqq (h^{|H_2|}, h^{|H_1|}) \in H_1 \times H_2$. By the Chinese Remainder Theorem, Φ is an isomorphism. Moreover, the polynomials $\Phi_1(X) \coloneqq X^{|H_2|}$ and $\Phi_2(X) \coloneqq X^{|H_1|}$ agree with Φ on H, and can be evaluated in $O(\log(|H_1|) + \log(|H_2|)) = O(\log(|H|))$ field operations. □

Lemma 6.5. *Let $\mathbb{F}$ be a finite field, and let V, W be* **additive** *subgroups of $\mathbb{F}$ with respective sizes m, n and such that $V \cap W = \{0\}$. Then there exists an efficient bivariate embedding $\Phi\colon V \oplus W \to \mathbb{Z}_W(V) \times \mathbb{Z}_V(W)$.*

Proof. Map each $v + w \in V \oplus W$ (with $v \in V$ and $w \in W$) to the pair $\Phi(v + w) \coloneqq (\mathbb{Z}_W(v), \mathbb{Z}_V(w)) \in \mathbb{Z}_W(V) \times \mathbb{Z}_V(W)$, where $\mathbb{Z}_W$ and $\mathbb{Z}_V$ are the vanishing polynomials of V and W. Since $\mathbb{Z}_W$ is injective on V and $\mathbb{Z}_V$ is injective on W, Φ is a bijection. Moreover, the polynomials $\Phi_1(X) \coloneqq \mathbb{Z}_W(X)$ and $\Phi_2(X) \coloneqq \mathbb{Z}_V(X)$ can be evaluated in $O(\log^2(|W|) + \log^2(|V|)) = O(\log^2(|H|))$ field operations. □

6.1.1 Linear-algebraic properties

We use bivariate embeddings as a natural, and algebraically friendly, way to identify the tensor product space $\mathbb{F}^{H_1} \otimes \mathbb{F}^{H_2}$ with $\mathbb{F}^H$. This is expressed via the definitions and propositions below, which we state without proof. The propositions follow from standard linear algebra and the fact that a bivariate embedding extends bijections $\gamma_1\colon H_1 \to [|H_1|]$ and $\gamma_2\colon H_2 \to [|H_2|]$ to a bijection $\gamma\colon H \to [|H_1|] \times [|H_2|]$.

Definition 6.6. *Define $\otimes_\Phi \colon \mathbb{F}^{H_1} \times \mathbb{F}^{H_2} \to \mathbb{F}^H$ to be the bilinear function that maps $u \in \mathbb{F}^{H_1}$ and $v \in \mathbb{F}^{H_2}$ to $u \otimes_\Phi v \in \mathbb{F}^H$ where $(u \otimes_\Phi v)(h) \coloneqq u(\Phi_1(h)) \cdot v(\Phi_2(h))$ for every $h \in H$.*

Proposition 6.7. *The function $\otimes_\Phi \colon \mathbb{F}^{H_1} \times \mathbb{F}^{H_2} \to \mathbb{F}^H$ is a tensor product.*

Definition 6.8. *Let $\Phi \colon H \to H_1 \times H_2$ be a bivariate embedding. The* **Kronecker product** *of matrices $A \in \mathbb{F}^{H_1 \times H_1}$ and $B \in \mathbb{F}^{H_2 \times H_2}$ with respect to Φ is the matrix $A \otimes_\Phi B \in \mathbb{F}^{H \times H}$ where $(A \otimes_\Phi B)(h, h') \coloneqq A(\Phi_1(h), \Phi_1(h')) \cdot B(\Phi_2(h), \Phi_2(h'))$ for every $h, h' \in H$.*

Proposition 6.9. *Let $A \in \mathbb{F}^{H_1 \times H_1}$ and $B \in \mathbb{F}^{H_2 \times H_2}$. Then $(A \otimes_\Phi B)$ is the unique linear map such that, for every $u \in \mathbb{F}^{H_1}$ and $v \in \mathbb{F}^{H_2}$, $(A \otimes_\Phi B)(u \otimes_\Phi v) = (Au) \otimes_\Phi (Bv)$, under usual matrix multiplication.*

6.2 Successor orderings

A *successor ordering* of a set S is a pair $\mathcal{O} = (\mathbf{1}_S, N)$ where $\mathbf{1}_S \in S$ is a distinguished first element and $N \in \mathbb{F}[X]$ is a degree-1 *successor polynomial* that induces an ordering by mapping each element of S to its "successor". That is, $\mathcal{O}$ induces a bijection $\gamma_\mathcal{O} \colon [|S|] \to S$ given by $\gamma_\mathcal{O}(1) = \mathbf{1}_S$, $\gamma_\mathcal{O}(2) = N(\mathbf{1}_S)$, $\gamma_\mathcal{O}(3) = N(N(\mathbf{1}_S))$, and so on. We call $\mathcal{O}$ *efficient* if N can be evaluated in $\operatorname{polylog}(|S|)$ field operations.

We show that all multiplicative subgroups S of $\mathbb{F}$ have efficient successor orderings, by relying on any isomorphism $S \cong \mathbb{Z}_{|S|}$. We also show that, for $|\mathbb{F}| = p^e$, there exists an additive subgroup S of $\mathbb{F}$ of size p^i with an efficient successor ordering, if p is small enough. This relies on "simulating" the field $\mathbb{F}_{p^i}$ inside S.

6.2.1 Multiplicative subgroups

The construction for multiplicative subgroups is straightforward. The following lemma gives the construction of a successor ordering for any multiplicative subgroup of $\mathbb{F}$, that is, of any size dividing $|\mathbb{F}| - 1$.

Lemma 6.10. *Let $\mathbb{F}$ be a finite field. Every multiplicative subgroup S of $\mathbb{F}$ has an efficient successor ordering.*

Proof. Choose a generator g of S (note that S is cyclic), and then let $\mathbf{1}_S \coloneqq 1_\mathbb{F}$ and $N(X) \coloneqq gX$. □

6.2.2 Additive subgroups

In order to give the construction for additive subgroups, we first need to generalize the definition of successor ordering given above. That is, we need to be more permissive about the successor polynomials N we allow.

In later sections, we use successor orderings by composing the polynomial N with other polynomials that enforce correct computation. We must ensure that this composition does not have too large of a degree. Ideally we would like N to have degree $O(1)$ because then $\deg(g \circ N) = O(\deg g)$ for any $g \in \mathbb{F}[X]$. When S is a multiplicative subgroup we can achieve this (as N has degree 1), but when S is an additive subgroup we get $\deg(N) = \Omega(|S|)$, which would give $\deg(g \circ N) = \Omega(|S| \deg g)$. Since $|S|$ and $\deg(g)$ will be each approximately the computation length T, the degree of the composed polynomial would be $\Omega(T^2)$. This would prevent us from achieving, e.g., IOPs of linear proof length.

To deal with this, we use the fact that the additive N satisfies a useful structural property, to which we refer as being *piecewise polynomial*. A function f is a piecewise polynomial of degree d with respect to a partition $(S_1, \ldots, S_\ell)$ of S if there exist polynomials $(f_1, \ldots, f_\ell)$ of degree d such that

$$\forall\, i \in [\ell],\, \forall\, \alpha \in S_i\,,\ f(\alpha) = f_i(\alpha)\ .$$

If s_i is a degree-$|S|$ extension of the indicator for S_i in S then $\sum_{i=1}^{\ell} s_i(X) g(f_i(X))$ has degree $O(|S| + \deg g)$ and agrees with $g \circ N$ on S. This is an additive rather than multiplicative increase in the degree, and later will yield a degree bound of $O(T)$ as required. We now define formally the notion of a piecewise polynomial.

Definition 6.11. *Let $\mathbb{F}$ be a finite field and $S \subseteq \mathbb{F}$. A* **piecewise polynomial on** *S is a pair $F = (\mathcal{S}, \mathcal{F})$ where $\mathcal{S} = (S_1, \ldots, S_\ell)$ is a partition of S and $\mathcal{F} = (f_1, \ldots, f_\ell) \in \mathbb{F}[X]^\ell$. Let s_i be the unique extension of degree less than $|S|$ of the indicator for S_i in S. The* **value** *of $F = (\mathcal{S}, \mathcal{F})$ at $\alpha \in \mathbb{F}$ is $F(\alpha) \coloneqq \sum_{i=1}^{\ell} s_i(\alpha) f_i(\alpha)$.*

A piecewise polynomial $(\mathcal{S}, \mathcal{F})$ has **piecewise degree** *d if $\max_{f \in \mathcal{F}} \deg(f) \leq d$. A piecewise polynomial is* **efficient** *if each s_i, f_i can be evaluated in time $\mathrm{polylog}(|S|)$, and $\ell = \mathrm{polylog}(|S|)$.*

The following proposition shows that the special structure of piecewise polynomials allows for composition with an additive, rather than multiplicative, dependence on $|S|$.

Proposition 6.12. *If $F = (\mathcal{S}, \mathcal{F})$ is piecewise polynomial on S with degree d and $g \in \mathbb{F}[X]$, then there is a polynomial h of degree $|S| + d \cdot \deg(g)$ that agrees with $g \circ F$ on S. Moreover, if F is efficient and g can be evaluated in $\mathrm{polylog}(|S|)$ field operations then h can be evaluated in $\mathrm{polylog}(|S|)$ field operations.*

Proof. Let $h(X) \coloneqq \sum_{i=1}^{\ell} s_i(X) g(f_i(X)) \in \mathbb{F}[X]$, where s_i is the unique extension of degree less than $|S|$ of the indicator function of S_i in S. Then since the s_i are indicators for a partition of S,

$$\forall\, \alpha \in S\,,\ g(f(\alpha)) = g\left(\sum_{i=1}^{\ell} s_i(\alpha) f_i(\alpha)\right) = \sum_{i=1}^{\ell} s_i(\alpha) g(f_i(\alpha)) = h(\alpha)\ .$$

Observe that h has the required degree and can be evaluated in $\mathrm{polylog}(|S|)$ field operations. □

Now we are ready to formally define a successor ordering.

Definition 6.13. *Let $\mathbb{F}$ be a finite field and $S \subseteq \mathbb{F}$. A* **successor ordering** *on S is a pair $\mathcal{O} = (\mathbf{1}_S, N)$ where $\mathbf{1}_S \in S$ and N is a piecewise polynomial on S of degree 1 such that $S = \{\alpha_1, \ldots, \alpha_{|S|}\}$, where*

$$\alpha_1 \coloneqq \mathbf{1}_S \text{ and } \alpha_{i+1} \coloneqq N(\alpha_i) \text{ inductively for every } i \in \{1, \ldots, |S| - 1\}.$$

Let $\gamma_{\mathcal{O}} \colon S \to \{1, \ldots, |S|\}$ be the ordering on S induced by $\mathcal{O}$, i.e., $\gamma_{\mathcal{O}}(\alpha_i) \coloneqq i$ for every $i \in \{1, \ldots, |S|\}$.

A successor ordering is **efficient** *if the piecewise polynomial N is efficient.*

Note that since a polynomial of degree d has a trivial piecewise representation of degree d, the foregoing definition also captures the simpler multiplicative case.

Let p be prime. For the field $\mathbb{F}$ of size p^e, we give constructions of efficient ordered subsets of any size p^i (for $1 \leq i < e$) provided p is small enough relative to i.

The following lemma appears in [BS08; BCGT13b; BBHR18a]. We restate it here in the language of successor orderings, and for completeness we also provide a proof.

Lemma 6.14. *Let $\mathbb{F}$ be a finite field of size p^e. For every $i \in \{1, \ldots, e-1\}$ there exists an additive subgroup S in $\mathbb{F}$ of size p^i with a successor ordering $\mathcal{O}$. If in addition $p = O(\log |S|)$ then $\mathcal{O}$ is efficient.*

Proof. We view $\mathbb{F}$ as $\mathbb{F}_p[\xi]/(f)$ for some $f \in \mathbb{F}_p[\xi]$. To prove the lemma, we first move to the ring $\mathbb{F}_p[\xi]$ and show that there exists a subspace of $\mathbb{F}_p[\xi]$ that is isomorphic to a "multiplicative" subset of $\mathbb{F}_p[\xi]$.

Claim 6.15. *Let $S \coloneqq \mathrm{span}\{1, \xi, \ldots, \xi^{i-1}\} \subseteq \mathbb{F}_p[\xi]$, and let $G \coloneqq \{0, 1, \xi, \xi^2, \ldots, \xi^{p^i - 2}\} \subseteq \mathbb{F}_p[\xi]$. Let $g(\xi) \in \mathbb{F}_p[\xi]$ be a primitive polynomial of degree i, and let $\varphi \colon \mathbb{F}_p[\xi] \to \mathbb{F}_p[\xi]$ be defined as $\varphi(f) \coloneqq f \bmod g$ where division is in the ring $\mathbb{F}_p[\xi]$. Then $S = \varphi(G)$.*

Proof of claim. It suffices to show that φ is a bijection on G, since S is the image of φ (i.e., $S = \varphi(\mathbb{F}_p[\xi])$). Since g is primitive, g has a root $\omega \in \mathbb{F}_{p^i}$ such that $\mathbb{F}_{p^i} = \{0, 1, \omega, \omega^2, \ldots, \omega^{p^i - 2}\}$. Since $\mathbb{F}_p[\xi]/(g(\xi))$ is isomorphic to $\mathbb{F}_{p^i}$ via the map $\xi \mapsto \omega$, we deduce that $|\varphi(G)| = |G|$. Thus $S = \varphi(G)$ as claimed. □

We have that S is isomorphic to $S + (f) \subseteq \mathbb{F}$, and so we can regard S as a subset of $\mathbb{F}$. We now construct a successor ordering $\mathcal{O} = (\mathbf{1}_S, N)$ on S. We set $\mathbf{1}_S \coloneqq 0$, and are then left to define the piecewise polynomial N. We divide S into cosets of $S' \coloneqq \operatorname{span}\{1, \xi, \ldots, \xi^{i-2}\}$ as $S = \bigcup_{c \in \mathbb{F}_p} c\xi^{i-1} + S'$. These cosets have the property that if $\varphi(\xi^j) \in c\xi^{i-1} + S'$ then $\varphi(\xi^{j+1}) = \xi \cdot \varphi(\xi^j) - c \cdot g(\xi)$. Note that this is true in $\mathbb{F}$ since it is true in $\mathbb{F}_p[\xi]$ and $\deg(f) > i = \deg(g)$.

Our partition of S consists of the sets $\{0\}$, $S' \setminus \{0\}$, and the set $S' + c\xi^{i-1}$ for all $c \in \mathbb{F}_p \setminus \{0\}$. The corresponding polynomial partition is $\{L_{S,0}, L_0(\mathbb{Z}_{S'}(X)) - L_{S,0}(X)\} \cup \{L_c \circ \mathbb{Z}_{S'}\}_{c \in \mathbb{F}_p}$, where

- $L_{S,0}$ is the unique polynomial of degree less than $|S|$ with $L_{S,0}(0) = 1$ and $L_{S,0}(\gamma) = 0$ for $\gamma \in S \setminus \{0\}$;
- L_c is the unique polynomial of degree less than p such that

$$L_c(c \cdot \mathbb{Z}_{S'}(\xi^{i-1})) = 1 \quad \text{and} \quad L_c(\gamma \cdot \mathbb{Z}_{S'}(\xi^{i-1})) = 0 \text{ for every } \gamma \in \mathbb{F}_p \setminus \{c\} \ .$$

By the $\mathbb{F}_p$-linearity of $\mathbb{Z}_{S'}$, we have that, for every $\alpha \in S$, $L_c(\mathbb{Z}_{S'}(\alpha)) = 1$ if $\alpha \in S' + c\xi^{i-1}$ and 0 otherwise. Hence this is indeed the desired partition. The value of N on $a \in S$ should be

$$N(a) = \begin{cases} 1 & \text{if } a = 0 \\ \xi \cdot a & \text{if } a \in S' \setminus \{0\} \\ \xi \cdot a - c \cdot g(\xi) & \text{if } a \in S' + c\xi^{i-1} \end{cases}$$

This gives $N(0) = 1 = \varphi(1)$, and $N(\varphi(\xi^j)) = \varphi(\xi^{j+1})$ for every $j \in \{0, 1, \ldots, p^i - 2\}$.

Thus, the polynomial

$$N(X) \coloneqq L_{S,0}(X) \cdot 1 + (L_0(\mathbb{Z}_{S'}(X)) - L_{S,0}(X)) \cdot \xi \cdot X + \sum_{c \in \mathbb{F}_p \setminus \{0\}} L_c(\mathbb{Z}_{S'}(X)) \cdot (\xi \cdot X - c \cdot g(\xi))$$

takes the correct values on S, and has piecewise degree 1.

Finally, observe that N can be evaluated in $\operatorname{poly}(p, \log(|S|))$ field operations, which is $\operatorname{polylog}(|S|))$ when $p = O(\log |S|)$. Indeed: (1) since S is a subspace, a Lagrange polynomial $L_{S,0}$ can be evaluated in $\operatorname{polylog}(|S|)$ field operations; (2) L_c is a Lagrange polynomial over the field of size p and thus can be evaluated in $O(\log p)$ field operations; and (3) since S' is a subspace, the vanishing polynomial $\mathbb{Z}_{S'}$ can be evaluated in $\operatorname{polylog}(|S'|)$ field operations. The dependence on p (rather than $\log p$) in the cost for evaluating N comes from adding the terms in the sum. □

7 A succinct lincheck protocol

We describe *succinct lincheck*, an oracle reduction for checking a useful class of succinctly-represented linear relations on Reed–Solomon codewords. This oracle reduction is later used to obtain an oracle reduction for R1CS automata (see Section 8), and can be viewed as a succinct analogue of the univariate lincheck protocol in [BCRSVW19]. We recall the *lincheck relation* defined in that work, and for simplicity focus on the special case of linear relations defined by *square* constraint matrices.

Definition 7.1. *The* **lincheck relation** $\mathcal{R}_{\text{LIN}}$ *consists of pairs* $\big((\mathbb{F}, L, H, \rho, M), (f_1, f_2)\big)$ *where* $\mathbb{F}$ *is a finite field,* $L, H \subseteq \mathbb{F}$*,* $\rho \in (0,1)$*,* $f_1, f_2 \in \mathrm{RS}\,[L, \rho]$*,* $M \in \mathbb{F}^{H \times H}$*, and* $\forall\, a \in H\ \ \hat{f}_1(a) = \sum_{b \in H} M_{a,b} \cdot \hat{f}_2(b)$*.*

Recall that the protocol for $\mathcal{R}_{\text{LIN}}$ in [BCRSVW19], which we outlined in Section 2.2, supports *any* constraint matrix $M \in \mathbb{F}^{H \times H}$, and so it cannot run in time $o(||M||)$ (let alone in our target time of polylogarithmic in $||M||$) for *every* matrix M, because the verifier must at least *read* the description of M.

In this section, we show that, for an expressive family of constraint matrices, we can design a "succinct" lincheck protocol wherein the verifier runs *exponentially* faster than in [BCRSVW19]. To this end, we first observe that the verifier's expensive work in the lincheck protocol consists of evaluating low-degree extensions of the two vectors $r_\alpha := (g_h(\alpha))_{h \in H} \in \mathbb{F}^H$ and $s_\alpha := r_\alpha M \in \mathbb{F}^H$; this involves $\Omega(||M|| + |H|)$ field operations. Then we show how to perform such evaluations in $\mathrm{polylog}(|H|)$ field operations (and thereby make the verifier exponentially faster) for a class of matrices M that arise from T-time computations. Informally, we rely on a clever choice of linearly-independent polynomials $(g_h(\alpha))_{h \in H}$ and also on special algebraic properties of the matrices M, related to the algebraic structure of $\mathbb{F}$.

We now motivate the algebraic properties that we use, and then state our result.

Semisuccinct matrices. The lincheck protocol in [BCRSVW19] chooses the linearly independent polynomials $(g_h(X))_{h \in H}$ to be the *standard basis* $(1, X, X^2, \ldots, X^{|H|-1})$. We do not know how to efficiently evaluate the low-degree extension of r_α with respect to this basis. But there is another natural choice of basis for polynomials, the *Lagrange basis* $(L_{H,h})_{h \in H}$. In Section 7.1 we show that the low-degree extension $\hat{r}_\alpha$ of r_α with respect to this basis has a simple form that allows one to evaluate $\hat{r}_\alpha$ in time $\mathrm{polylog}(|H|)$.

The foregoing suggests a natural condition on the matrix M to require: if we can efficiently compute a low-degree extension of a vector $v \in \mathbb{F}^H$, then we should also be able to efficiently compute a low-degree extension of the vector $vM \in \mathbb{F}^H$. This notion of (algebraic) *succinctness* is formally captured as follows.

Definition 7.2. *Let* $H \subseteq \mathbb{F}$ *and* $\kappa \colon \mathbb{N} \to \mathbb{N}$*. A vector* $v \in \mathbb{F}^H$ *is* d**-extendable** *if there is* $p \in \mathbb{F}[X]$ *of degree at most* d *that agrees with* v *on* H *and* p *can be evaluated at any* $\alpha \in \mathbb{F}$ *in* $\mathrm{polylog}(|H|)$ *field operations. A matrix* $A \in \mathbb{F}^{H \times H}$ *is* κ**-succinct** *if, for every* $d \geq |H| - 1$*,* vA *is* $\kappa(d)$*-extendable whenever* v *is* d*-extendable.*

The identity matrix trivially satisfies the above definition, and later on we will show that the matrix with 1s on the superdiagonal is also algebraically succinct for certain choices of $\mathbb{F}$ and H (see Lemma 8.4).

Unfortunately, Definition 7.2 turns out to be too restrictive for us. But it is a starting point for a more general notion that suffices. Concretely, the matrices that arise in Section 8 are not themselves succinct, but they can be *decomposed* into a part that is succinct and a part that is "small". This is analogous to a Turing machine, where a computation is specified by repeated applications of a small transition function.

Definition 7.3. *Let* $\Phi \colon H \to H_1 \times H_2$ *be a bivariate embedding, and let* $\kappa \colon \mathbb{N} \to \mathbb{N}$*. A matrix* $M \in \mathbb{F}^{H \times H}$ *is* (Φ, κ)**-semisuccinct** *if* $M = A \otimes_\Phi B$ *for* κ*-succinct* $A \in \mathbb{F}^{H_1 \times H_1}$ *and arbitrary* $B \in \mathbb{F}^{H_2 \times H_2}$*.*

In the definition there is an asymmetry between H_1 and H_2, since we think of H_1 as large and H_2 as small.

To handle semisuccinct matrices we need a slightly different property from the polynomials $(g_h(X))_{h\in H}$. Namely we need that the vector $r_\alpha := (g_h(\alpha))_{h\in H}$ can be written as a *tensor product* of vectors $r_\alpha^{(1)} \in \mathbb{F}^{H_1}$ and $r_\alpha^{(2)} \in \mathbb{F}^{H_2}$ such that we can efficiently compute the low-degree extension of $r_\alpha^{(1)}$. Then, by definition of the Kronecker product, $r_\alpha M = (r_\alpha^{(1)} A) \otimes (r_\alpha^{(2)} B)$. Since A is succinct and B is small, we can compute the LDEs of $r_\alpha^{(1)} A$ and $r_\alpha^{(2)} B$ efficiently, then use Lemma 7.7 to compute the LDE of their tensor product. In Section 7.2 we show how to construct $(g_h(X))_{h\in H}$ from the Lagrange bases on H_1 and H_2.

A succinct lincheck protocol. The main result of the section is an oracle reduction, with linear length and locality 2, from lincheck on sums of semisuccinct matrices. The verifier uses $\mathrm{poly}(|\mathbb{x}|) = \mathrm{poly}(\ell, n, \log N)$ field operations, which is *polylogarithmic* in the size of the succinct part of the matrix.

Lemma 7.4. *Let $\mathbb{F}$ be a finite field with an efficient bivariate embedding $\Phi\colon H \to H_1 \times H_2$ where H is a coset in $\mathbb{F}$. Suppose that $M \in \mathbb{F}^{H\times H}$ has the form $M = \sum_{i=1}^{\ell} M_i$ where each $M_i \in \mathbb{F}^{H\times H}$ is (Φ, κ)-semisuccinct.*

Setting $N := |H_1|$, $n := |H_2|$, and $d := n \cdot (N + \kappa(N))$, there is a 1-round RS oracle reduction of proximity (Protocol 7.12) for $\mathcal{R}_{\mathrm{LIN}}$ for instances of the form $\mathbb{x} = (\mathbb{F}, L, H, \rho, M)$ with the following parameters:

length	$\lvert L\rvert$	*soundness error*	$d/\lvert\mathbb{F}\rvert$	*prover time*	$\lvert L\rvert \log \lvert L\rvert + \lvert\lvert M\rvert\rvert$
locality	2	*distance*	$\frac{1}{2}(1-\rho-\frac{d}{\lvert L\rvert})$	*verifier time*	$\mathrm{poly}(\lvert\mathbb{x}\rvert)$

(Above $||M||$ denotes the number of non-zero entries in the matrix $M \in \mathbb{F}^{H\times H}$.)

Organization. In Sections 7.1 and 7.2 we develop algebraic preliminaries. In Section 7.3 we prove Lemma 7.4. In Section 7.4 we extend Lemma 7.4 to handle a *block-matrix* lincheck relation. This latter relation is the one that we actually use in Section 8 to obtain an oracle reduction for interactive R1CS automata.

7.1 Properties of the Lagrange basis

Let $\mathbb{F}$ be a finite field and S a subset of $\mathbb{F}$. The *Lagrange basis* over S are the polynomials $L_S := (L_{S,\alpha})_{\alpha\in S}$ where $L_{S,\alpha}$ is the unique polynomial of degree less than $|S|$ with $L_{S,\alpha}(\alpha) = 1$ and $L_{S,\alpha}(\gamma) = 0$ for all $\gamma \in S \setminus \{\alpha\}$. Recall that L_S is a basis for the (vector space of) polynomials in $\mathbb{F}[X]$ of degree less than $|S|$.

In this work it will be convenient to consider an *unnormalized* version of the Lagrange basis. We let $L'_{S,\beta} := \mathbb{Z}_S(X)/(X-\beta)$, and define the vector of polynomials

$$\mathbf{r}_S := (L'_{S,\beta})_{\beta\in S} = \left(\frac{\mathbb{Z}_S(X)}{X-\beta}\right)_{\beta\in S} \in \mathbb{F}[X]^S \ .$$

Observe that $L'_{S,\beta}$ is a polynomial of degree less than $|S|$ that is zero on $S \setminus \{\beta\}$, and is therefore equal to $L_{S,\beta}$ up to a multiplicative (nonzero) constant factor. It follows that $\mathbf{r}_S$ is also a basis for the vector space of polynomials of degree less than $|S|$. The following lemma shows that the unique low-degree extension of $\mathbf{r}_S \in \mathbb{F}[X]^S$, which is a bivariate polynomial $\hat{r}_S \in \mathbb{F}[X,Y]$, has a simple explicit form.

Lemma 7.5. *$\hat{r}_S(X,Y) := (\mathbb{Z}_S(X) - \mathbb{Z}_S(Y))/(X-Y)$ is the unique low-degree extension of $\mathbf{r}_S$.*

Proof. Observe that $\hat{r}_S$ is a polynomial of degree $|S|-1$ in Y because $X-Y$ divides $\mathbb{Z}_S(X) - \mathbb{Z}_S(Y)$. Moreover, for every $\beta \in S$, $\hat{r}_S(X,\beta) = \mathbb{Z}_S(X)/(X-\beta) = L'_{S,\beta}(X)$, which is the β-th entry of $\mathbf{r}_S$. □

Given a polynomial $f \in \mathbb{F}[X]$, we define $\mathbf{r}_S^f \in \mathbb{F}[X]^S$ to be the vector of polynomials obtained by composing each entry of $\mathbf{r}_S$ with f:

$$\mathbf{r}_S^f := (L'_{S,\beta} \circ f)_{\beta\in S} = \left(\frac{\mathbb{Z}_S(f(X))}{f(X)-\beta}\right)_{\beta\in S} \in \mathbb{F}[X]^S \ .$$

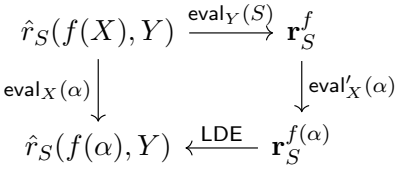

Figure 2: Commutative diagram showing the relationship between the vector of (univariate) polynomials $\mathbf{r}_S$ and the bivariate polynomial $\hat{r}_S$. The function $\mathsf{eval}_Y(S)$ maps a polynomial $g(X, Y)$ to the vector of polynomials $(g(X, \beta))_{\beta \in S}$. The function $\mathsf{eval}_X(\alpha)$ maps a polynomial $g(X, Y)$ to the polynomial $g(\alpha, Y)$. The function $\mathsf{eval}'_X(\alpha)$ maps a vector of polynomials $(g_\beta(X))_{\beta \in S}$ to the vector of polynomials $(g_\beta(\alpha))_{\alpha \in S}$. The function LDE maps a vector $(r_\beta)_{\beta \in S} \in \mathbb{F}^S$ to the unique polynomial g of degree less than $|S|$ such that $g(\beta) = r_\beta$ for all $\beta \in S$.

Note that when $f = \alpha \in \mathbb{F}$, this corresponds to evaluating each entry of the vector $\mathbf{r}_S$ at the point α. Also, Lemma 7.5 implies that the unique low-degree extension of $\mathbf{r}_S^f$ is $\hat{r}_S(f(X), Y)$. The next lemma shows that for certain sets S we can evaluate this low-degree extension very efficiently.

Lemma 7.6. *Let S be a subset of $\mathbb{F}$ whose vanishing polynomial $\mathbb{Z}_S$ and its formal derivative $\mathrm{D}\mathbb{Z}_S$ can both be evaluated at any point in $\mathrm{polylog}(|S|)$ field operations. Then, given any $\alpha, \beta \in \mathbb{F}$, the unique low-degree extension of $\mathbf{r}_S^\alpha$, which is $\hat{r}_S(\alpha, Y)$, can be evaluated at β in $\mathrm{polylog}(|S|)$ field operations. In particular, this holds when the set S is an additive or multiplicative subgroup of $\mathbb{F}$.*

Proof. If $\alpha - \beta \neq 0$, we can evaluate $\hat{r}_S(\alpha, \beta)$ directly by computing $(\mathbb{Z}_S(\alpha) - \mathbb{Z}_S(\beta))/(\alpha - \beta)$ in $\mathrm{polylog}(|S|)$ field operations. If instead $\alpha - \beta = 0$ then we use a different approach: observing that the polynomial $\hat{r}_S(X, X)$ is the formal derivative of $\mathbb{Z}_S(X)$, we evaluate $\hat{r}_S(\alpha, \beta)$ by computing $\hat{r}_S(\beta, \beta) = (\mathrm{D}\mathbb{Z}_S)(\beta)$. (Note that when S is a multiplicative subgroup then $\mathrm{D}\mathbb{Z}_S(X) = |S|X^{|S|-1}$. And when S is an additive subgroup, $\mathbb{Z}_S$ is a linearized polynomial, so its derivative is the coefficient of the linear term.) □

7.2 Efficient linear independence via the tensor product

We use the bivariate polynomial introduced in Section 7.1 to find a vector of linearly independent polynomials that decomposes via the tensor product (see Definition 6.6). We begin by noting that the efficiency of evaluating low-degree extensions is preserved by tensor products.

Lemma 7.7. *Let $\Phi \colon H \to H_1 \times H_2$ be an efficient bivariate embedding (see Definition 6.3). Let $\hat{u} \in \mathbb{F}[X]$ be a degree-d_u extension of $u \in \mathbb{F}^{H_1}$, and $\hat{v} \in \mathbb{F}[X]$ a degree-d_v extension of $v \in \mathbb{F}^{H_2}$. For every $\beta \in \mathbb{F}$, if $\hat{u}, \hat{v}$ can be evaluated at β in time T_u, T_v, then an extension of $u \otimes_\Phi v \in \mathbb{F}^H$ of degree $|H_2| \cdot d_u + |H_1| \cdot d_v$ can be evaluated at β in time $O(T_u + T_v + \mathrm{polylog}\,|H|)$.*

Proof. The polynomial $\hat{u}(\Phi_1(X))\hat{v}(\Phi_2(X)) \in \mathbb{F}[X]$ agrees with the vector $u \otimes_\Phi v \in \mathbb{F}^H$ on H, has degree $|H_2| \cdot d_u + |H_1| \cdot d_v$, and can be evaluated in time $O(T_u + T_v + \mathrm{polylog}\,|H|)$. □

We define a vector of polynomials $\mathbf{t}_H^\Phi \in \mathbb{F}[X]^H$ and, for every $\alpha \in \mathbb{F}$, the vector of elements $\mathbf{t}_H^{\Phi(\alpha)} \in \mathbb{F}^H$ obtained by evaluating each polynomial. We prove that the polynomials in $\mathbf{t}_H^\Phi$ are linearly independent (Lemma 7.9), that a low-degree extension of $\mathbf{t}_H^{\Phi(\alpha)}$ can be efficiently evaluated (Corollary 7.10), and that a low-degree extension of $\mathbf{t}_H^{\Phi(\alpha)} M$ can be efficiently evaluated when M is semisuccinct (Corollary 7.11).

Definition 7.8. *Let $\Phi\colon H \to H_1 \times H_2$ be a bivariate embedding. We define $\mathbf{t}_H^{\Phi} \in \mathbb{F}[X]^H$ to be the vector of polynomials given by*

$$\mathbf{t}_H^{\Phi} \coloneqq \mathbf{r}_{H_1}^{\Phi_1} \otimes_{\Phi} \mathbf{r}_{H_2}^{\Phi_2} = \left(L'_{H_1,h_1}(\Phi_1(X))\right)_{h_1 \in H_1} \otimes_{\Phi} \left(L'_{H_2,h_2}(\Phi_2(X))\right)_{h_2 \in H_2} .$$

Moreover, for any $\alpha \in \mathbb{F}$, we define $\mathbf{t}_H^{\Phi(\alpha)} \in \mathbb{F}^H$ to be the vector of field elements given by

$$\mathbf{t}_H^{\Phi(\alpha)} \coloneqq \mathbf{r}_{H_1}^{\Phi_1(\alpha)} \otimes_{\Phi} \mathbf{r}_{H_2}^{\Phi_2(\alpha)} = \left(L'_{H_1,h_1}(\Phi_1(\alpha))\right)_{h_1 \in H_1} \otimes_{\Phi} \left(L'_{H_2,h_2}(\Phi_2(\alpha))\right)_{h_2 \in H_2} .$$

Lemma 7.9. *Let $\Phi\colon H \to H_1 \times H_2$ be a bivariate embedding. Then $\mathbf{t}_H^{\Phi} \in \mathbb{F}[X]^H$ is a vector of $|H|$ linearly independent polynomials of degree less than $2|H|$. (Note that $\mathbf{t}_H^{\Phi}$ is not a* basis *for the space of polynomials of degree less than $2|H|$ because it contains only $|H|$ elements.)*

Proof. For every $h \in H$ there exists $c \in \mathbb{F}$ such that

$$(\mathbf{r}_{H_1}^{\Phi_1} \otimes_{\Phi} \mathbf{r}_{H_2}^{\Phi_2})(h) = c \cdot L_{H_1,\Phi_1(h)}(\Phi_1(X)) \cdot L_{H_2,\Phi_2(h)}(\Phi_2(X)) .$$

This is a polynomial of degree less than $2|H_1||H_2| = 2|H|$ that is zero everywhere on H except at h. □

Corollary 7.10. *Let $\Phi\colon H \to H_1 \times H_2$ be an efficient bivariate embedding. For every $\alpha \in \mathbb{F}$, a degree-$2|H|$ extension of $\mathbf{t}_H^{\Phi(\alpha)} \in \mathbb{F}^H$ can be evaluated at any $\beta \in \mathbb{F}$ in $\mathrm{poly}(\log|H_1|, \log|H_2|)$ field operations.*

Proof. The polynomials Φ_1 and Φ_2 can be evaluated at α in $\mathrm{polylog}(|H|)$ field operations, since Φ is efficient. By Lemma 7.6, the unique low-degree extensions of $\mathbf{r}_{H_1}^{\Phi_1(\alpha)}$ and $\mathbf{r}_{H_2}^{\Phi_2(\alpha)}$ can each be evaluated at β in time $\mathrm{polylog}(|H|)$. Note that these polynomials have degrees $|H_1|$ and $|H_2|$ respectively. Thus, by Lemma 7.7, an extension of $\mathbf{t}_H^{\Phi(\alpha)} = \mathbf{r}_{H_1}^{\Phi_1(\alpha)} \otimes_{\Phi} \mathbf{r}_{H_2}^{\Phi_2(\alpha)} \in \mathbb{F}^H$ of degree $|H_2| \cdot |H_1| + |H_1| \cdot |H_2| = 2|H|$ can be evaluated at β in time $\mathrm{polylog}(|H|) = \mathrm{poly}(\log|H_1|, \log|H_2|)$. □

Corollary 7.11. *Let $\Phi\colon H \to H_1 \times H_2$ be an efficient bivariate embedding. Let M be a (Φ, κ)-semisuccinct matrix, and let $d \coloneqq |H_2| \cdot (|H_1| + \kappa(|H_1|))$. For every $\alpha \in \mathbb{F}$, a degree-d extension of $\mathbf{t}_H^{\Phi(\alpha)} M \in \mathbb{F}^H$ can be evaluated at any $\beta \in \mathbb{F}$ in time $\mathrm{poly}(\log|H_1|, |H_2|)$.*

Proof. The polynomials Φ_1 and Φ_2 can be evaluated at α in $\mathrm{polylog}(|H|)$ field operations, since Φ is efficient. We can write $M = A \otimes_{\Phi} B$ where $A \in \mathbb{F}^{H_1 \times H_1}$ is κ-succinct and $B \in \mathbb{F}^{H_2 \times H_2}$ is arbitrary, since M is (Φ, κ)-semisuccinct (see Definition 7.3). Since A is κ-succinct, we can evaluate a degree-$\kappa(|H_1|)$ extension of $\mathbf{r}_{H_1}^{\Phi_1(\alpha)} A$ at β in time $\mathrm{polylog}(|H|)$ (see Definition 7.2). We can evaluate a degree-$|H_2|$ extension of $\mathbf{r}_{H_2}^{\Phi_2(\alpha)} B$ at β in time $\mathrm{poly}(\log|H|, |H_2|)$ by direct interpolation. Thus, by Lemma 7.7, we can evaluate a degree-d extension of $\mathbf{t}_H^{\Phi(\alpha)} M = (\mathbf{r}_{H_1}^{\Phi_1(\alpha)} A) \otimes_{\Phi} (\mathbf{r}_{H_2}^{\Phi_2(\alpha)} B)$ in time $\mathrm{poly}(\log|H|, |H_2|)$. □

7.3 Proof of Lemma 7.4

We describe *succinct lincheck*, the RS oracle reduction that proves Lemma 7.4. Below we use as a subroutine the univariate sumcheck protocol from [BCRSVW19], which is an RS oracle reduction $(P_{\mathrm{SUM}}, V_{\mathrm{SUM}})$ for the relation $\mathcal{R}_{\mathrm{SUM}}$ of instance-witness pairs $(\mathtt{x}_{\mathrm{SUM}}, \mathtt{w}_{\mathrm{SUM}}) = \big((\mathbb{F}, L, H, \rho, \mu), f\big)$ such that $\mathbb{F}$ is a finite field, L is a subset of $\mathbb{F}$, H is a coset in $\mathbb{F}$, $\rho \in (0,1)$ is a rate parameter, μ is an element in $\mathbb{F}$, f is a codeword in $\mathrm{RS}[L, \rho]$, and $\sum_{a \in H} \hat{f}(a) = \mu$. In this (non-interactive) reduction, the prover sends a proof oracle π_{Σ} to the verifier, and the verifier outputs the rates $(\rho_0, \rho_1, \rho_2) = (\rho, \rho - |H|/|L|, (|H| - 1)/|L|)$ and the virtual oracles $(\mathbf{\Pi}_0, \mathbf{\Pi}_1, \mathbf{\Pi}_2) = (f, \pi_{\Sigma}, \mathbf{\Pi}_{\Sigma}[f, \pi_{\Sigma}])$ for some $\mathbf{\Pi}_{\Sigma}$ (whose exact form depends on H and μ).

Protocol 7.12. Let $\mathbb{F}$ be a finite field, and $\Phi\colon H \to H_1 \times H_2$ an efficient bivariate embedding in $\mathbb{F}$ where H is a coset in $\mathbb{F}$; set $N \coloneqq |H_1|$ and $n \coloneqq |H_2|$. Succinct lincheck is a RS oracle reduction (P, V) that works for lincheck instances $\mathbb{x} = (\mathbb{F}, L, H, \rho, M)$ for which the matrix M has the form $\sum_{i=1}^{\ell} M_i \in \mathbb{F}^{H \times H}$ and each matrix $M_i \in \mathbb{F}^{H \times H}$ is (Φ, κ)-semisuccinct.

We describe the interaction between a prover P and verifier V that both receive an input a lincheck instance $\mathbb{x}$ as above, and where the prover P additionally receives a lincheck witness (f_1, f_2) (see Definition 7.1).

First, the verifier V draws a uniformly random $\alpha \in \mathbb{F}$ and sends it to the prover P. The element α defines polynomials $\hat{p}_\alpha^{(1)}(Y)$ and $\hat{p}_\alpha^{(2)}(Y)$ in $\mathbb{F}[Y]$ known to both prover and verifier:

- $\hat{p}_\alpha^{(1)}(Y)$, a degree-$2nN$ extension of the vector $\mathbf{t}_H^{\Phi(\alpha)} \in \mathbb{F}^H$ (see Definition 7.8). By Corollary 7.10, $\hat{p}_\alpha^{(1)}(Y)$ can be evaluated anywhere in $\mathrm{poly}(\log N, \log n)$ field operations.
- $\hat{p}_\alpha^{(2)}(Y)$, a degree-$(n \cdot (N + \kappa(N)))$ extension of the vector $\mathbf{t}_H^{\Phi(\alpha)} M \in \mathbb{F}^H$. Note that

$$\mathbf{t}_H^{\Phi(\alpha)} M = \mathbf{t}_H^{\Phi(\alpha)} \cdot \left(\sum_{i=1}^{\ell} M_i \right) = \sum_{i=1}^{\ell} \mathbf{t}_H^{\Phi(\alpha)} M_i \ .$$

  Each M_i is (Φ, κ)-semisuccinct so, by Corollary 7.11, a degree-$(n \cdot (N + \kappa(N)))$ extension of $\mathbf{t}_H^{\Phi(\alpha)} M_i$ can be evaluated anywhere in $\mathrm{poly}(\log N, n)$ field operations. By linearity, $\hat{p}_\alpha^{(2)}(Y)$, which is a degree-$(n \cdot (N + \kappa(N)))$ extension of $\mathbf{t}_H^{\Phi(\alpha)} M$, can be evaluated anywhere in $\mathrm{poly}(\ell, \log N, n)$ field operations.

Moreover, the element α and the witness codewords $f_1, f_2 \in \mathrm{RS}[L, \rho]$ jointly define the polynomial $\hat{q}_\alpha(Y)$ in $\mathbb{F}[Y]$ defined as follows:

$$\hat{q}_\alpha(Y) \coloneqq \hat{p}_\alpha^{(1)}(Y) \hat{f}_1(Y) - \hat{p}_\alpha^{(2)}(Y) \hat{f}_2(Y) \ .$$

Observe that $\hat{q}_\alpha(Y)$ allegedly sums to zero on H, and has degree $\max\{2nN, n \cdot (N + \kappa(N))\}$, which is $n \cdot (N + \kappa(N))$ since $\kappa(N) \geq N$. In particular, $q_\alpha = \hat{q}_\alpha|_L$ (the restriction of the polynomial $\hat{q}_\alpha(Y)$ to the domain L) is a codeword in $\mathrm{RS}[L, \rho']$ for the rate parameter $\rho' \coloneqq \rho + \frac{n \cdot (N + \kappa(N))}{|L|}$.

Next, the prover P and verifier V assemble the sumcheck instance $\mathbb{x}_{\mathrm{SUM}} = (\mathbb{F}, L, H, \rho', 0)$, and then run the univariate sumcheck protocol (an oracle reduction) with P playing the role of $P_{\mathrm{SUM}}^{q_\alpha}(\mathbb{x}_{\mathrm{SUM}})$ and V playing the role of $V_{\mathrm{SUM}}(\mathbb{x}_{\mathrm{SUM}})$. This results in $P_{\mathrm{SUM}}^{q_\alpha}(\mathbb{x}_{\mathrm{SUM}})$ sending a proof oracle π_Σ and then $V_{\mathrm{SUM}}(\mathbb{x}_{\mathrm{SUM}})$ outputting a list of instances and corresponding virtual oracles, as required of an oracle reduction.

Since $(P_{\mathrm{SUM}}, V_{\mathrm{SUM}})$ is an RS oracle reduction, we know that the instances output by V_{SUM} are rate parameters for the relation $\mathcal{R}_{\mathrm{RS}}$ over the domain L (see Definition 5.6). In the particular case at hand, V_{SUM} outputs the rate parameters $(\rho', \rho' - |H|/|L|, (|H| - 1)/|L|)$ and corresponding virtual oracles $(q_\alpha, \pi_\Sigma, \mathbf{\Pi}_\Sigma[q_\alpha, \pi_\Sigma])$.

Finally, the verifier V outputs the rate parameters $(\rho, \rho, \rho_1, \rho_2) \coloneqq (\rho, \rho, \rho' - |H|/|L|, (|H| - 1)/|L|)$ and the virtual oracles $(\mathbf{\Pi}_{f_1}, \mathbf{\Pi}_{f_2}, \mathbf{\Pi}_1, \mathbf{\Pi}_2)$ defined as follows:

$$\mathbf{\Pi}_{f_1}[f_1, f_2, \pi_\Sigma] \coloneqq f_1 \quad \mathbf{\Pi}_{f_2}[f_1, f_2, \pi_\Sigma] \coloneqq f_2 \quad \mathbf{\Pi}_1[f_1, f_2, \pi_\Sigma] \coloneqq \pi_\Sigma \quad \mathbf{\Pi}_2[f_1, f_2, \pi_\Sigma] \coloneqq \mathbf{\Pi}_\Sigma[q_\alpha, \pi_\Sigma] \ .$$

Queries to q_α are simulated via queries to f_1 and f_2.

Note that since $(P_{\mathrm{SUM}}, V_{\mathrm{SUM}})$ is an RS oracle reduction then so is (P, V): each of the virtual oracles output by V are rational constraints, and every oracle sent by the prover appears as some virtual oracle.

Since $\mathbf{t}_H^\Phi \in \mathbb{F}[X]^H$ is a vector of linearly independent polynomials, the lincheck condition holds if and only if a certain polynomial equation in X holds:

$$\left\{ \hat{f}_1(a) = \sum_{b \in H} M_{a,b} \cdot \hat{f}_2(b) \right\}_{a \in H} \longleftrightarrow \sum_{a \in H} \mathbf{t}_H^\Phi[a] \hat{f}_1(a) \equiv \sum_{a \in H} \mathbf{t}_H^\Phi[a] \sum_{b \in H} M_{a,b} \hat{f}_2(b) \ .$$

Rearranging the right-hand side of the polynomial equation yields:

$$\sum_{a \in H} \mathbf{t}_H^\Phi[a] \cdot \sum_{b \in H} M_{a,b} \hat{f}_2(b) \equiv \sum_{b \in H} \hat{f}_2(b) \cdot \sum_{a \in H} \mathbf{t}_H^\Phi[a] M_{a,b} \equiv \sum_{b \in H} \hat{f}_2(b) \cdot (\mathbf{t}_H^\Phi M)[b] \ .$$

For any choice of $\alpha \in \mathbb{F}$, we can evaluate each side of the polynomial equation:

$$\sum_{a \in H} \hat{p}_\alpha^{(1)}(a) \hat{f}_1(a) = \left(\sum_{a \in H} \mathbf{t}_H^\Phi[a] \hat{f}_1(a) \right)(\alpha) \ ,$$
$$\sum_{b \in H} \hat{p}_\alpha^{(2)}(b) \hat{f}_2(b) = \left(\sum_{b \in H} (\mathbf{t}_H^\Phi M)[b] \hat{f}_2(b) \right)(\alpha) \ .$$

These evaluations are equal if and only if $\sum_{a \in H} \hat{q}_\alpha(a) = 0$.

Completeness. Suppose that the lincheck condition holds. This implies that f_1 is a codeword in $\mathrm{RS}[L, \rho]$, and thus so is its corresponding virtual oracle, $\mathbf{\Pi}_{f_1}[f_1, f_2, \pi_\Sigma]$. Similarly for f_2 and $\mathbf{\Pi}_{f_2}[f_1, f_2, \pi_\Sigma]$. Moreover, for *every* $\alpha \in \mathbb{F}$ it holds that $\sum_{a \in H} \hat{q}_\alpha(a) = 0$, which means that $(\mathbb{x}_{\mathrm{SUM}}, q_\alpha)$ is a valid instance-witness pair for the sumcheck relation, and so completeness of the sumcheck protocol implies that $\mathbf{\Pi}_1[f_1, f_2, \pi_\Sigma] = \pi_\Sigma$ is a codeword in $\mathrm{RS}[L, \rho_1]$, and $\mathbf{\Pi}_2[f_1, f_2, \pi_\Sigma] = \mathbf{\Pi}_\Sigma[q_\alpha, \pi_\Sigma]$ is a codeword in $\mathrm{RS}[L, \rho_2]$.

Soundness. Suppose that the lincheck condition does not hold. If either $f_1 = \mathbf{\Pi}_{f_1}[f_1, f_2, \pi_\Sigma]$ or $f_2 = \mathbf{\Pi}_{f_2}[f_1, f_2, \pi_\Sigma]$ is not a codeword in $\mathrm{RS}[L, \rho]$, then we are done. So suppose instead that f_1, f_2 are codewords in $\mathrm{RS}[L, \rho]$, which means that $q_\alpha \in \mathrm{RS}[L, \rho']$. In this case there must exist $a \in H$ such that $\hat{f}_1(a) \neq \sum_{b \in H_0} M_{a,b} \cdot \hat{f}_2(b)$. With probability at least $1 - \frac{n \cdot (N + \kappa(N))}{|\mathbb{F}|}$, it holds that $\sum_{a \in H} \hat{q}_\alpha(a) \neq 0$. By soundness of the sumcheck protocol, either $\pi_\Sigma \notin \mathrm{RS}[L, \rho_1]$ or $\mathbf{\Pi}_\Sigma[q_\alpha, \pi_\Sigma] \notin \mathrm{RS}[L, \rho_2]$. This means that either $\mathbf{\Pi}_1[f_1, f_2, \pi_\Sigma] \notin \mathrm{RS}[L, \rho_1]$ or $\mathbf{\Pi}_2[f_1, f_2, \pi_\Sigma] \notin \mathrm{RS}[L, \rho_2]$, and again we are done. (The distance in the statement of Lemma 7.4 follows via an application of Lemma 5.8.)

Efficiency. The length of the reduction is the same as that of the sumcheck protocol, which is $|L|$. The locality is one more than that of the sumcheck protocol (since a query to q_α translates to a query to each of f_1 and f_2), for a total of 3. The running time of the verifier is dominated by the cost of constructing the virtual oracles $\mathbf{\Pi}_1$ and $\mathbf{\Pi}_2$ each of which requires producing a circuit that answers a query to q_α by combining answers to queries to f_1 and f_2. This requires producing circuits for evaluating $\hat{p}_\alpha^{(1)}$ and $\hat{p}_\alpha^{(2)}$ on a point in L. Corollaries 7.10 and 7.11 give the claimed running time.

7.4 Extension to block-matrix lincheck

The lincheck relation in Definition 7.1 is a special case of a relation that we use in the proof of Lemma 8.2 (see Section 8.2), in order to obtain an oracle reduction for interactive R1CS automata. We now describe the more general relation, and then explain how the ideas discussed so far directly extend to handle it.

The lincheck relation requires checking that $v_1 = M v_2$ where $v_1, v_2 \in \mathbb{F}^H$ are encoded by Reed–Solomon codewords $f_1, f_2 \in \mathrm{RS}[L, \rho]$ respectively. Later on, we will want to check such conditions for

vectors $v_1 \coloneqq v_1^{(1)} \| \dots \| v_i^{(r)}$ and $v_2 \coloneqq v_2^{(1)} \| \dots \| v_2^{(s)}$ such that each $v_1^{(i)}, v_2^{(j)} \in \mathbb{F}^H$ is *individually* encoded by a Reed–Solomon codeword $f_1^{(i)}, f_2^{(j)}$ respectively. We decompose M into several $H \times H$ block matrices $M^{(i,j)}$, so that $v_1 = M v_2$ if and only if, for all $i \in [r]$, it holds that $v_1^{(i)} = \sum_{j=1}^{s} M^{(i,j)} v_2^{(j)}$. The block-matrix form of the lincheck relation is obtained by re-writing this condition in terms of codewords $f_1^{(i)}, f_2^{(j)}$.

Definition 7.13. *For $r, s \in \mathbb{N}$, the* **block-matrix lincheck relation** $\mathcal{R}_{\text{LIN}}^{r,s}$ *consists of instance-witness pairs*

$$(\mathbb{x}, \mathbb{w}) = \big((\mathbb{F}, L, H, \rho, \mathbf{M}), (\mathbf{f}_1, \mathbf{f}_2)\big)$$

where $\mathbb{F}$ is a finite field, L, H are subsets of $\mathbb{F}$, ρ is a rate parameter in $(0, 1)$, $\mathbf{f}_1 = (f_1^{(1)}, \dots, f_1^{(r)})$ and $\mathbf{f}_2 = (f_2^{(1)}, \dots, f_2^{(s)})$ are lists of codewords in $\mathrm{RS}[L, \rho]$, $\mathbf{M} = (M^{(i,j)})_{i \in [r], j \in [s]}$ is a block matrix with each $M_{(i,j)} \in \mathbb{F}^{H \times H}$, and for all $i \in [r]$ and $a \in H$ it holds that $\hat{f}_1^{(i)}(a) = \sum_{j=1}^{s} \sum_{b \in H} M_{a,b}^{(i,j)} \hat{f}_2^{(j)}(b)$.

The succinctness condition that we now consider on the block matrix is that *each* block $M^{(i,j)}$ is a sum of semisuccinct matrices. We can then extend Protocol 7.12, in a straightforward way, to obtain an oracle reduction for the block-matrix lincheck relation $\mathcal{R}_{\text{LIN}}^{r,s}$. Informally, the verifier's first message contains, in addition to $\alpha \in \mathbb{F}$, random elements $\beta_1, \dots, \beta_r \in \mathbb{F}$. We then consider the virtual oracle q induced by the polynomial

$$\hat{q}(Y) \coloneqq \sum_{i=1}^{r} \beta_i \left(\hat{p}_\alpha^{(1)}(Y) \hat{f}_1^{(i)}(Y) - \sum_{j=1}^{s} \hat{p}_\alpha^{(i,j)}(Y) \hat{f}_2^{(i)}(Y) \right)$$

where $\hat{p}_\alpha^{(i,j)}$ is defined like $\hat{p}_\alpha^{(2)}$ but with respect to the matrix $M^{(i,j)} \in \mathbb{F}^{H \times H}$.

The foregoing ideas allow us to extend Lemma 7.4 to the block-matrix lincheck relation, and we obtain the following lemma, which we state without proof. The verifier uses $\mathrm{poly}(|\mathbb{x}|) = \mathrm{poly}(\ell, r, s, n, \log N)$ field operations, which is *polylogarithmic* in the size of the succinct part of the matrix.

Lemma 7.14. *Let $\mathbb{F}$ be a finite field with an efficient bivariate embedding $\Phi \colon H \to H_1 \times H_2$ where H is a coset in $\mathbb{F}$. Suppose that $\mathbf{M} = (M^{(i,j)})_{i \in [r], j \in [s]}$ is a block matrix where each block has the form $M^{(i,j)} = \sum_{\iota=1}^{\ell} M_\iota^{(i,j)}$ and each $M_\iota^{(i,j)} \in \mathbb{F}^{H \times H}$ is (Φ, κ)-semisuccinct.*

Setting $N \coloneqq |H_1|$, $n \coloneqq |H_2|$, and $d \coloneqq n \cdot (N + \kappa(N))$, there is a 1-round RS oracle reduction of proximity for $\mathcal{R}_{\text{LIN}}^{r,s}$ for instances of the form $\mathbb{x} = (\mathbb{F}, L, H, \rho, \mathbf{M})$ with the following parameters:

length	$\lvert L \rvert$	*soundness error*	$(d+1)/\lvert\mathbb{F}\rvert$	*prover time*	$\lvert L \rvert \log \lvert L \rvert + \lvert\lvert \mathbf{M} \rvert\rvert$
locality	2	*distance*	$\frac{1}{2}(1 - \rho - \frac{d}{\lvert L \rvert})$	*verifier time*	$\mathrm{poly}(\lvert \mathbb{x} \rvert)$

(Above $||\mathbf{M}||$ denotes the total number of non-zero entries across all blocks of the matrix $\mathbf{M} = (M^{(i,j)})_{i \in [r], j \in [s]}$.)

8 Probabilistic checking of interactive automata

We define the *R1CS automata* relation, which considers checking algebraic computation that has no external memory, and show how to reduce it to the lincheck relation (see Definition 7.13), via an oracle reduction. Combining this reduction with results in Section 7, we obtain an oracle reduction from R1CS automata to testing proximity to the Reed–Solomon code (see Lemma 8.2 below). This oracle reduction is later used to obtain an oracle reduction for a relation about *R1CS machines* (see Section 9), which have external memory.

Informally, we define computation on R1CS automata as follows. Let $\mathbb{F}$ be a finite field, $T \in \mathbb{N}$ be a computation *time*, and $k \in \mathbb{N}$ a computation *width*. We consider execution traces $f\colon [T] \to \mathbb{F}^k$ that represent T-time computations in which each state $f(t)$ of the computation is a vector of k elements in $\mathbb{F}$. An *R1CS automaton* is specified by matrices $A, B, C \in \mathbb{F}^{k \times 2k}$ that define R1CS time constraints, and a set of boundary constraints $\mathcal{B} \subseteq [T] \times [k] \times \mathbb{F}$. An execution trace $f\colon [T] \to \mathbb{F}^k$ is accepted by the automaton if:

- f satisfies the R1CS *time constraints*, namely, for every $t \in [T-1]$, letting $f(t, t+1)$ be the concatenation of the consecutive states $f(t) \in \mathbb{F}^k$ and $f(t+1) \in \mathbb{F}^k$, it holds that $Af(t, t+1) \circ Bf(t, t+1) = Cf(t, t+1)$;
- f satisfies the *boundary constraints*, namely, for every $(t, j, \alpha) \in \mathcal{B}$ it holds that $f(t)_j = \alpha$.

Intuitively, time constraints enforce that each state in the execution trace is consistent with the prior one. Boundary constraints enforce that given locations in the execution trace have given values, for example, they could ensure that the computation started at a certain initial state and halted at a certain final state.

Interactive automata. We shall in fact consider a more general notion of computation that allows interaction with a prover, which we call *interactive R1CS automata*. This notion enables an efficient oracle reduction from *R1CS machines*, as described in Section 9.

Informally, in *interactive* R1CS automata, instead of considering execution traces $f\colon [T] \to \mathbb{F}^k$, we consider prover strategies $P_{\mathbb{w}}$ that output an execution *sub*-trace $w_i\colon [T] \to \mathbb{F}^s$ in each round of a public-coin protocol of r rounds. By juxtaposing the sub-traces $w_1, \ldots, w_r$ one obtains a trace $f\colon [T] \to \mathbb{F}^{rs}$, which is then accepted if it satisfies time constraints and boundary constraints similarly as above. Crucially, the matrices defining time constraints can depend on the verifier's randomness in the public-coin protocol.

Below we define a universal relation $\mathcal{R}^{r,\epsilon}_{\mathrm{R1A}}$ that captures computations on r-round interactive R1CS automata with soundness error ϵ. We denote the computation time by T, the length of a verifier message by ℓ, the width of a sub-trace by s, and the width of a trace by $k \coloneqq rs$. Given a trace $f\colon [T] \to \mathbb{F}^k$, we denote by $f(i, j) \in \mathbb{F}^{2k}$ the concatenation of the two states $f(i) \in \mathbb{F}^k$ and $f(j) \in \mathbb{F}^k$.

Definition 8.1. *The promise relation $\mathcal{R}^{r,\epsilon}_{\mathrm{R1A}} = (\mathcal{R}_{\mathsf{Yes}}, \mathcal{L}_{\mathsf{No}})$ of bounded accepting computation problems on* **interactive R1CS automata** *is defined as follows. An instance $\mathbb{x} = (\mathbb{F}, \mathbf{A}, \mathbf{B}, \mathbf{C}, \mathcal{B}, \ell)$ consists of a finite field $\mathbb{F}$, functions $\mathbf{A}, \mathbf{B}, \mathbf{C}\colon \mathbb{F}^{r\ell} \to \mathbb{F}^{s \times 2rs}$ defining time constraints, and boundary constraints $\mathcal{B} \subseteq [T] \times [rs] \times \mathbb{F}$. A witness $P_{\mathbb{w}}$ is a prover strategy for the following game defined by $\mathbb{x}$. Let $V_{\mathbb{x}}$ be the interactive Turing machine that interacts with $P_{\mathbb{w}}$ for r rounds, where in the i-th round $V_{\mathbb{x}}$ sends a random $a_i \in \mathbb{F}^\ell$, and $P_{\mathbb{w}}$ replies with a message $w_i\colon [T] \to \mathbb{F}^s$. At the end of the interaction, letting $a \coloneqq (a_1, \ldots, a_r) \in \mathbb{F}^{r\ell}$ and $f\colon [T] \to \mathbb{F}^{rs}$ be defined as $f(t) \coloneqq w_1(t) \| \ldots \| w_r(t)$, $V_{\mathbb{x}}$ accepts if the following holds.*

- Time constraints: $\forall\, t \in [T-1],\ \ \mathbf{A}(a)f(t, t+1) \circ \mathbf{B}(a)f(t, t+1) = \mathbf{C}(a)f(t, t+1)\,.$
- Boundary constraints: $\forall\, (t, j, \alpha) \in \mathcal{B},\ \ f(t)_j = \alpha\,.$

A pair $(\mathbb{x}, P_{\mathbb{w}})$ is in $\mathcal{R}_{\mathsf{Yes}}$ if $(P_{\mathbb{w}}, V_{\mathbb{x}})$ accepts with probability 1. On the other hand, an instance $\mathbb{x}$ is in $\mathcal{L}_{\mathsf{No}}$ if for every prover strategy $\tilde{P}$ it holds that $(\tilde{P}, V_{\mathbb{x}})$ accepts with probability at most ϵ.

In this section we obtain an oracle reduction, with linear length and locality $r+1$, from interactive R1CS automata to testing proximity to the Reed–Solomon code. The verifier uses $\mathrm{poly}(|\mathbb{x}|) = \mathrm{poly}(r, s, |\mathcal{B}|, \log T)$ field operations, which is *exponentially* faster than the computation time T, since the dependence on the T is polylogarithmic instead of polynomial.

Lemma 8.2. *Let $\mathbb{F}$ be a finite field, H a coset in $\mathbb{F}$, and L a subset of $\mathbb{F}$ with $L \cap H = \emptyset$. Let $\mathcal{T} \colon [T] \times [s] \to H$ be a trace embedding in $\mathbb{F}$. There is an RS oracle reduction over domain L (Protocol 8.7) for instances $\mathbb{x}$ of $\mathcal{R}^{r,\epsilon}_{\mathrm{R1A}}$ over $\mathbb{F}$ and of computation time T and width $k = rs$. The reduction has $r+1$ rounds and the following parameters:*

length	$(r+4)\lvert L\rvert$	*soundness error*	$\epsilon + (3sT+1)/\lvert\mathbb{F}\rvert$	*prover time*	$\lvert L\rvert \cdot (\log \lvert L\rvert + \lvert\mathbb{x}\rvert)$
locality	$r+1$	*distance*	$\frac{1}{2}(1 - 4sT/\lvert L\rvert)$	*verifier time*	$\mathrm{poly}(\lvert\mathbb{x}\rvert)$

The rest of this section is dedicated to proving Lemma 8.2. Our high-level approach is to first identify a family of *semisuccinct* matrices (see Definition 7.3) that can express computation via R1CS automata. We call this family *staircase matrices* and, in Section 8.1, prove that they are indeed semisuccinct. Then, in Section 8.2 we prove Lemma 8.2 by invoking the succinct lincheck protocol in Section 7 (which requires semisuccinct matrices) on carefully chosen staircase matrices, derived from an interactive automaton.

8.1 Staircase matrices

We introduce the notion of *staircase matrices* and prove that they are the sum of two semisuccinct matrices. This requires establishing simple algebraic properties of the identity matrix and a related matrix.

First recall from Definition 7.2 that a matrix $A \in \mathbb{F}^{S \times S}$ is *κ-succinct* if, for every $d \geq |S| - 1$ and $v \in \mathbb{F}^S$, a degree-$\kappa(d)$ extension of vA can be evaluated anywhere in $\mathbb{F}$ in time $\mathrm{polylog}(|S|)$ whenever a degree-d extension of v can be evaluated anywhere in $\mathbb{F}$ in time $\mathrm{polylog}(|S|)$. The *identity matrix* on a subset S of $\mathbb{F}$ (the matrix $I \in \mathbb{F}^{S \times S}$ with $I(\alpha, \alpha) = 1$ for all $\alpha \in S$ and 0 elsewhere) is trivially κ-succinct for $\kappa(d) \coloneqq d$: if a degree-d extension of v can be evaluated anywhere in $\mathbb{F}$ in time T, then so can a degree-d extension of vI.

We now define the *shifted identity matrix*, for a given successor ordering, and prove that it is κ-succinct, where κ depends on algebraic properties of the successor ordering on the subset S that we consider. Recall from Definition 6.13 that a *successor ordering* on $S \subseteq \mathbb{F}$ is a pair $\mathcal{O} = (\mathbf{1}_S, N)$ where $\mathbf{1}_S \in S$ and N is a piecewise polynomial on S of degree 1 such that $S = \{\alpha_1, \ldots, \alpha_{|S|}\}$, where $\alpha_1 \coloneqq \mathbf{1}_S$ and $\alpha_{i+1} \coloneqq N(\alpha_i)$ inductively for every $i \in \{1, \ldots, |S| - 1\}$.

Definition 8.3. *Let $\mathbb{F}$ be a field, S a subset of $\mathbb{F}$, and $\mathcal{O} = (\mathbf{1}_S, N)$ a successor ordering on S. The* **shifted identity matrix** *$I^{\rightarrow}_{\mathcal{O}} \in \mathbb{F}^{S \times S}$ has $I^{\rightarrow}_{\mathcal{O}}(\alpha, N(\alpha)) = 1$ for all $\alpha \in S$ such that $\gamma_{\mathcal{O}}(\alpha) < |S|$, and 0 elsewhere. Under the ordering $\gamma_{\mathcal{O}}$, we can view $I^{\rightarrow}_{\mathcal{O}}$ as the following matrix:*

$$I^{\rightarrow}_{\mathcal{O}} = \begin{pmatrix} 0 & 1 & & \\ & 0 & \ddots & \\ & & \ddots & 1 \\ & & & 0 \end{pmatrix} .$$

Lemma 8.4. *Let S be a subset of $\mathbb{F}$ whose vanishing polynomial can be computed in $\mathrm{polylog}(|S|)$ field operations and $\mathcal{O} = (\mathbf{1}_S, N)$ be an efficient successor ordering on S (see Definition 6.13). Then $I^{\rightarrow}_{\mathcal{O}}$ is κ-succinct for $\kappa(d) \coloneqq |S| + d$.*

Proof. Let $r \in \mathbb{F}^S$ and $d \geq |S|$ be such that a degree-d extension $\hat{r} \in \mathbb{F}[X]$ of r can be evaluated in T field operations. We prove that a degree-$(|S|+d)$ extension $\hat{r}^{\rightarrow}_{\mathcal{O}} \in \mathbb{F}[X]$ of $r^{\rightarrow}_{\mathcal{O}} \coloneqq rI^{\rightarrow}_{\mathcal{O}} \in \mathbb{F}^S$ can be evaluated in $O(T + \mathrm{polylog}(|S|))$ field operations.

Observe that $r^{\rightarrow}_{\mathcal{O}} = (0, r_1, \ldots, r_{|S|-1})$ is the right shift of r, so that, for every $\alpha \in S$, we have

$$r^{\rightarrow}_{\mathcal{O}}(\alpha) = r(N(\alpha)) - r(N(\mathbf{1}_S)) \cdot \mathbb{I}[\alpha = \mathbf{1}_S]$$

where $\mathbb{I}[\alpha = \mathbf{1}_S]$ is the indicator function for $\mathbf{1}_S$ on S. Let $(\mathcal{S}, \mathcal{F}) = \big((S_1, \ldots, S_\ell), (f_1, \ldots, f_\ell)\big)$ be a piecewise polynomial of piecewise degree 1 that computes N (see Definition 6.11), and let s_i be the unique extension of degree less than $|S|$ of the indicator for S_i in S. We have that

$$r(N(\alpha)) = r\left(\sum_{i=1}^{\ell} s_i(\alpha) f_i(\alpha)\right) = \sum_{i=1}^{\ell} s_i(\alpha) r(f_i(\alpha)) \ .$$

We deduce that

$$\hat{r}^{\rightarrow}_{\mathcal{O}}(X) \coloneqq \left(\sum_{i=1}^{\ell} s_i(X) \cdot \hat{r}(f_i(X))\right) - \left(\sum_{i=1}^{\ell} s_i(\mathbf{1}_S) \cdot \hat{r}(f_i(\mathbf{1}_S))\right) \cdot L_{S,\mathbf{1}_S}(X)$$

is a degree-$(|S|+d)$ extension of $r^{\rightarrow}_{\mathcal{O}}$. Note that $L_{S,\mathbf{1}_S}$ can be evaluated in $\mathrm{polylog}(|S|)$ operations, since the vanishing polynomial of S can be evaluated in $\mathrm{polylog}(|S|)$ operations. (See Section 4.1). Hence, taking also into account that N is efficient, we conclude that $\hat{r}^{\rightarrow}_{\mathcal{O}}$ can be evaluated in $O(T + \mathrm{polylog}(|S|))$ operations. □

The staircase matrix of two matrices M and M' is the block matrix whose diagonal consists of blocks of M and its superdiagonal consists of blocks of M'. Algebraically, we capture this by considering the matrix that consists of the sum of two terms: (1) the tensor product of the identity matrix with M; and (2) the tensor product of the shifted identity matrix with M'. We formally define this notion, and then use Lemma 8.4 to deduce that each of these two terms is semisuccinct. Recall from Definition 7.3 that a matrix $M \in \mathbb{F}^{H \times H}$ is (Φ, κ)-*semisuccinct*, where $\Phi \colon H \to H_1 \times H_2$ is a bivariate embedding and $\kappa \colon \mathbb{N} \to \mathbb{N}$, if M can be written as $A \otimes_\Phi B$ for κ-succinct $A \in \mathbb{F}^{H_1 \times H_1}$ and arbitrary $B \in \mathbb{F}^{H_2 \times H_2}$.

Definition 8.5. *Let $\mathbb{F}$ be a finite field, $\Phi \colon H \to H_1 \times H_2$ a bivariate embedding in $\mathbb{F}$, and $\mathcal{O}$ a successor ordering on H_1. The* **staircase matrix** *of two matrices $M, M' \in \mathbb{F}^{H_2 \times H_2}$ is the matrix in $\mathbb{F}^{H \times H}$ defined as*

$$S_{\Phi,\mathcal{O}}(M, M') \coloneqq I \otimes_\Phi M + I^{\rightarrow}_{\mathcal{O}} \otimes_\Phi M'$$

where I and $I^{\rightarrow}_{\mathcal{O}}$ are the identity and shifted identity matrices in $\mathbb{F}^{H_1 \times H_1}$, respectively. Under the appropriate ordering on H, we can write $S_{\Phi,\mathcal{O}}(M, M')$ as the block matrix:

$$S_{\Phi,\mathcal{O}}(M, M') = \begin{pmatrix} M & M' & & & & \\ & M & M' & & & \\ & & M & M' & & \\ & & & \ddots & \ddots & \\ & & & & M & M' \\ & & & & & M \end{pmatrix} .$$

Corollary 8.6. *Let $\mathbb{F}$ be a finite field, $\Phi \colon H \to H_1 \times H_2$ an efficient bivariate embedding in $\mathbb{F}$, and $\mathcal{O} = (\mathbf{1}_{H_1}, N)$ an efficient successor ordering on H_1. Then each of the terms in a staircase matrix $S_{\Phi,\mathcal{O}}(M, M') = I \otimes_\Phi M + I^{\rightarrow}_{\mathcal{O}} \otimes_\Phi M'$ is (Φ, κ)-semisuccinct for $\kappa(d) \coloneqq |H_1| + d$.*

8.2 Proof of Lemma 8.2

We show an oracle reduction, with linear length and locality $r+1$, from the interactive R1CS automata relation $\mathcal{R}^{r,\epsilon}_{\mathrm{R1A}}$ to testing proximity to the Reed–Solomon code. Let $\mathbb{x} = (\mathbb{F}, \mathbf{A}, \mathbf{B}, \mathbf{C}, \mathcal{B}, \ell)$ be an instance of the relation $\mathcal{R}^{r,\epsilon}_{\mathrm{R1A}}$, and let $P_{\mathbb{w}}$ be a candidate witness for $\mathbb{x}$. Recall that the witness $P_{\mathbb{w}}$ is a prover strategy for the r-round public-coin game defined by the verifier $V_{\mathbb{x}}$ determined by $\mathbb{x}$.

In round i of this game, $V_{\mathbb{x}}$ sends a uniformly random $a_i \in \mathbb{F}^{\ell}$, and $P_{\mathbb{w}}$ replies with a message $w_i \colon [T] \to \mathbb{F}^s$. Letting $a \coloneqq (a_1, \ldots, a_r) \in \mathbb{F}^{r\ell}$ and $f \colon [T] \to \mathbb{F}^{rs}$ be defined as $f(t) \coloneqq w_1(t) \| \ldots \| w_r(t)$, to check membership in the relation $\mathcal{R}^{r,\epsilon}_{\mathrm{R1A}}$ it suffices to verify the following.

(i) Time constraints: $\forall\, t \in [T-1]\,,\ \mathbf{A}(a)f(t,t+1) \circ \mathbf{B}(a)f(t,t+1) = \mathbf{C}(a)f(t,t+1)$.
(ii) Boundary constraints: $\forall\, (t,j,\alpha) \in \mathcal{B}\,,\ f(t)_j = \alpha$.

Our task now is to specify the prover P and the verifier V of a suitable Reed–Solomon oracle reduction. Informally, the prover P and verifier V engage in an "encoded" version of the game defined by $\mathbb{x}$. Then, the verifier uses the prover's encoded messages to reduce checking the time constraints and the boundary constraints to membership in the Reed–Solomon code.

The role of *staircase matrices* (defined in Section 8.1) for these checks can be explained as follows. Let $\mathbf{A}(a)_1$ be the first rs columns of $\mathbf{A}(a)$, and $\mathbf{A}(a)_2$ the remaining rs columns; similarly for $\mathbf{B}(a)$ and $\mathbf{C}(a)$. Viewing f as a vector $f(1)\| \cdots \| f(T) \in \mathbb{F}^{kT}$, we left-multiply the vector f by staircase matrices, obtaining:

$$
\begin{aligned}
f_{\mathbf{A}} &\coloneqq S(\mathbf{A}(a)_1, \mathbf{A}(a)_2) \cdot f = \mathbf{A}(a)f(1,2)\| \cdots \| \mathbf{A}(a)f(T-1,T) \| \mathbf{A}(a)_1 f(T) \in \mathbb{F}^{sT}\ , \\
f_{\mathbf{B}} &\coloneqq S(\mathbf{B}(a)_1, \mathbf{B}(a)_2) \cdot f = \mathbf{B}(a)f(1,2)\| \cdots \| \mathbf{B}(a)f(T-1,T) \| \mathbf{B}(a)_1 f(T) \in \mathbb{F}^{sT}\ , \\
f_{\mathbf{C}} &\coloneqq S(\mathbf{C}(a)_1, \mathbf{C}(a)_2) \cdot f = \mathbf{C}(a)f(1,2)\| \cdots \| \mathbf{C}(a)f(T-1,T) \| \mathbf{C}(a)_1 f(T) \in \mathbb{F}^{sT}\ .
\end{aligned}
$$

(For simplicity, we suppress the bivariate embedding and successor ordering used to define a staircase matrix.)

The time constraints are equivalent to checking that $f_{\mathbf{A}}, f_{\mathbf{B}}, f_{\mathbf{C}}$ are consistent with f, which we can do via our new succinct lincheck protocol from Section 7, and also checking that $f_{\mathbf{A}} \circ f_{\mathbf{B}}$ and $f_{\mathbf{C}}$ agree on their first $s \cdot (T-1)$ entries, which we can do via other (standard) probabilistic checking tools.

We now provide a formal description of the reduction and then discuss its properties.

Protocol 8.7. Let $\mathbb{F}$ be a finite field, H a coset in $\mathbb{F}$, and L a subset of $\mathbb{F}$ with $L \cap H = \emptyset$. Let $\mathcal{T} = (\Phi \colon H \to H_1 \times H_2, \mathcal{O}, \gamma)$ be a trace embedding in $\mathbb{F}$ with $T = |H_1|$ and $s = |H_2|$ (see Definition 6.1). We show a Reed–Solomon oracle reduction over domain L, which works on instances $\mathbb{x} = (\mathbb{F}, \mathbf{A}, \mathbf{B}, \mathbf{C}, \mathcal{B}, \ell)$ for the relation $\mathcal{R}^{r,\epsilon}_{\mathrm{R1A}}$ (see Definition 8.1) that have computation time T and width $k = rs$. The oracle reduction is specified by the prover P and verifier V described below. Recall that P and V receive the instance $\mathbb{x}$ as input, while P additionally receives a witness $P_{\mathbb{w}}$ for $\mathbb{x}$.

1. **Interaction.** The prover P and verifier V engage in an "encoded" version of the r-round game induced by $\mathbb{x}$. In round i, first V behaves exactly as $V_{\mathbb{x}}$ by sending random elements $a_i \in \mathbb{F}^{\ell}$; then P obtains a message $w_i \colon [T] \to \mathbb{F}^s$ from $P_{\mathbb{w}}$ and, instead of sending w_i, sends its encoding $\pi_{w_i} \coloneqq \hat{\pi}_{w_i}|_L$, where $\hat{\pi}_{w_i}$ is the unique polynomial of degree less than sT such that

$$
\forall\, t \in [T]\,,\ \forall\, j \in [s] \quad \hat{\pi}_{w_i}\big(\mathcal{T}(t,j)\big) = (w_i(t))[j]\ .
$$

   For each $i \in [r]$, the verifier V outputs the rate parameter $\rho_{w_i} \coloneqq sT/|L|$ and virtual oracle $\mathbf{\Pi}_{w_i} \coloneqq \pi_{w_i}$.

2. **Proof oracles.** The prover P uses the verifier randomness $a = (a_1, \ldots, a_r) \in \mathbb{F}^{r\ell}$ to compute the matrices $\mathbf{A}(a), \mathbf{B}(a), \mathbf{C}(a) \in \mathbb{F}^{s \times 2rs}$. We view these as $2r$ block matrices with blocks of size $s \times s$:

$$\begin{aligned}
\mathbf{A}(a) &= \begin{pmatrix} A_1^{(1)} & \cdots & A_1^{(r)} & A_2^{(1)} & \cdots & A_2^{(r)} \end{pmatrix} \ , \\
\mathbf{B}(a) &= \begin{pmatrix} B_1^{(1)} & \cdots & B_1^{(r)} & B_2^{(1)} & \cdots & B_2^{(r)} \end{pmatrix} \ , \\
\mathbf{C}(a) &= \begin{pmatrix} C_1^{(1)} & \cdots & C_1^{(r)} & C_2^{(1)} & \cdots & C_2^{(r)} \end{pmatrix} \ .
\end{aligned}$$

Next, P computes the unique polynomials $\hat{\pi}_{\mathbf{A}}, \hat{\pi}_{\mathbf{B}}, \hat{\pi}_{\mathbf{C}}$ of degree less than sT such that

$$\begin{aligned}
\forall\, h \in H \qquad \hat{\pi}_{\mathbf{A}}(h) &= \Big(\sum_{i=1}^{r} S_{\Phi,\mathcal{O}}(A_1^{(i)}, A_2^{(i)}) \cdot \hat{\pi}_{w_i}|_H \Big)[h] \ , \\
\forall\, h \in H \qquad \hat{\pi}_{\mathbf{B}}(h) &= \Big(\sum_{i=1}^{r} S_{\Phi,\mathcal{O}}(B_1^{(i)}, B_2^{(i)}) \cdot \hat{\pi}_{w_i}|_H \Big)[h] \ , \\
\forall\, h \in H \qquad \hat{\pi}_{\mathbf{C}}(h) &= \Big(\sum_{i=1}^{r} S_{\Phi,\mathcal{O}}(C_1^{(i)}, C_2^{(i)}) \cdot \hat{\pi}_{w_i}|_H \Big)[h] \ .
\end{aligned}$$

Finally, P sends to V the codewords in $\mathrm{RS}[L, sT/|L|]$ obtained by restricting the above polynomials to L:

$$\pi_{\mathbf{A}} \coloneqq \hat{\pi}_{\mathbf{A}}|_L \quad \pi_{\mathbf{B}} \coloneqq \hat{\pi}_{\mathbf{B}}|_L \quad \pi_{\mathbf{C}} \coloneqq \hat{\pi}_{\mathbf{C}}|_L \ .$$

The verifier V outputs rate parameters $(\rho_{\mathbf{A}}, \rho_{\mathbf{B}}, \rho_{\mathbf{C}})$ and virtual oracles $(\mathbf{\Pi}_{\mathbf{A}}, \mathbf{\Pi}_{\mathbf{B}}, \mathbf{\Pi}_{\mathbf{C}})$ defined as:

$$\begin{aligned}
\rho_{\mathbf{A}} &\coloneqq sT/|L| \qquad \mathbf{\Pi}_{\mathbf{A}} \coloneqq \pi_{\mathbf{A}} \ , \\
\rho_{\mathbf{B}} &\coloneqq sT/|L| \qquad \mathbf{\Pi}_{\mathbf{B}} \coloneqq \pi_{\mathbf{B}} \ , \\
\rho_{\mathbf{C}} &\coloneqq sT/|L| \qquad \mathbf{\Pi}_{\mathbf{C}} \coloneqq \pi_{\mathbf{B}} \ .
\end{aligned}$$

3. **Succinct lincheck.** The prover P and verifier V invoke the block-matrix succinct lincheck of Lemma 7.14 on the instance $\mathbb{x}_{\mathrm{LIN}} \coloneqq (\mathbb{F}, L, H, \rho, \mathbf{M})$ and witness $\mathbb{w}_{\mathrm{LIN}} \coloneqq ((\pi_{\mathbf{A}}, \pi_{\mathbf{B}}, \pi_{\mathbf{C}}), (\pi_{w_1}, \ldots, \pi_{w_r}))$, where

$$\mathbf{M} \coloneqq \begin{pmatrix} S_{\Phi,\mathcal{O}}(A_1^{(1)}, A_2^{(1)}) & \cdots & S_{\Phi,\mathcal{O}}(A_1^{(r)}, A_2^{(r)}) \\ S_{\Phi,\mathcal{O}}(B_1^{(1)}, B_2^{(1)}) & \cdots & S_{\Phi,\mathcal{O}}(B_1^{(r)}, B_2^{(r)}) \\ S_{\Phi,\mathcal{O}}(C_1^{(1)}, C_2^{(1)}) & \cdots & S_{\Phi,\mathcal{O}}(C_1^{(r)}, C_2^{(r)}) \end{pmatrix} \ .$$

The verifier V outputs the rate parameters and virtual oracles output by the verifier of this RS oracle reduction. The maximum rate across these is $\rho + d/|L|$, where $\rho \coloneqq sT/|L|$ and $d \coloneqq s \cdot (T + \kappa(T)) = 3sT$, since $\kappa(T) = |H_1| + T = 2T$ by Corollary 8.6.

4. **Rowcheck.** Define the set $H_{\mathrm{ROW}} \coloneqq H \setminus \{\mathcal{T}(T,1), \ldots, \mathcal{T}(T,s)\}$. The verifier V outputs the rate parameter ρ_{ROW} and virtual oracle $\mathbf{\Pi}_{\mathrm{ROW}}$ defined as:

$$\rho_{\mathrm{ROW}} \coloneqq \frac{2sT - |H_{\mathrm{ROW}}|}{|L|} \quad \text{and} \quad \mathbf{\Pi}_{\mathrm{ROW}}(\alpha) \coloneqq \frac{\pi_{\mathbf{A}}(\alpha) \cdot \pi_{\mathbf{B}}(\alpha) - \pi_{\mathbf{C}}(\alpha)}{\mathbb{Z}_{H_{\mathrm{ROW}}}(\alpha)} \ ,$$

where $\mathbb{Z}_{H_{\mathrm{ROW}}}(X) := \mathbb{Z}_{H_1}(\Phi_1(X))/\mathbb{Z}_{\{\mathcal{T}(T,1),\ldots,\mathcal{T}(T,s)\}}(X)$ is (a multiple of) the vanishing polynomial of H_{ROW} because $\mathbb{Z}_{H_1}(\Phi_1(X))$ is (a multiple of) the vanishing polynomial of H. Observe that $\mathbb{Z}_{H_{\mathrm{ROW}}}$ can be evaluated in $\mathrm{poly}(\log T, s)$ field operations because: (1) the definition of a trace embedding (Definition 6.1) requires that $\mathbb{Z}_{H_1}$ and Φ_1 can each be evaluated in $\mathrm{polylog}(|H|) = \mathrm{polylog}(Ts)$ field operations; (2) the denominator can be evaluated in $\mathrm{poly}(s)$ field operations. (We also know that H is a coset in $\mathbb{F}$, a condition inherited from univariate sumcheck, which means that the vanishing polynomial of H can directly be evaluated in $\mathrm{polylog}(|H|)$ field operations. The above reasoning assumes less.)

The above is an RS oracle reduction of proximity over domain L for the *rowcheck relation* $\mathcal{R}_{\mathrm{ROW}}$, which consists of instance-witness pairs $(\mathbb{x}_{\mathrm{ROW}}, \mathbb{w}_{\mathrm{ROW}})$, where $\mathbb{x}_{\mathrm{ROW}} = (\mathbb{F}, L, H_{\mathrm{ROW}}, \rho_{\mathrm{ROW}})$ and $\mathbb{w}_{\mathrm{ROW}} = (\pi_1, \pi_2, \pi_3)$, such that $\pi_1, \pi_2, \pi_3 \in \mathrm{RS}[L, \rho]$ and, for every $\alpha \in H_{\mathrm{ROW}}$, $\hat{\pi}_1(\alpha) \cdot \hat{\pi}_2(\alpha) - \hat{\pi}_3(\alpha) = 0$.

5. **Enforce boundary constraints.** The verifier V partitions the boundary constraints $\mathcal{B}$ into $(\mathcal{B}_1, \ldots, \mathcal{B}_r)$ so that the constraints in $\mathcal{B}_i$ apply to the message $w_i \colon [T] \to \mathbb{F}^s$. Namely, for each $i \in [r]$, V defines

$$\mathcal{B}_i := \left\{ (t, j', \alpha) \;\middle|\; \exists\, j' \in \{1, \ldots, s\} \text{ s.t. } (t, j' + s \cdot (i-1), \alpha) \in \mathcal{B} \right\} .$$

Let $E_i := \{\mathcal{T}(t, j') : (t, j', \alpha) \in \mathcal{B}_i\}$ be the set of locations in H contained in $\mathcal{B}_i$. Let B_i be the polynomial of degree less than $|E_i|$ such that $B_i(\mathcal{T}(t, j')) = \alpha$ for every $(t, j', \alpha) \in \mathcal{B}_i$. The verifier outputs rate parameters $(\rho_{\mathcal{B}_1}, \ldots, \rho_{\mathcal{B}_r})$ and virtual oracles $(\mathbf{\Pi}_{\mathcal{B}_1}, \ldots, \mathbf{\Pi}_{\mathcal{B}_r})$ where each rate and virtual oracle is defined as follows:

$$\rho_{\mathcal{B}_i} := \frac{sT - |E_i|}{|L|} \quad \text{and} \quad \mathbf{\Pi}_{\mathcal{B}_i}(\alpha) := \frac{\pi_{w_i}(\alpha) - B_i(\alpha)}{\mathbb{Z}_{E_i}(\alpha)} .$$

We conclude the proof of Lemma 8.2 by showing its completeness, soundness, and efficiency.

Completeness. Suppose that $(\mathbb{x}, P_{\mathbb{w}}) \in \mathcal{R}_{\mathsf{Yes}}$, and consider the honest prover strategy P described above. We argue that every pair $(\rho, \mathbf{\Pi})$ output by the verifier V belongs to the Reed–Solomon relation $\mathcal{R}_{\mathrm{RS}}$ (see Definition 5.6). We separately consider each step in the reduction. In Item 1, for every $i \in [r]$, the virtual oracle $\mathbf{\Pi}_{w_i} = \pi_{w_i}$ indeed has rate $\rho_{w_i} = sT/|L|$. In Item 2, the virtual oracles $(\mathbf{\Pi}_{\mathbf{A}}, \mathbf{\Pi}_{\mathbf{B}}, \mathbf{\Pi}_{\mathbf{C}})$ indeed have rates $(\rho_{\mathbf{A}}, \rho_{\mathbf{B}}, \rho_{\mathbf{C}}) = (sT/|L|, sT/|L|, sT/|L|)$. In Item 3, the constructed instance-witness pair $(\mathbb{x}_{\mathrm{LIN}}, \mathbb{w}_{\mathrm{LIN}})$ satisfies the lincheck relation, and thus we rely on the completeness of the succinct lincheck protocol. In Item 4 and in Item 5, the constructed polynomials in the numerator are divisible by the denominator if and only if the rowcheck condition and boundary conditions hold respectively.

Soundness. Suppose that $\mathbb{x} \in \mathcal{L}_{\mathsf{No}}$. Suppose first that the oracles sent by the prover in Item 1 and Item 2 belong to the prescribed code:

$$\tilde{\pi}_{w_1} \in \mathrm{RS}[L, \rho_1], \ldots, \tilde{\pi}_{w_r} \in \mathrm{RS}[L, \rho_r] \;,\; \tilde{\pi}_{\mathbf{A}} \in \mathrm{RS}[L, \rho_{\mathbf{A}}] \;,\; \tilde{\pi}_{\mathbf{B}} \in \mathrm{RS}[L, \rho_{\mathbf{B}}] \;,\; \tilde{\pi}_{\mathbf{C}} \in \mathrm{RS}[L, \rho_{\mathbf{C}}] \;.$$

Indeed, if any of the above conditions does not hold, then the weak soundness condition is immediately fulfilled by the violating oracle (see Lemma 5.8). Note that all the rates above equal $sT/|L|$.

Let $\tilde{f} \colon [T] \to \mathbb{F}^{rs}$ be the computation trace induced by the oracles sent by the prover, that is, $\tilde{f}(t) := \tilde{w}_1(t) \| \ldots \| \tilde{w}_r(t)$ for every $t \in [T]$, where each $\tilde{w}_i$ is the sub-trace encoded in $\tilde{\pi}_{w_i}$. By Definition 8.1, we know that, with probability at least $1 - \epsilon$ over the verifier's randomness $a = (a_1, \ldots, a_r) \in \mathbb{F}^{r\ell}$, either $\tilde{f}$ does not satisfy some time constraint or some boundary constraint. We analyze each of these two cases.

- *A time constraint is violated*, i.e., there exists t such that $\mathbf{A}(a)\tilde{f}(t, t+1) \circ \mathbf{B}(a)\tilde{f}(t, t+1) \neq \mathbf{C}(a)\tilde{f}(t, t+1)$.

If the rowcheck condition is violated, then the interpolations $\hat{\pi}_{\mathbf{A}}, \hat{\pi}_{\mathbf{B}}, \hat{\pi}_{\mathbf{C}}$ of $\tilde{\pi}_{\mathbf{A}}, \tilde{\pi}_{\mathbf{B}}, \tilde{\pi}_{\mathbf{C}}$ are such that there exists $\alpha \in H_{\mathrm{ROW}}$ for which $\hat{\pi}_{\mathbf{A}}(\alpha) \cdot \hat{\pi}_{\mathbf{B}}(\alpha) - \hat{\pi}_{\mathbf{C}}(\alpha) \neq 0$, which means that $\hat{\pi}_{\mathbf{A}}\hat{\pi}_{\mathbf{B}} - \hat{\pi}_{\mathbf{C}}$ is not divisible by $\mathbb{Z}_{H_{\mathrm{ROW}}}$, and thus it is not the case that $\mathbf{\Pi}_{\mathrm{ROW}}[\tilde{\pi}_{\mathbf{A}}, \tilde{\pi}_{\mathbf{B}}, \tilde{\pi}_{\mathbf{C}}] \in \mathrm{RS}\left[L, \rho_{\mathrm{ROW}}\right]$.

Otherwise, it must be the case that the lincheck condition is violated, which means that with probability at least $1 - \frac{3sT+1}{\mathbb{F}}$, one of the instances output by the lincheck verifier is not the prescribed code.

- *A boundary constraint is violated*, i.e., there exists $(t, j, \alpha) \in \mathcal{B}$ such that $\tilde{f}(t)_j \neq \alpha$.

  Let $i \in [r]$ and $j' \in \{1, \ldots, s\}$ be such that $j = j' + s \cdot (i-1)$, which means that $(t, j', \alpha) \in \mathcal{B}_i$ (see Item 5); note that $\tilde{f}(t)_j = \tilde{w}_i(t)_{j'}$. Suppose that $\mathbf{\Pi}_{\mathcal{B}_i}[\tilde{\pi}_{w_i}] \in \mathrm{RS}\left[L, \rho_{\mathcal{B}_i}\right]$ (otherwise we are done). Then, by definition of $\mathbf{\Pi}_{\mathcal{B}_i}[\tilde{\pi}_{w_i}]$, the interpolation of $\mathbf{\Pi}_{\mathcal{B}_i}[\tilde{\pi}_{w_i}]$ times the polynomial $\mathbb{Z}_{E_i}$ equals $\hat{\pi}_{w_i} - B_i$, where $\hat{\pi}_{w_i}$ is the interpolation of $\tilde{\pi}_{w_i}$. Using the fact that $\mathbb{Z}_{E_i}$ vanishes at $\mathcal{T}(t, j')$, we conclude that $\tilde{f}(t)_j = \tilde{w}_i(t)_{j'} = \hat{\pi}_{w_i}(\mathcal{T}(t, j')) = B_i(\mathcal{T}(t, j')) = \alpha$, which is a contradiction.

Efficiency. The oracle reduction in Protocol 8.7 adds a single round of interaction to the r-round interactive automaton at hand, and thus the round complexity is $r + 1$. The length of the reduction is determined by the $r + 1$ oracles that are sent in the interaction phase and the 3 oracles sent in the lincheck protocol; each oracle is of length $|L|$, and thus the total length is $(r + 4)|L|$. In each round there is one probe to the virtual oracles, and hence the locality is $r + 1$. The distance follows by invoking Lemma 5.8 with respect to the maximum rate of $4sT/|L|$ for the lincheck reduction, yielding distance $\frac{1}{2}(1 - 4sT/|L|)$. Finally, the prover and verifier time complexity follows immediately from the time complexity of the lincheck and rowcheck protocols.

9 Reducing machines to interactive automata

We define the *R1CS machines* relation, which is about checking algebraic computation with external memory. We then show how to reduce it, via an oracle reduction of linear length and locality 3, to testing proximity to the Reed-Solomon code (see Lemma 9.2 below). This reduction builds on the results from Section 8.

Loosely speaking, the R1CS machines relation asserts that a machines's *execution trace* and *memory trace* satisfy a rank-1 constraint system, i.e., each pair of consecutive rows in one of the traces satisfies an R1CS equation. Additionally, the relation ensures that the execution and memory traces are consistent by checking that they are *permutations of each other*.[9] The relation also includes boundary constraints to ensure, e.g., that the machine starts its computation in a valid initial state and halts in an accepting final state.

In the following, we denote the computation time by T, the computation width by k, and the number of constraints in an R1CS matrix by m. Given a trace $f\colon [T] \to \mathbb{F}^k$, we denote by $f(i,j) \in \mathbb{F}^{2k}$ the concatenation of $f(i)$ and $f(j)$. We formally define the R1CS machines relation $\mathcal{R}_{\mathrm{R1M}}$ as follows.

Definition 9.1. *The relation $\mathcal{R}_{\mathrm{R1M}}$ of bounded accepting computations on* **R1CS machines** *consists of pairs $(\mathbb{x}, \mathbb{w})$ defined as follows. An instance $\mathbb{x} = (\mathbb{F}, (A,B,C), (A',B',C'), \mathcal{B})$ consists of a finite field $\mathbb{F}$, matrices $A,B,C \in \mathbb{F}^{m\times 2k}$ defining time constraints, matrices $A',B',C' \in \mathbb{F}^{m\times 2k}$ defining memory constraints, and a set of boundary constraints $\mathcal{B} \subseteq [T]\times[k]\times\mathbb{F}$. A witness $\mathbb{w} = (f,\pi)$ consists of an execution trace $f\colon [T]\to\mathbb{F}^k$ and permutation $\pi\colon [T]\to[T]$. A pair $(\mathbb{x},\mathbb{w})$ is in $\mathcal{R}_{\mathrm{R1M}}$ if the following holds.*

- Time constraints: $\forall\, t \in [T-1],\ \ Af(t,t+1) \circ Bf(t,t+1) = Cf(t,t+1)$.
- Memory constraints: $\forall\, t \in [T-1],\ \ A'f(\pi(t),\pi(t+1)) \circ B'f(\pi(t),\pi(t+1)) = C'f(\pi(t),\pi(t+1))$.
- Boundary constraints: $\forall\,(t,j,\alpha)\in\mathcal{B},\ \ f(t)_j = \alpha$.

The main result of this section is an oracle reduction, with linear length and locality 3, from R1CS machines to testing proximity to the Reed–Solomon code. The verifier uses $\mathrm{poly}(|\mathbb{x}|) = \mathrm{poly}(m,k,\log T)$ field operations, which is *exponentially* faster than the computation time T, since the dependence on the T is polylogarithmic instead of polynomial.

Lemma 9.2. *Let $\mathbb{F}$ be a finite field, H a coset in $\mathbb{F}$, and $L \subseteq \mathbb{F}$ with $L\cap H = \emptyset$. Let $\mathcal{T}\colon [N]\times[n]\to H$ be a trace embedding in $\mathbb{F}$. There is an RS oracle reduction over domain L (Protocol 9.6) for instances $\mathbb{x}$ of $\mathcal{R}_{\mathrm{R1M}}$ over $\mathbb{F}$, with computation time $T = N-1$, width $k \le n/2$, and $m \le n-2$ constraints. The reduction has 3 rounds of interaction and the following parameters:*

length	$6\|L\|$	*soundness error*	$(kT+3Nn+1)/\|\mathbb{F}\|$	*prover time*	$\|L\|\cdot(\log\|L\|+\|\mathbb{x}\|)$
locality	3	*distance*	$\frac{1}{2}(1-4Nn/\|L\|)$	*verifier time*	$\mathrm{poly}(\|\mathbb{x}\|)$

As discussed in Section 2.4, the main technical tool is a *matrix permutation check protocol*, which checks that two matrices are row permutations of one another. Our high-level strategy for proving Lemma 9.2 is to use the foregoing interactive protocol to check consistency between the execution and memory traces of an R1CS machine, and then check each of the traces via the oracle reduction for interactive R1CS automata.

Organization. In Section 9.1 we describe the matrix permutation check protocol, and explain how to represent it via an R1CS equation. In Section 9.2 we prove Lemma 9.2 by using the foregoing sub-protocol and the oracle reduction for interactive R1CS automata in Section 8.

[9] The use of permutations to express machine computations dates back at least to the seminal work of Babai, Fortnow, Levin, and Szegedy [BFLS91], and originates in the study of nearly-linear time reductions among different computation models [GS89; Rob91].

9.1 Matrix permutation check protocol

We describe a protocol for checking that two matrices over a (sufficiently large) finite field $\mathbb{F}$ are row permutations of one another. The protocol leverages interaction with a prover to avoid more expensive tools that establish equivalence under permutations, such as sorting or routing networks. Since we ultimately wish to express the protocol via an interactive R1CS automaton (see Section 9.2), we present the protocol as a distribution over R1CS matrices. We note, however, that this protocol can also be formalized in other ways.

For notational convenience, we view a $T \times k$ matrix over $\mathbb{F}$ as a function $f\colon [T] \to \mathbb{F}^k$. Then $f'\colon [T] \to \mathbb{F}^k$ is a permutation of f if there is a permutation $\pi\colon [T] \to [T]$ such that $f'(t) = f(\pi(t))$ for all $t \in [T]$.

From permutation to identity testing. Checking that two matrices are row permutations of one another can be expressed as a polynomial identity testing problem, as the permutation condition is an equality problem between multi-sets over vectors. We can encode each vector $v \in \mathbb{F}^k$ as a univariate polynomial $q_v(Y) \coloneqq \sum_{j=1}^{k} Y^j v_j$, and encode a multi-set of vectors $S = \{v^{(t)}\}_{t\in[T]}$ as a bivariate polynomial $q_S(X,Y) \coloneqq \prod_{t=1}^{T}(X - q_{v^{(t)}}(Y))$. Then, two multi-sets S and S' are equal if and only if $q_S(X,Y) \equiv q_{S'}(X,Y)$.

This suggests a probabilistic protocol to check the permutation condition. The verifier sends to the prover two random elements $\alpha, \beta \in \mathbb{F}$. Then, the prover has to convince the verifier that $q_S(\alpha,\beta) = q_{S'}(\alpha,\beta)$. This suffices since if $q_S(X,Y) \equiv q_{S'}(X,Y)$, then $q_S(\alpha,\beta) = q_{S'}(\alpha,\beta)$ with probability 1 over α, β; if instead $q_S(X,Y) \not\equiv q_{S'}(X,Y)$, then $q_S(\alpha,\beta) \neq q_{S'}(\alpha,\beta)$ with probability at least $1 - kT/|\mathbb{F}|$ over α, β.

Intuitively, evaluation at β plays the role of a hash function: if two vectors $v, u \in \mathbb{F}^k$ are not equal then, with probability at least $1 - k/|\mathbb{F}|$ over β, it holds that $q_v(\beta) \neq q_u(\beta)$. This "collapses" all vectors in S and S' to single field element such that, with high probability, distinct vectors hash to distinct elements. Evaluation at α plays the role of another hash function, except that this time it is with respect to the vectors of all hashes, namely $(q_v(\beta))_{v\in S}$ and $(q_u(\beta))_{u\in S'}$. After these two hash function evaluations, only two elements need to be compared, namely $q_S(\alpha,\beta)$ and $q_{S'}(\alpha,\beta)$.

Probabilistic checks for multi-set equality like the above ones are familiar techniques from program checking [Lip89; BK95], and have been recently applied to check machine computations (see, e.g., [ZGKPP18]).

From identity testing to a protocol. We need to design a protocol that enables a prover to convince the verifier that a function $f'\colon [T] \to \mathbb{F}^k$ is a permutation of another function $f\colon [T] \to \mathbb{F}^k$. The discussion so far tells us that it suffices for the verifier to learn the random evaluation of bivariate polynomials related to f and f', but does not tell us what protocol to run.

Towards this end, consider the bivariate polynomials $\{\chi_t(X,Y)\}_{t\in[T]}$ defined as follows:

$$\chi_t(X,Y) \coloneqq \prod_{i=1}^{t}\left(X - \sum_{j=1}^{k} Y^j f(i)_j\right) - \prod_{i=1}^{t}\left(X - \sum_{j=1}^{k} Y^j f'(i)_j\right) \quad . \tag{2}$$

Observe that $\chi_T \equiv 0$ if and only if there exists a permutation π such that $f'(t) = f(\pi(t))$ for all $t \in [T]$.

For any choice of $\alpha, \beta \in \mathbb{F}$, we consider an auxiliary trace $g\colon [T+1] \to \mathbb{F}^3$ that "incrementally computes" $\chi_T(\alpha,\beta)$ as follows. The first and second columns of g are defined so that $g(1)_1 = g(1)_2 = 1_{\mathbb{F}}$ and, for $1 < t \leq [T+1]$, $g(t)_1$ and $g(t)_2$ respectively contain the first and second terms of $\chi_{t-1}(\alpha,\beta)$:

$$g(t)_1 \coloneqq \prod_{i=1}^{t-1}\left(\alpha - \sum_{j=1}^{k} \beta^j f(i)_j\right) = g(t-1)_1 \cdot \left(\alpha - \sum_{j=1}^{k} \beta^j f(t-1)_j\right) \quad ,$$

$$g(t)_2 \coloneqq \prod_{i=1}^{t-1}\left(\alpha - \sum_{j=1}^{k} \beta^j f'(i)_j\right) = g(t-1)_2 \cdot \left(\alpha - \sum_{j=1}^{k} \beta^j f'(t-1)_j\right) \quad .$$

Note that $g(t)_1$ and $g(t)_2$ can be derived from $g(t-1)_1$ and $g(t-1)_2$ respectively. The third column of g is the difference of the first two columns: for every $t \in [T+1]$ we define $g(t)_3 = g(t)_1 - g(t)_2 = \chi_{t-1}(\alpha, \beta)$. Observe that, if $\chi_T \not\equiv 0$ then $g(T+1)_3 = \chi_T(\alpha, \beta) = 0$ with probability at most $kT/|\mathbb{F}|$.

We can summarize the above via a statement that involves local constraints among adjacent variables.

Lemma 9.3. *Given $f, f' \colon [T] \to \mathbb{F}^k$, define the probability*

$$\mu(f, f') \coloneqq \Pr_{\alpha,\beta \leftarrow \mathbb{F}} \left[\begin{array}{l} \exists\, g \colon [T+1] \to \mathbb{F}^3 \text{ such that} \\ \bullet\ g(1)_1 = g(1)_2 = 1 \text{ and } g(T+1)_3 = 0 \\ \bullet\ \forall\, t \in [T]\,,\ g(t+1)_1 = g(t)_1 \cdot (\alpha - \sum_{j=1}^{k} \beta^j f(t)_j) \\ \bullet\ \forall\, t \in [T]\,,\ g(t+1)_2 = g(t)_2 \cdot (\alpha - \sum_{j=1}^{k} \beta^j f'(t)_j) \\ \bullet\ \forall\, t \in [T+1]\,,\ g(t)_3 = g(t)_1 - g(t)_2 \end{array} \right] .$$

Then the following conditions hold.

- Completeness: *if f' is a permutation of f then $\mu(f, f') = 1$.*
- Soundness: *if f' is not a permutation of f then $\mu(f, f') \leq kT/|\mathbb{F}|$.*

The protocol via an R1CS equation. We re-write Lemma 9.3 in the language of R1CS equations, so that in Section 9.2 we can embed the matrix permutation check protocol in an interactive R1CS automaton.

For functions $a \colon [T] \to \mathbb{F}^k$ and $b \colon [T'] \to \mathbb{F}^{k'}$, we denote by $a \| b \colon [\max(T, T')] \to \mathbb{F}^{k+k'}$ the function defined as $(a\|b)(t) \coloneqq a(t)\|b(t)$, where we pad a or b via all-zero rows if $T \neq T'$. We construct a distribution of R1CS matrices such that: if f' is a permutation of f, then there exists an auxiliary trace g such that $f\|f'\|g$ satisfies the constraints of the R1CS matrices, and otherwise, with high probability, there is no auxiliary trace g that makes $f\|f'\|g$ satisfy the constraints of the R1CS matrices.

Lemma 9.4. *Let $T, k, n \in \mathbb{N}$ with $n \geq 2k$. Let $\mathsf{0} \colon [T] \to \mathbb{F}^{n-2k}$ always output zeros. There exists a polynomial-time samplable distribution $\mathcal{D}$ over tuples of matrices $(A_{\mathrm{p}}, B_{\mathrm{p}}, C_{\mathrm{p}}) \in (\mathbb{F}^{3\times 4n})^3$, where drawing a sample from $\mathcal{D}$ requires two random elements in $\mathbb{F}$, such that for every $f, f' \colon [T] \to \mathbb{F}^k$ the probability*

$$\mu(f, f') \coloneqq \Pr_{(A_{\mathrm{p}}, B_{\mathrm{p}}, C_{\mathrm{p}}) \leftarrow \mathcal{D}} \left[\begin{array}{c} \exists\, g \colon [T+1] \to \mathbb{F}^n \text{ with } g(1)_1 = g(1)_2 = 1 \text{ and } g(T+1)_3 = 0 \text{ s.t.} \\ \text{letting } h \coloneqq f\|f'\|\mathsf{0}\|g\,,\ \forall\, t \in [T]\,,\ A_{\mathrm{p}} h(t, t+1) \circ B_{\mathrm{p}} h(t, t+1) = C_{\mathrm{p}} h(t, t+1) \end{array} \right],$$

satisfies the following conditions.

- Completeness: *if f' is a permutation of f, then $\mu(f, f') = 1$.*
- Soundness: *if f' is not a permutation of f, then $\mu(f, f') \leq kT/|\mathbb{F}|$.*

Proof. We construct $(A_{\mathrm{p}}, B_{\mathrm{p}}, C_{\mathrm{p}})$ so that, for $h \coloneqq f\|f'\|\mathsf{0}\|g$ with $g(1)_1 = g(1)_2 = 1$ and $g(T+1)_3 = 0$,

$$\{A_{\mathrm{p}} h(t, t+1) \circ B_{\mathrm{p}} h(t, t+1) = C_{\mathrm{p}} h(t, t+1)\}_{t \in [T]} \longrightarrow \{g(t)_3 = g(t)_1 - g(t)_2 = \chi_{t-1}(\alpha, \beta)\}_{t \in [T]} .$$

Viewing $h(t, t+1)$ as a vector $u^{(1)}\|v^{(1)}\|0^{n-2k}\|w^{(1)}\|u^{(2)}\|v^{(2)}\|0^{n-2k}\|w^{(2)} \in \mathbb{F}^{4n}$, for $u^{(1)}, u^{(2)}, v^{(1)}, v^{(2)} \in \mathbb{F}^k$ and $w^{(1)}, w^{(2)} \in \mathbb{F}^{2k}$, we choose the matrices to enforce the following rank-1 constraints:

$$w_1^{(1)} \cdot \left(\alpha - \sum_{j=1}^{k} \beta^j u_j^{(2)} \right) = w_1^{(2)} \ ,$$

$$w_2^{(1)} \cdot \left(\alpha - \sum_{j=1}^{k} \beta^j v_j^{(2)} \right) = w_2^{(2)} \ ,$$
$$w_1^{(2)} - w_2^{(2)} = w_3^{(2)} \ .$$

Note that $A_{\mathrm{p}}, B_{\mathrm{p}}, C_{\mathrm{p}}$ are $3 \times 4n$ matrices over $\mathbb{F}$, as claimed. □

9.2 Proof of Lemma 9.2

We provide an oracle reduction from checking the R1CS machines relation to testing proximity to the Reed-Solomon code. We rely on two ingredients: (a) the matrix permutation check protocol (Lemma 9.4), and (b) the oracle reduction from the R1CS automata relation to testing the Reed-Solomon code (Lemma 8.2).

Observe that checking membership in the relation $\mathcal{R}_{\mathrm{R1M}}$ amounts to checking that: (1) the execution trace f satisfies the time constraints; (2) the memory trace f' satisfies the memory constraints; (3) the memory trace f' is a permutation of the execution trace f; and (4) the execution trace f satisfies the boundary constraints. We "program" an automaton to check time constraints and memory constraints, and additionally program the (interactive) automaton to check the permutation condition via the matrix permutation check protocol in Section 9.1. This latter is the only place wherein we use the interactivity of the automata.

We now elaborate on this plan, by first describing the reduction from machines to interactive automata, and then describing the oracle reduction induced by it.

From machines to interactive automata. Let $\mathbb{x} = (\mathbb{F}, (A, B, C), (A', B', C'), \mathcal{B})$ be an instance for the relation $\mathcal{R}_{\mathrm{R1M}}$ (see Definition 9.1), in which the time constraints are matrices $A, B, C \in \mathbb{F}^{m \times 2k}$, the memory constraints are matrices $A', B', C' \in \mathbb{F}^{m \times 2k}$, and the boundary constraints are a set $\mathcal{B} \subseteq [T] \times [k] \times \mathbb{F}$.

Below we describe how to construct an instance $\mathbb{x}' = (\mathbb{F}, \mathbf{A}, \mathbf{B}, \mathbf{C}, \mathcal{B}', \ell = 2)$ for the relation $\mathcal{R}_{\mathrm{R1A}}^{2,\epsilon}$ with $\epsilon := Nn/|\mathbb{F}|$ (see Definition 8.1). After that we describe how to transform a witness for $\mathbb{x}$ into one for $\mathbb{x}'$.

- **Instance reduction.** First we split each matrix $M \in \{A, B, C, A', B', C'\}$ into halves $M_1, M_2 \in \mathbb{F}^{m \times k}$ by putting the first k columns into M_1 and other k into M_2. Now let $A_{\mathrm{p}}(\alpha, \beta), B_{\mathrm{p}}(\alpha, \beta), C_{\mathrm{p}}(\alpha, \beta) \in \mathbb{F}^{3 \times 4n}$ be the matrices obtained from the distribution $\mathcal{D}$ in Lemma 9.4 with randomness $\alpha, \beta \in \mathbb{F}$.

  We now define the function $\mathbf{A} \colon \mathbb{F}^2 \to \mathbb{F}^{n \times 4n}$.

$$\mathbf{A}(\alpha, \beta) := \begin{array}{|c|c|c|c|c|c|c|c|} \hline A_1 & 0^{m \times k} & 0^{m \times n-2k} & 0^{m \times n} & A_2 & 0^{m \times k} & 0^{m \times n-2k} & 0^{m \times n} \\ \hline 0^{m \times k} & A_1' & 0^{m \times n-2k} & 0^{m \times n} & 0^{m \times k} & A_2' & 0^{m \times n-2k} & 0^{m \times n} \\ \hline \multicolumn{8}{|c|}{A_{\mathrm{p}}(\alpha, \beta)} \\ \hline \multicolumn{8}{|c|}{0^{(n-2m-3) \times 4n}} \\ \hline \end{array}$$

  We similarly define the functions $\mathbf{B}, \mathbf{C} \colon \mathbb{F}^2 \to \mathbb{F}^{n \times 4n}$.[10]

  Observe that we have constructed the functions above so that, given a trace $h \colon [T+1] \to \mathbb{F}^{2n}$ parsed as the concatenation of traces $f \colon [T] \to \mathbb{F}^k$, $f' \colon [T] \to \mathbb{F}^k$, $g \colon [T+1] \to \mathbb{F}^n$ (discarding columns as appropriate), if

$$\forall\, t \in [T]\,,\ \mathbf{A}(\alpha, \beta) h(t, t+1) \circ \mathbf{B}(\alpha, \beta) h(t, t+1) = \mathbf{C}(\alpha, \beta) h(t, t+1) \ ,$$

[10] The functions $\mathbf{A}, \mathbf{B}, \mathbf{C}$ are actually supposed to be from $\mathbb{F}^{r\ell}$ to $\mathbb{F}^{n \times 2rn}$ where r is the number of rounds and ℓ is the number of field elements sent by the verifier in each round (see Definition 8.1). But here the verifier's first message is empty and its second message has two field elements. So, given that $r = 2$, we find it more convenient to take $\mathbf{A}, \mathbf{B}, \mathbf{C}$ to be functions from $\mathbb{F}^2$ to $\mathbb{F}^{n \times 4n}$.

then we know that f satisfies time constraints, f' satisfies memory constraints, and h satisfies the constraints in Lemma 9.4 induced by the randomness α, β.

We define the boundary constraints $\mathcal{B}' \subseteq [T+1] \times [4n] \times \mathbb{F}$ to be the union of the boundary constraints $\mathcal{B}$ in $\mathbb{x}$ and the boundary constraints from the matrix permutation check (in Lemma 9.4). More precisely, $(t, j, \alpha) \in \mathcal{B}'$ if and only if $(t, j, \alpha) \in \mathcal{B}$ or $(t, j, \alpha) \in \{(1, n+1, 1), (1, n+2, 1), (T+1, n+3, 0)\}$.

Note that transforming $\mathbb{x}$ into $\mathbb{x}'$ can be performed in linear time.

- **Witness reduction.** Suppose that $\mathbb{x}$ has a valid witness $\mathbb{w} = (f, \pi)$, namely, $(\mathbb{x}, \mathbb{w}) \in \mathcal{R}_{\mathrm{R1M}}$. Let $f' \colon [T] \to \mathbb{F}^k$ be the memory trace obtained by permuting f according to π: for every $t \in [T]$, we define $f'(t) \coloneqq f(\pi(t))$. Let $P_{\mathbb{w}'}$ be the prover strategy for the game defined by $\mathbb{x}'$ that works as follows:
  1. the prover sends $(f \| f' \| 0) \colon [T] \to \mathbb{F}^{2n}$, the padded concatenation of the execution and memory traces;
  2. the prover receives two elements $\alpha, \beta \in \mathbb{F}$ from the verifier;
  3. the prover uses α, β to construct an auxiliary trace $g \colon [T+1] \to \mathbb{F}^n$ such that $\mu(f, f') = 1$ (as guaranteed by Lemma 9.4);
  4. the prover sends g.

Observe that the foregoing is a (standard, polynomial-time) reduction from $\mathcal{R}_{\mathrm{R1M}}$ to $\mathcal{R}_{\mathrm{R1A}}^{2,\epsilon} =: (\mathcal{R}_{\mathsf{Yes}}, \mathcal{L}_{\mathsf{No}})$.

Claim 9.5. *If $(\mathbb{x}, \mathbb{w}) \in \mathcal{R}_{\mathrm{R1M}}$, then $(\mathbb{x}', P_{\mathbb{w}'}) \in \mathcal{R}_{\mathsf{Yes}}$. If instead $\mathbb{x} \notin \mathcal{L}(\mathcal{R}_{\mathrm{R1M}})$, then $\mathbb{x}' \in \mathcal{L}_{\mathsf{No}}$.*

Proof. Suppose that $(\mathbb{x}, \mathbb{w}) \in \mathcal{R}_{\mathrm{R1M}}$. Then $(\mathbb{x}', P_{\mathbb{w}'}) \in \mathcal{R}_{\mathsf{Yes}}$ because by construction the R1CS automaton defined by $\mathbb{x}'$ runs the permutation check on f, f' (which always passes), time constraints check on f (which always passes), and memory constraints check on f' (which always passes).

Suppose instead that $\mathbb{x} \notin \mathcal{L}(\mathcal{R}_{\mathrm{R1M}})$. We argue that $\mathbb{x}' \in \mathcal{L}_{\mathsf{No}}$. Consider a candidate prover strategy $P_{\mathbb{w}'}$. We need to argue that $P_{\mathbb{w}'}$ wins the game defined by $\mathbb{x}'$ with probability at most $\epsilon = kT/|\mathbb{F}|$. Let $(f \| f') \colon [T] \to \mathbb{F}^{2k}$ be the first message sent by $P_{\mathbb{w}'}$. If f does not satisfy the time constraints in $\mathbb{x}$ or f' does not satisfy the memory constraints in $\mathbb{x}$, then the prover loses with probability 1, because the R1CS automaton $\mathbb{x}'$ checks both of these conditions. So suppose that f and f' satisfy the time and memory constraints respectively. This means that f' is *not* a permutation of f (for otherwise $\mathbb{x}$ would have been in the language of $\mathcal{R}_{\mathrm{R1M}}$), and so we can use the soundness condition of Lemma 9.4. Indeed, we know that, with probability at least $1 - kT/|\mathbb{F}|$ over the verifier's choice of $\alpha, \beta \in \mathbb{F}$, the second message $g \colon [T+1] \to \mathbb{F}^3$ of the prover does *not* satisfy the permutation check constraints, in which case the prover loses. □

Protocol 9.6. Let $\mathbb{F}$ be a finite field, H a coset in $\mathbb{F}$, and $L \subseteq \mathbb{F}$ with $L \cap H = \emptyset$. Let $\mathcal{T} \colon [N] \times [n] \to H$ be a trace embedding in $\mathbb{F}$ with $N = T + 1$ and $n \geq 2k$. We need to construct an oracle reduction (P, V) that works for instances $\mathbb{x} = (\mathbb{F}, (A, B, C), (A', B', C'), \mathcal{B})$ of computation time T and width k.

The oracle reduction is straightforward: we reduce the R1CS machine to an interactive R1CS automaton and then invoke the oracle reduction (P', V') from Lemma 8.2. (Note that the hypothesis in Lemma 9.2 is the same as in Lemma 8.2, so that we can indeed invoke the latter.)

In more detail, the prover P and verifier V each transform the given instance $\mathbb{x}$ for the relation $\mathcal{R}_{\mathrm{R1M}}$ into the instance $\mathbb{x}'$ for the relation $\mathcal{R}_{\mathrm{R1A}}^{2,\epsilon} = (\mathcal{R}_{\mathsf{Yes}}, \mathcal{L}_{\mathsf{No}})$, following the instance reduction described above. Also, the prover P transforms a witness $\mathbb{w} = (f, \pi)$ for $\mathbb{x}$ into a witness $P_{\mathbb{w}'}$ for $\mathbb{x}'$, following the witness reduction described above. Then, letting the prover P and V engage in an oracle reduction with P playing the role of $P'(\mathbb{x}', P_{\mathbb{w}'})$ and V playing the role of $V'(\mathbb{x}')$. Finally, the verifier V outputs whatever V' outputs.

Completeness. If $(\mathbb{x}, \mathbb{w}) \in \mathcal{R}_{\mathrm{R1M}}$, then $(\mathbb{x}', P_{\mathbb{w}'}) \in \mathcal{R}_{\mathsf{Yes}}$, and completeness of the oracle reduction (P', V') for automata implies completeness of the oracle reduction (P, V) for machines.

Soundness. If $\mathbb{x} \notin \mathcal{L}(\mathcal{R}_{\mathrm{R1M}})$, then $\mathbb{x} \in \mathcal{L}_{\mathsf{No}}$, and so we obtain the soundness guarantee of the oracle reduction (P', V').

Efficiency. The prover P runs in time $\mathrm{poly}(|\mathbb{x}| + T)$ and the verifier V runs in time $\mathrm{poly}(|\mathbb{x}|)$ because the respective running times in the oracle reduction (P', V') are $\mathrm{poly}(|\mathbb{x}'| + N)$ and the verifier V runs in time $\mathrm{poly}(|\mathbb{x}'|)$ and it holds that $|\mathbb{x}'| = O(|\mathbb{x}|)$ and $N = O(T)$. (Also, $\mathbb{x}'$ can be efficiently derived from $\mathbb{x}$.)

The locality of (P, V) is 3 because the locality of (P', V') is $r + 1$ when the interactive automaton has r rounds, which in the case of $\mathbb{x}'$ is $r = 2$. The length of (P, V) is $6|L|$.

10 Proofs of main results

In Section 10.1 we prove Theorem 3, and in Section 10.2 we prove Theorem 2.

10.1 Checking satisfiability of algebraic machines

In Section 9 we obtained an oracle reduction from the R1CS machine relation to testing proximity to the Reed–Solomon code. We now combine this oracle reduction with a linear-size IOP of proximity for the Reed–Solomon code of [BCGRS17] to obtain our main result. We first recall this latter result.

Lemma 10.1 ([BCGRS17, Theorem 5.1])**.** *Fix a rate parameter $\rho \in (0,1)$ and a proximity parameter $\delta \in (0,(1-\rho)/2)$. Let $\mathbb{F}$ be a finite field, L_0 a subgroup of $\mathbb{F}$ that itself has a subgroup of size $\Theta(|L_0|^\alpha)$ for some $\alpha \in (0,1)$, and let L be a coset of L_0. There exists a 2-round IOPP system for $\mathrm{RS}[L,\rho]$ with linear proof length, $\mathsf{q}_\mathsf{w} = 1$, $\mathsf{q}_\pi = 2$, distance parameter δ, constant soundness error, and constant query complexity. The prover uses $O(|L|\operatorname{polylog}|L|)$ field operations and the verifier uses $\operatorname{polylog}(|L|)$ field operations. The verifier's first message is empty.*

The result in [BCGRS17] is stated with soundness $1/2$ and some constant query complexity; applying query reduction to the protocol's second round yields the same result but with 3 queries and constant soundness.

Next we introduce our definition for *large smooth fields*, which captures the properties that we use to construct suitable trace embeddings. When a field is $(T(n),k(n),\rho(n))$-smooth according to the definition below, we can use the algorithm of Lemma 6.2 to construct a trace embedding of size $T(n)\times O(k(n))$ in time $\operatorname{poly}(\log T(n),k(n))$. For example, the family $\{\mathbb{F}_{p^{2^n}}\}_{n\in\mathbb{N}}$ is $(p^n,O(n),O(1))$-smooth; the same is true of fields with smooth multiplicative subgroups in the sense of [BS08]. The additional, and somewhat technical, conditions in the definition ensure the existence of a subgroup L_0 of $\mathbb{F}$ of size $\Theta(k(n)\cdot T(n))$ with a suitable coset L, so that we can invoke Lemma 10.1 with rate parameter $\rho(n)$.

Definition 10.2. *A family of fields $\{\mathbb{F}(n)\}_{n\in\mathbb{N}}$ is $(T(n),k(n),\rho(n))$-**smooth** if there exists $\alpha\in(0,1)$ and a family $\{H_1(n),H_2(n),L_0(n)\}_{n\in\mathbb{N}}$, such that $H_1(n),H_2(n),L_0(n)$ are subgroups of $\mathbb{F}(n)$, where for all n:*

- *$|H_1(n)| = T(n)$, $k(n)\le|H_2(n)| = O(k(n))$, and $|H_1(n)\cap H_2(n)| = 1$;*
- *$L_0(n)$ has a subgroup of size $\Theta(|L_0(n)|^\alpha)$;*
- *if $H(n)$ is the smallest subgroup of $\mathbb{F}(n)$ containing $H_1(n)$ and $H_2(n)$ then $L_0(n)$ contains $H(n)$ with $\rho(n)\ge|H(n)|/|L_0(n)| = \Omega(\rho(n))$.*

Note that if $\{\mathbb{F}(n)\}_{n\in\mathbb{N}}$ is $(T(n),k(n),\rho(n))$-smooth then it is also $(T(n),ck(n),c'\rho(n))$-smooth for any constants $c\in(0,1]$, $c'>1$.

The above condition suffices for the construction to be feasible, but for prover efficiency we require much more structure. The following condition guarantees the existence of a fast Fourier transform which runs in time $O(n\log n)$; it is a usual notion of smoothness for integers. Again, ensembles of binary fields or fields with smooth multiplicative subgroups satisfy this definition. Crucially for us, if our field family satisfies the following condition then the prover time in Lemma 10.1 can be reduced to $O(|L|\log|L|)$.

Definition 10.3. *A family of fields $\{\mathbb{F}(n)\}_{n\in\mathbb{N}}$ is $(T(n),k(n),\rho(n))$-**very smooth** if it is $(T(n),k(n),\rho(n))$-smooth and there exists a constant c such that the prime factors of $|L_0(n)|$ are all at most c for all n.*

To give the formal statement of our main theorem, we first define a parameterized version of the R1CS machine relation.

Definition 10.4. *The relation $\mathcal{R}_{\mathrm{R1M}}[\mathbb{F}(n), T(n), k(n)]$ consists of pairs $(\mathtt{x}, \mathtt{w}) \in \mathcal{R}_{\mathrm{R1M}}$ such that $\mathbb{F} = \mathbb{F}(n)$, $T = T(n)$ and $k \leq k(n)$.*

We derive our main result (informally stated in Theorem 3) by combining Lemma 9.2 and Lemma 10.1. We assume that $\mathbb{F}(n)$ is uniformly specified via a primitive element, and also via the factorization of $|\mathbb{F}(n)|$ and $|\mathbb{F}(n)^*|$ so that we can efficiently construct any (additive or multiplicative) subgroup of $\mathbb{F}(n)$.

Theorem 10.5. *Let $\mathcal{F} = \{\mathbb{F}(n)\}_{n\in\mathbb{N}}$ be a $(T(n)+1, 2k(n), \rho(n))$-smooth field family with $T(n) \geq n$, $k(n) = \mathrm{poly}(n)$; let $S(n) := k(n)T(n)/\rho(n)$. There exists a universal constant ϵ_0 such that there exists a 5-round IOP for $\mathcal{R}_{\mathrm{R1M}}[\mathbb{F}(n), T(n), k(n)]$ with a proof length of $O(S(n))$ field elements, 5 queries, and soundness error ϵ_0. The verifier uses $\mathrm{poly}(n, \log T(n))$ field operations.*

Moreover, if $\mathcal{F}$ is very smooth then the prover uses $O(S(n)(\log S(n) + n))$ field operations.

Proof. Let $\mathtt{x}$ be an instance of the R1CS machine relation $\mathcal{R}_{\mathrm{R1M}}[\mathbb{F}(n), T(n), k(n)]$ (see Definition 9.1). Since $\mathbb{F}(n)$ comes from a smooth family, we can use Lemma 6.2 to efficiently construct a trace embedding $\mathcal{T}\colon H(n) \to H_1(n) \times H_2(n)$ with $|H_1(n)| = T(n) + 1$ and $2k(n) \leq |H_2(n)| = O(k(n))$; these choices, from the theorem's hypothesis, are compatible with Lemma 9.2.

Moreover, let $L_0(n)$ be as guaranteed by the smoothness condition, and let $L(n)$ be a coset of $L_0(n)$ which is not $L_0(n)$ itself. Since $H(n) \subseteq L_0(n)$, we have $L(n) \cap H(n) = \varnothing$ (as required by Lemma 9.2).

By Lemma 9.2, there exists a 3-round RS oracle reduction over domain L from $\mathcal{R}_{\mathrm{R1M}}$. By Lemma 10.1, and the smoothness condition, there exists a 2-round IOP of proximity for $\mathcal{R}_{\mathrm{RS}}$ over L. Applying Corollary 5.10 to these components yields a 5-round IOP for $\mathcal{R}_{\mathrm{RS}}$ with the stated parameters. □

10.2 Checking satisfiability of succinct arithmetic circuits

In this section we prove our result for the relation Succinct-ASAT, which consists of succinctly-represented arithmetic circuits that are satisfiable. We begin by defining this relation.

Definition 10.6. *Let $m \in \mathbb{N}$, $E\colon \mathbb{F}^m \to \mathbb{F}^m$ be an arithmetic circuit, $o \in \mathbb{F}^m$, and $T \in \mathbb{N}$. We define $H_{E,o,T}$ to be the set $\{a_1, \ldots, a_T\} \subseteq \mathbb{F}^m$, where $a_1 := o$ and, for all $i \in \{1, \ldots, T-1\}$, $a_{i+1} := E(a_i)$. In other words, E* **enumerates** *the set $H_{E,o,T}$ as o, $E(o)$, $E(E(o))$, and so on.*

Definition 10.7. *The relation* Succinct-ASAT *consists of pairs $\big((\mathbb{F}, m, E, o, T, I, D), w\big)$, where $\mathbb{F}$ is a finite field, $m \in \mathbb{N}$, $E\colon \mathbb{F}^m \to \mathbb{F}^m$ is an arithmetic circuit for enumerating gates, $o \in \mathbb{F}^m$ is the label of the output gate, $T \in \mathbb{N}$ is the number of gates, I is a subset of $H_{E,o,T}$ representing the input gates, $D\colon \mathbb{F}^m \to (\{+, \times\} \times H_{E,o,T} \times H_{E,o,T}) \cup \{\mathbb{F}\}$ is an arithmetic circuit that describes an arithmetic circuit C, and the witness $w\colon I \to \mathbb{F}$ is such that $C(w) = 0$.*

We define a parameterized version of Succinct-ASAT.

Definition 10.8. *The relation* Succinct-ASAT$[\mathbb{F}(n), T(n), k(n)]$ *consists of pairs $(\mathtt{x}, \mathtt{w}) \in$ Succinct-ASAT such that $\mathbb{F} = \mathbb{F}(n)$, $T = T(n)$ and $|E| + |D| \leq k(n)$.*

Next we give the formal statement of Theorem 2.

Theorem 10.9. *There exist universal constants $\epsilon_0 \in (0, 1)$, $c \in \mathbb{N}$ such that for any $(T(n)+1, ck(n), O(1))$-very smooth field family with $T(n) \geq n$, there is a 5-round IOP for* Succinct-ASAT$[\mathbb{F}(n), T(n), k(n)]$ *with a proof length of $O(S)$ field elements for $S := T \cdot k(n)$, 5 queries, and soundness error ϵ_0. The prover uses $O(S \log S)$ field operations and the verifier uses $\mathrm{poly}(|E|, |D|, k(n))$.*

Above, $|E|, |D|$ are the number of gates in E, D respectively. Note that C has T gates whose names are in $\mathbb{F}^m$, and hence the number of field elements needed to represent C is $\Theta(T \cdot m)$, because the representation includes for each gate in $H_{E,o,T} \subseteq \mathbb{F}^m$ the names of the gates to which that gate is connected. In particular, the proof length in Theorem 10.9 is linear in the number of field elements to represent C when $|E|, |D| = O(m)$.

The proof of Theorem 10.9 is a direct implication of the following lemma.

Lemma 10.10. *There exists a universal constant $c \in N$ such that there is a polynomial-time reduction from* Succinct-ASAT$[\mathbb{F}(n), T(n), k(n)]$ *to* $\mathcal{R}_{\mathrm{R1M}}[\mathbb{F}(n), T(n), ck(n)]$. *The size of the $\mathcal{R}_{\mathrm{R1M}}$ instance is linear in the size of the* Succinct-ASAT *instance.*

Proof sketch. We begin by describing the witness reduction. Let $\big((\mathbb{F}, m, E, o, T, I, D), w\big) \in$ Succinct-ASAT. The witness w assigns a value to every wire of C.

- The first part of the witness for the algebraic machine is a function $f\colon [3T] \times [k] \to \mathbb{F}$, where $k = c(|E| + |D|)$, as follows. For each gate $g \in H_{E,o,T}$, ordered by E starting from o, we add three rows corresponding to g, labelled l, r, o respectively. These rows include the gate label g and the labels and values of the input wires.
- The second part of the witness f' lists, for each gate g ordered by E starting from o, all the rows in which it appears in f: first the unique row labelled o corresponding to g, then all rows labelled l where g is a left input, then all rows labelled r where g is a right input.

The label l, r, o determines uniquely, for each row, to which gate it belongs in f'. Hence f' is a permutation of f; this permutation π is the last part of the witness. The whole witness consists of $O(T \cdot (|E| + |D|))$ field elements.

Now we discuss the instance reduction. The time constraints check that f is ordered correctly according to E, each gate g is correctly evaluated according to D, and that the left, right and output entries for g are all present. The memory constraints check that f' is ordered correctly according to E, and that the assignment of the g-wire in each row is consistent with the value of the gate g. Clearly the size of this constraint system is linear in $|E| + |D|$.

Finally, we have a boundary constraint that checks that the output of C is 0. □

Acknowledgments

We thank Michael Forbes for helpful discussions. This work was supported in part by: the UKRI Future Leaders Fellowship MR/S031545/1, and donations from the Ethereum Foundation and the Interchain Foundation.

References

[AGHP92] Noga Alon, Oded Goldreich, Johan Håstad, and René Peralta. "Simple Construction of Almost k-wise Independent Random Variables". In: *Random Structures and Algorithms* 3.3 (1992), pp. 289–304.

[ALMSS98] Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy. "Proof verification and the hardness of approximation problems". In: *Journal of the ACM* 45.3 (1998). Preliminary version in FOCS '92., pp. 501–555.

[AS98] Sanjeev Arora and Shmuel Safra. "Probabilistic checking of proofs: a new characterization of NP". In: *Journal of the ACM* 45.1 (1998). Preliminary version in FOCS '92., pp. 70–122.

[BBHR18a] Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, and Michael Riabzev. *Scalable, transparent, and post-quantum secure computational integrity*. Cryptology ePrint Archive, Report 2018/046. 2018.

[BBHR18b] Eli Ben-Sasson, Iddo Bentov, Ynon Horesh, and Michael Riabzev. "Fast Reed–Solomon Interactive Oracle Proofs of Proximity". In: *Proceedings of the 45th International Colloquium on Automata, Languages and Programming*. ICALP '18. 2018, 14:1–14:17.

[BCFGRS17] Eli Ben-Sasson, Alessandro Chiesa, Michael A. Forbes, Ariel Gabizon, Michael Riabzev, and Nicholas Spooner. "Zero Knowledge Protocols from Succinct Constraint Detection". In: *Proceedings of the 15th Theory of Cryptography Conference*. TCC '17. 2017, pp. 172–206.

[BCGGHJ17] Jonathan Bootle, Andrea Cerulli, Essam Ghadafi, Jens Groth, Mohammad Hajiabadi, and Sune K. Jakobsen. "Linear-Time Zero-Knowledge Proofs for Arithmetic Circuit Satisfiability". In: *Advances in Cryptology - ASIACRYPT 2017 - 23rd International Conference on the Theory and Applications of Cryptology and Information Security*. 2017, pp. 336–365.

[BCGRS17] Eli Ben-Sasson, Alessandro Chiesa, Ariel Gabizon, Michael Riabzev, and Nicholas Spooner. "Interactive Oracle Proofs with Constant Rate and Query Complexity". In: *Proceedings of the 44th International Colloquium on Automata, Languages and Programming*. ICALP '17. 2017, 40:1–40:15.

[BCGT13a] Eli Ben-Sasson, Alessandro Chiesa, Daniel Genkin, and Eran Tromer. "Fast Reductions from RAMs to Delegatable Succinct Constraint Satisfaction Problems". In: *Proceedings of the 4th Innovations in Theoretical Computer Science Conference*. ITCS '13. 2013, pp. 401–414.

[BCGT13b] Eli Ben-Sasson, Alessandro Chiesa, Daniel Genkin, and Eran Tromer. "On the Concrete Efficiency of Probabilistically-Checkable Proofs". In: *Proceedings of the 45th ACM Symposium on the Theory of Computing*. STOC '13. 2013, pp. 585–594.

[BCGV16] Eli Ben-Sasson, Alessandro Chiesa, Ariel Gabizon, and Madars Virza. "Quasilinear-Size Zero Knowledge from Linear-Algebraic PCPs". In: *Proceedings of the 13th Theory of Cryptography Conference*. TCC '16-A. 2016, pp. 33–64.

[BCRSVW19] Eli Ben-Sasson, Alessandro Chiesa, Michael Riabzev, Nicholas Spooner, Madars Virza, and Nicholas P. Ward. "Aurora: Transparent Succinct Arguments for R1CS". In: *Proceedings of the 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques*. EUROCRYPT '19. Full version available at `https://eprint.iacr.org/2018/828`. 2019, pp. 103–128.

[BCS16] Eli Ben-Sasson, Alessandro Chiesa, and Nicholas Spooner. "Interactive Oracle Proofs". In: *Proceedings of the 14th Theory of Cryptography Conference*. TCC '16-B. 2016, pp. 31–60.

[BFLS91] László Babai, Lance Fortnow, Leonid A. Levin, and Mario Szegedy. "Checking computations in polylogarithmic time". In: *Proceedings of the 23rd Annual ACM Symposium on Theory of Computing*. STOC '91. 1991, pp. 21–32.

[BGHSV05] Eli Ben-Sasson, Oded Goldreich, Prahladh Harsha, Madhu Sudan, and Salil Vadhan. "Short PCPs Verifiable in Polylogarithmic Time". In: *Proceedings of the 20th Annual IEEE Conference on Computational Complexity*. CCC '05. 2005, pp. 120–134.

[BGHSV06] Eli Ben-Sasson, Oded Goldreich, Prahladh Harsha, Madhu Sudan, and Salil P. Vadhan. "Robust PCPs of Proximity, Shorter PCPs, and Applications to Coding". In: *SIAM Journal on Computing* 36.4 (2006), pp. 889–974.

[BK95] Manuel Blum and Sampath Kannan. "Designing Programs That Check Their Work". In: *Journal of the ACM* 42.1 (1995). Preliminary version in STOC '89., pp. 269–291.

[BKKMS13] Eli Ben-Sasson, Yohay Kaplan, Swastik Kopparty, Or Meir, and Henning Stichtenoth. "Constant Rate PCPs for Circuit-SAT with Sublinear Query Complexity". In: *Proceedings of the 54th Annual IEEE Symposium on Foundations of Computer Science*. FOCS '13. 2013, pp. 320–329.

[BKS18] Eli Ben-Sasson, Swastik Kopparty, and Shubhangi Saraf. "Worst-Case to Average Case Reductions for the Distance to a Code". In: *Proceedings of the 33rd ACM Conference on Computer and Communications Security*. CCS '18. 2018, 24:1–24:23.

[BS08] Eli Ben-Sasson and Madhu Sudan. "Short PCPs with Polylog Query Complexity". In: *SIAM Journal on Computing* 38.2 (2008). Preliminary version appeared in STOC '05., pp. 551–607.

[BSVW03] Eli Ben-Sasson, Madhu Sudan, Salil Vadhan, and Avi Wigderson. "Randomness-efficient low degree tests and short PCPs via epsilon-biased sets". In: *Proceedings of the 35th Annual ACM Symposium on Theory of Computing*. STOC '03. 2003, pp. 612–621.

[Bab85] László Babai. "Trading group theory for randomness". In: *Proceedings of the 17th Annual ACM Symposium on Theory of Computing*. STOC '85. 1985, pp. 421–429.

[Bow+18] Sean Bowe et al. *Implementation Track Proceeding*. Tech. rep. `https://zkproof.org/documents.html`. ZKProof Standards, 2018.

[CZ15] Alessandro Chiesa and Zeyuan Allen Zhu. "Shorter arithmetization of nondeterministic computations". In: *Theoretical Computer Science* 600 (2015), pp. 107–131.

[Din07] Irit Dinur. "The PCP theorem by gap amplification". In: *Journal of the ACM* 54.3 (2007), p. 12.

[FGLSS96] Uriel Feige, Shafi Goldwasser, Laszlo Lovász, Shmuel Safra, and Mario Szegedy. "Interactive proofs and the hardness of approximating cliques". In: *Journal of the ACM* 43.2 (1996). Preliminary version in FOCS '91., pp. 268–292.

[GI05] Venkatesan Guruswami and Piotr Indyk. "Linear-time encodable/decodable codes with near-optimal rate". In: *IEEE Transactions on Information Theory* 51.10 (2005). Preliminary version appeared in STOC '03., pp. 3393–3400.

[GMR89] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. "The knowledge complexity of interactive proof systems". In: *SIAM Journal on Computing* 18.1 (1989). Preliminary version appeared in STOC '85., pp. 186–208.

[GS06] Oded Goldreich and Madhu Sudan. "Locally testable codes and PCPs of almost-linear length". In: *Journal of the ACM* 53 (4 2006). Preliminary version in STOC '02., pp. 558–655.

[GS89] Yuri Gurevich and Saharon Shelah. "Nearly linear time". In: *Logic at Botik '89, Symposium on Logical Foundations of Computer Science*. 1989, pp. 108–118.

[HS00] Prahladh Harsha and Madhu Sudan. "Small PCPs with Low Query Complexity". In: *Computational Complexity* 9.3–4 (2000). Preliminary version in STACS '01., pp. 157–201.

[IMSX15] Yuval Ishai, Mohammad Mahmoody, Amit Sahai, and David Xiao. *On Zero-Knowledge PCPs: Limitations, Simplifications, and Applications*. Available at `http://www.cs.virginia.edu/~mohammad/files/papers/ZKPCPs-Full.pdf`. 2015.

[KR08] Yael Kalai and Ran Raz. "Interactive PCP". In: *Proceedings of the 35th International Colloquium on Automata, Languages and Programming*. ICALP '08. 2008, pp. 536–547.

[Kil92] Joe Kilian. "A note on efficient zero-knowledge proofs and arguments". In: *Proceedings of the 24th Annual ACM Symposium on Theory of Computing*. STOC '92. 1992, pp. 723–732.

[LN97] Rudolf Lidl and Harald Niederreiter. *Finite Fields*. Second Edition. Cambridge University Press, 1997.

[Lip89] Richard J. Lipton. "New Directions In Testing". In: *Proceedings of a DIMACS Workshop in Distributed Computing And Cryptography*. 1989, pp. 191–202.

[Mic00] Silvio Micali. "Computationally Sound Proofs". In: *SIAM Journal on Computing* 30.4 (2000). Preliminary version appeared in FOCS '94., pp. 1253–1298.

[Mie09] Thilo Mie. "Short PCPPs verifiable in polylogarithmic time with O(1) queries". In: *Annals of Mathematics and Artificial Intelligence* 56 (3 2009), pp. 313–338.

[PS94] Alexander Polishchuk and Daniel A. Spielman. "Nearly-linear size holographic proofs". In: *Proceedings of the 26th Annual ACM Symposium on Theory of Computing*. STOC '94. 1994, pp. 194–203.

[PY86] Christos H. Papadimitriou and Mihalis Yannakakis. "A Note on Succinct Representations of Graphs". In: *Information and Control* 71.3 (1986), pp. 181–185.

[RR19] Noga Ron-Zewi and Ron D. Rothblum. *Local Proofs Approaching the Witness Length*. Cryptology ePrint Archive, Report 2019/1062. 2019. URL: `https://eprint.iacr.org/2019/1062`.

[RRR16] Omer Reingold, Ron Rothblum, and Guy Rothblum. "Constant-Round Interactive Proofs for Delegating Computation". In: *Proceedings of the 48th ACM Symposium on the Theory of Computing*. STOC '16. 2016, pp. 49–62.

[Rob91] J. M. Robson. "An O(T log T) reduction from RAM computations to satisfiability". In: *Theoretical Computer Science* 82.1 (1991), pp. 141–149.

[SY10] Amir Shpilka and Amir Yehudayoff. "Arithmetic Circuits: A survey of recent results and open questions". In: *Foundations and Trends in Theoretical Computer Science* 5.3-4 (2010), pp. 207–388.

[Spi96] Daniel A. Spielman. "Linear-time encodable and decodable error-correcting codes". In: *IEEE Transactions on Information Theory* 42.6 (1996). Preliminary version appeared in STOC '95., pp. 1723–1731.

[ZGKPP18] Yupeng Zhang, Daniel Genkin, Jonathan Katz, Dimitrios Papadopoulos, and Charalampos Papamanthou. "vRAM: Faster Verifiable RAM with Program-Independent Preprocessing". In: *Proceedings of the 39th IEEE Symposium on Security and Privacy*. S&P '18. 2018, pp. 908–925.

[Ben+17] Eli Ben-Sasson et al. "Computational integrity with a public random string from quasi-linear PCPs". In: *Proceedings of the 36th Annual International Conference on Theory and Application of Cryptographic Techniques*. EUROCRYPT '17. 2017, pp. 551–579.