Zero-Knowledge Succinct Arguments with a Linear-Time Prover

Jonathan Bootle
jbt@zurich.ibm.com
IBM Research – Zurich

Alessandro Chiesa
alexch@berkeley.edu
UC Berkeley

Siqi Liu
sliu18@berkeley.edu
UC Berkeley

December 4, 2020

Abstract

We construct a zero knowledge argument system with polylogarithmic communication complexity where the prover runs in linear time and the verifier runs in polylogarithmic time. This achieves a central goal in the area of efficient zero knowledge.

Our result is a direct consequence of a new interactive oracle proof (IOP) that simultaneously achieves linear-time proving and zero knowledge. We construct an IOP where, for the satisfiability of an N-gate arithmetic circuit over any field of size $\Omega(N)$, the prover uses $O(N)$ field operations and the verifier uses $\mathsf{polylog}(N)$ field operations (with proof length $O(N)$ and query complexity $\mathsf{polylog}(N)$). Polylogarithmic verification is achieved in the holographic setting for every circuit (the verifier has oracle access to a linear-time-computable encoding of the circuit whose satisfiability is being proved).

Keywords: succinct arguments; zero knowledge; interactive oracle proofs

Contents

1 Introduction

Zero knowledge proofs enable a prover to convince a verifier that a statement is true without revealing any further information about the statement [GMR89]. The main efficiency measures in a zero knowledge proof are the running time of the prover, the running time of the verifier, and the number of bits exchanged between them. A central goal in the study of zero knowledge proofs is to minimize the complexity of these measures.

Motivated by real-world applications, researchers across multiple communities have invested significant effort, and made much progress, in designing efficient zero knowledge protocols.

Several works (e.g., [IKOS07; GMO16; Cha+17; KKW18; HK20; WYKW20]) focus on prover time. They construct zero-knowledge proofs for circuit satisfiability where the prover's time complexity is linear in circuit size, which is asymptotically optimal.[1] The drawback of these constructions is that communication complexity and verifier time also grow linearly with circuit size, which is undesirable for many applications.

This drawback is inevitable because, even without zero knowledge, interactive proofs for hard languages with sublinear communication are unlikely [GH98; GVW02]. Nevertheless, if instead of considering proofs we consider *arguments* [BCC88], wherein soundness is required to hold only against efficient adversaries rather than all adversaries, then we can hope to avoid the drawback. For this, rather than studying proofs where zero knowledge holds computationally, one studies arguments where zero knowledge holds statistically.[2]

In a seminal work, Kilian [Kil92] constructed zero knowledge arguments that are *succinct*: communication complexity is *polylogarithmic* in computation size. While this is essentially optimal, the prover in Kilian's construction is a polynomial-time algorithm that fails to achieve the asymptotically-optimal linear time that can be achieved via (non-succinct) zero knowledge proofs.

The time complexity of the prover in succinct arguments has been the subject of intense study, which has led to zero-knowledge succinct arguments with a quasilinear-time prover (see Section 1.2) and also numerous instantiations that are useful in practice. However, the basic question of whether zero-knowledge succinct arguments can have linear-time provers has remained open.

Do there exist zero-knowledge succinct arguments with a linear-time prover?

In this paper we give a positive answer to this question. Prior to this work, the question was open even when dropping the requirement of zero knowledge.

1.1 Our results

We construct a zero-knowledge succinct argument with a linear-time prover, under standard complexity assumptions. Our protocol is for a standard generalization of arithmetic circuit satisfiability, known as *rank-1 constraint satisfiability* (R1CS), where the "circuit description" is given by coefficient matrices.[3]

Definition 1 (informal)**.** *The R1CS problem asks: given a finite field $\mathbb{F}$, coefficient matrices $A, B, C \in \mathbb{F}^{n \times n}$ each containing at most $m = \Omega(n)$ non-zero entries,[4] and an instance vector $x \in \mathbb{F}^*$, is there a witness vector $w \in \mathbb{F}^*$ such that $z := (x, w) \in \mathbb{F}^n$ and $Az \circ Bz = Cz$? (Here "$\circ$" denotes the entry-wise product.)*

[1] Several of these works additionally achieve excellent concrete efficiency, via experiments that demonstrate the ability to prove the satisfiability of circuits with billions of gates.

[2] As soundness is computational then we can hope for zero knowledge to be statistical.

[3] Satisfiability of an n-gate arithmetic circuit over the field $\mathbb{F}$ is reducible, *in linear time*, to an R1CS instance also over $\mathbb{F}$ where the coefficient matrices are $n \times n$ and have $m = O(n)$ non-zero entries. (In particular, the coefficient matrices are sparse.)

[4] Note that $m = \Omega(n)$ without loss of generality because if $m < n/3$ then there are variables of z that do not participate in any constraint, which can be dropped. Thus the main size measure for R1CS is the sparsity parameter m.

Merely checking the validity of a witness by directly checking the R1CS condition costs $O(m)$ field operations, so "linear time" for R1CS means computations that cost no more than $O(m)$ field operations.

The theorem below relies on any linear-time collision-resistant hash function (given as a black box). Such hash functions are known to exist, e.g., under certain assumptions about finding short codewords in linear codes [AHIKV17]; moreover, these candidate hash functions are not known to be insecure against quantum adversaries, and so our succinct argument is plausibly post-quantum secure.

Theorem 1 (informal)**.** *Using any linear-time collision-resistant hash function with security parameter λ as a black box, one can obtain a private-coin interactive argument for R1CS, over any field of size $\Omega(m)$, where:*
- *time complexity of the prover is bounded by the cost of $O(\lambda + m)$ field operations;*
- *time complexity of the verifier is bounded by the cost of $\mathsf{poly}(\lambda, |x|, \log m)$ field operations;*
- *round complexity is $O(\log m)$;*
- *communication complexity is $\mathsf{poly}(\lambda, \log m)$ field elements;*
- *soundness error is $O(1)$.*

Moreover, the argument is malicious-verifier zero-knowledge.

This gives the first cryptographic zero-knowledge argument system with linear prover complexity and both polylogarithmic verifier complexity and argument size.

The polylogarithmic verifier time in Theorem 1 is achieved in the preprocessing setting, which means that the verifier receives as input a short digest of the circuit that can be derived by anyone (in linear time). Some form of preprocessing is necessary for sublinear verification because just reading the circuit takes linear time.

Technical core: linear-time probabilistic proofs. The technical core of this paper is a probabilistic proof that simultaneously achieves zero knowledge, linear-time proving and polylogarithmic-time verification (as well as linear proof size and polylogarithmic query complexity). In more detail, we construct an interactive oracle proof (IOP) [BCS16; RRR16] for the R1CS problem with the parameters below. Our result significantly improves over prior linear-time IOPs, as summarized in Figure 1.

Theorem 2 (informal)**.** *There is a public-coin IOP for R1CS over any field $\mathbb{F}$ of size $\Omega(m)$, where:*
- *the prover uses $O(m)$ field operations;*
- *the verifier uses $\mathsf{poly}(|x|, \log m)$ field operations;*
- *round complexity is $O(\log m)$;*
- *proof length is $O(m)$ elements in $\mathbb{F}$;*
- *query complexity is $O(\log m)$;*
- *soundness error $O(1)$.*

Moreover, the IOP is semi-honest-verifier zero-knowledge.

The above theorem directly implies Theorem 1, via a known implication that involves combining IOPs and linear-time collision resistant hashing [BCGGHJ17]. We review this implication in Section 2.1 and, in the meantime, we remark that the sublinear verifier time in Theorem 2 is achieved in the holographic setting, which means that the verifier is given query access to a linear-size encoding of the coefficient matrices that is computable in linear time. Holographic proofs lead to preprocessing arguments [CHMMVW20; COS20], enabling the polylogarithmic verification to be preserved through this implication.

The notion of semi-honest-verifier zero-knowledge means that the protocol leaks no information to an honest verifier for any choice of verifier randomness. This suffices for our main result, as we explain in Section 2.1. We also prove results which allow us to prove a variant of Theorem 2 with the stronger property of bounded-query zero-knowledge (with increased verifier time). These results are described in more detail in Section 2.8.

IOP	encode circuit cost	prover cost	verifier cost	query complexity	zero-knowledge
[BCGGHJ17]	$O(n)$ $\mathbb{F}$-ops	$O(n)$ $\mathbb{F}$-ops	$O(\sqrt{n})$ $\mathbb{F}$-ops	$O(\sqrt{n})$	semi-honest zk
[BCG20]	$O(n)$ $\mathbb{F}$-ops	$O(n)$ $\mathbb{F}$-ops	$O(n^\epsilon)$ $\mathbb{F}$-ops	$O(n^\epsilon)$	not zk
this work	$O(n)$ $\mathbb{F}$-ops	$O(n)$ $\mathbb{F}$-ops	$\mathsf{polylog}(n)$ $\mathbb{F}$-ops	$O(\log n)$	semi-honest zk

Figure 1: Comparison of known IOPs with a linear-time prover. The parameters are for an n-gate arithmetic circuit defined over a field $\mathbb{F}$ of size $\Omega(n)$; and ϵ is any positive constant. The sublinear verification in all cases is achieved in the holographic setting (the verifier has oracle access to an encoding of the circuit).

1.2 Related work

The zero-knowledge succinct argument of Kilian [Kil92] is obtained by combining a collision-resistant hash function and a probabilistically-checkable proof (PCP). If we instantiate Kilian's construction with PCPs with a quasilinear-time prover and a polylogarithmic query complexity [BS08; BCGT13] rather than the PCP used at the time [BFLS91], we obtain a zero-knowledge succinct argument whose prover runs in quasilinear-time.

In terms of asymptotic complexity of the prover, the above remains, to a first order, the state of the art for succinct arguments (zero knowledge or not). In particular, the question of constructing PCPs with a linear-time prover and a polylogarithmic query complexity remains a major open problem.

Reducing the quasilinear cost. Researchers in the last decade have invested significant effort into reducing prover time in succinct arguments, eliminating many logarithmic multiplicative factors from the prover's quasilinear time. Some works have come close to (but did not achieve) a linear time prover, as we explain.

Essentially all approaches for constructing succinct arguments follow this high-level template: first construct a probabilistic proof in some proof model, and then use cryptography to compile the probabilistic proof into an argument system. The first step alone typically costs more than linear time because it involves (among other things) using the Fast Fourier Transform (FFT) to encode the computation as a polynomial.

Several works [BCCGP16; BBBPWM18; WTSTW18; XZZPS19; Set20; ZWZZ20; SL20; KMP20] construct various forms of succinct arguments *without FFTs* by first constructing linear-time probabilistic proofs in certain "algebraic" models and then compiling these into arguments by using homomorphic commitments. However, the cryptography introduces quasilinear work for the prover,[5] usually to perform a linear number of multi-exponentiations over a cryptographically-large group (which translates to a quasilinear number of group operations for the prover).[6] In sum, this line of works has contributed among the best asymptotic prover times for succinct arguments, but the cryptography has precluded linear-time provers.

Linear-time prover and sublinear communication. Bootle et al. [BCGGHJ17] observe that Kilian's approach to succinct arguments introduces only linear cryptographic costs, when the collision-resistant hash function used for the compilation is suitably instantiated. (We elaborate on this in Section 2.1.) Prior work leveraged this observation to achieve argument systems with linear-time prover and sublinear communication:

- [BCGGHJ17] achieves an honest-verifier zero knowledge argument system for arithmetic circuit satisfiability with a communication complexity of $O(\sqrt{n})$, where the prover performs $O(n)$ field operations and hash computations while the verifier performs $O(\sqrt{n})$ field operations and hash computations.
- [BCG20] achieves, for every $\epsilon > 0$, an argument system for R1CS with a communication complexity of $O(n^\epsilon)$, where the prover performs $O(n)$ field operations and hash computations while the verifier performs $O(n^\epsilon)$ field operations and hash computations. No zero knowledge property is achieved in this work.

[5] The quasilinear costs in some works (due to cryptography [XZZPS19; ZWZZ20] or an FFT [ZXZS20]) scale with witness size rather than computation size, and so the prover runs in linear time when the witness is small relative to the computation.

[6] Some of the cited works still refer to such prover time as "linear" or "asymptotically optimal". This is a misnomer.

There are linear-time candidates for the hash function [AHIKV17], leading to a linear-time prover.

In both cases the technical core is the construction of IOPs with a linear-time prover, but prior work only achieved sublinear query complexity thereby, after compilation, falling short of the goal of polylogarithmic communication complexity. The main challenge to design IOPs with linear-time provers is that one cannot adopt "useful" codes like the Reed–Solomon code since the encoding time is quasilinear. Instead, one resorts to using linear-time encodable codes (e.g., of Spielman [Spi96] or Druk–Ishai [DI14]) that, unfortunately, do not have the multiplication property, which makes designing IOPs more difficult.

2 Techniques

Our result about zero-knowledge succinct arguments (Theorem 1) is a direct implication of our result about zero-knowledge probabilistic proofs (Theorem 2). We review this implication in Section 2.1, and then dedicate the rest of the technical overview to the main ideas behind Theorem 2: we sketch the steps in our proof in Section 2.2 and then summarize the main ideas behind each step in Sections 2.3 to 2.6. Results related to our variant of Theorem 2 with bounded-query zero-knowledge are described in more detail in Section 2.8.

2.1 Linear-time succinct arguments from linear-time probabilistic proofs

Known approaches for constructing succinct arguments rely on cryptography to "compile" various forms of probabilistic proofs into argument systems. However, the cryptography used typically introduces super-linear overheads, ruling out a linear-time argument system even when compiling a linear-time probabilistic proof. Bootle et al. [BCGGHJ17] observe that Kilian's approach [Kil92] is a notable exception. We review this below because Theorem 2 implies Theorem 1 via this approach; our technical contribution is Theorem 2.

Linear-time arguments via Kilian's approach. The cryptography used in Kilian's approach is collision-resistant hash functions, for which there are linear-time candidates under standard assumptions (e.g., based on the hardness of finding short codewords in linear codes [AHIKV17]). If we use a linear-time hash function in Kilian's approach to compile a linear-time PCP (over a large-enough alphabet) then we obtain a linear-time argument system.[7] While constructions of linear-time PCPs are not known (and seem far beyond current techniques), the foregoing implication equally holds for IOPs [BCS16; RRR16], a multi-round relaxation of PCPs. This route was used in [BCGGHJ17; BCG20] to obtain sublinear-sized arguments with linear-time prover and -time verifier from IOPs with linear-time prover and sublinear-time verifier (see Figure 1). These arguments were "stuck" in the sublinear regime because the underlying IOPs were also in that regime.

In this paper we take the same route, but significantly improve on the prior state of the art for IOPs. Linear-time IOPs with polylogarithmic query complexity lead to linear-time arguments with polylogarithmic communication; moreover, if the IOP verifier runs in polylogarithmic time then so does the argument verifier.

The case of zero knowledge. The IOPs in Theorem 2 additionally satisfy a strong notion of zero knowledge, which enables us to obtain linear-time succinct arguments that are zero knowledge, as explained below.

Kilian's approach to additionally achieve zero knowledge makes a non-black-box use of the collision-resistant hash function and the probabilistic proof's verifier,[8] which is undesirable. Ishai et al. [IMSX15] proved that if the underlying probabilistic proof satisfies a mild notion of zero knowledge then Kilian's approach can be simplified to yield a zero-knowledge succinct argument where both collision-resistant hash function and the probabilistic proof are used as black boxes. This implication, too, preserves linear time of both building blocks to yield a zero-knowledge succinct argument with a linear-time prover.

The notion of zero-knowledge required of the underlying probabilistic proof depends on the desired notion of zero knowledge for the argument system. If the argument system is desired to be honest-verifier zero knowledge (this suffices, e.g., to subsequently apply the Fiat-Shamir heuristic) then the probabilistic proof must be honest-verifier zero knowledge. If instead the argument system is desired to be malicious-verifier zero-knowledge then the probabilistic proof must be *semi*-honest-verifier zero knowledge (the simulator works

[7] In Kilian's approach, the argument prover's cryptographic cost is dominated by the cost to commit to the PCP string via a Merkle tree. In particular, if the PCP has proof length l and the size of a proof symbol is linear in the input size of the hash function, then the running time of the argument prover is within a constant of the running time of the PCP prover.

[8] Modify the Merkle tree to be over hiding commitments of proof symbols (rather than over the proof symbols themselves) and then prove in zero knowledge that opening the queried locations would have made the probabilistic proof verifier accept.

for any possible fixed execution of the honest verifier). A further strengthening known as bounded-query zero knowledge, the hiding notion typically studied for PCPs [KPT97], enables reductions in communication.

2.2 Proof overview for Theorem 2

Having explained how Theorem 2 implies Theorem 1 via known implications, we now turn our attention to explaining our proof of Theorem 2, which is a new IOP construction. Below we summarize the steps in our proof approach; we provide further intuition about these steps in subsequent subsections.

Recall that an IOP is a proof model in which a prover and a verifier interact over multiple rounds, and in each round the prover sends a proof message and the verifier replies with a challenge message. The verifier has query access to all received proof messages, in the sense that it can query any of the proof messages at any desired location. The verifier decides to accept or reject depending on its input, its randomness, and answers to its queries. The main information-theoretic efficiency measures in an IOP are proof length (total size of all proof messages) and query complexity (number of read locations across all proof messages), while the main computational efficiency measures are prover running time and verifier running time.

Our starting point. The starting point of our work is the non-zero-knowledge IOP for R1CS in [BCG20], which has prover time $O(m)$ and verifier time $O(m^\epsilon)$ for any a-priori fixed constant $\epsilon > 0$. That IOP is obtained in two steps: first construct a *tensor IOP* for R1CS with linear prover time and constant query complexity; second apply a compiler that transforms any tensor IOP into a standard IOP. In a tensor IOP, the verifier may make multiple *tensor queries* directly to a proof message Π, each of the form $q = (q_1, \ldots, q_t)$, receiving the answers $v := \langle \otimes_i q_i, \Pi \rangle$. This differs from the usual setting where the verifier queries single locations of proof string values (i.e., makes *point queries*).

Steps in our proof. Our technical contribution is to reduce query complexity and verification time from sublinear to polylogarithmic (an exponential improvement) while preserving the linear time of the prover and also additionally achieving zero knowledge. This requires overcoming several technical challenges, across several steps that are sketched below (and displayed diagrammatically in Figure 2).

- *Tensor IOP for R1CS with zero knowledge.* We modify the tensor IOP for R1CS in [BCG20] to additionally achieve zero knowledge while preserving all of its efficiency parameters, including a linear prover time, constant query complexity, and logarithmic verifier time (as a holographic proof).

- *From tensor IOPs to standard IOPs while preserving zero knowledge.* We present a tensor-query to point-query compiler that is more efficient than the one in [BCG20] (smaller query complexity and verifier time) and additionally preserves zero knowledge. The efficiency improvements come from using proof composition techniques. To preserve the linear complexity and zero-knowledge properties of the given tensor IOP, the tensor codes used in our compiler must be linear-time encodable and satisfy a zero-knowledge property, whereby codewords do not reveal any information about the underlying message under adaptive queries below a fixed query bound. We review this property in Section 2.5.

  We establish structural properties of zero-knowledge codes and prove that they are preserved under tensor products. Thus, it suffices to find a linear-time encodable zero-knowledge code to act as the base of the tensor product code.

- *Constructing a base code with zero-knowledge.* To prove our main theorem, we use an explicit construction of zero-knowledge code based on [Spi96] codes, which protect against a single malicious query. This is enough to prove zero-knowledge against semi-honest verifiers in Theorem 2, which suffices for our main theorem. We also give a probabilistic construction of zero-knowledge codes based on [DI14] codes which

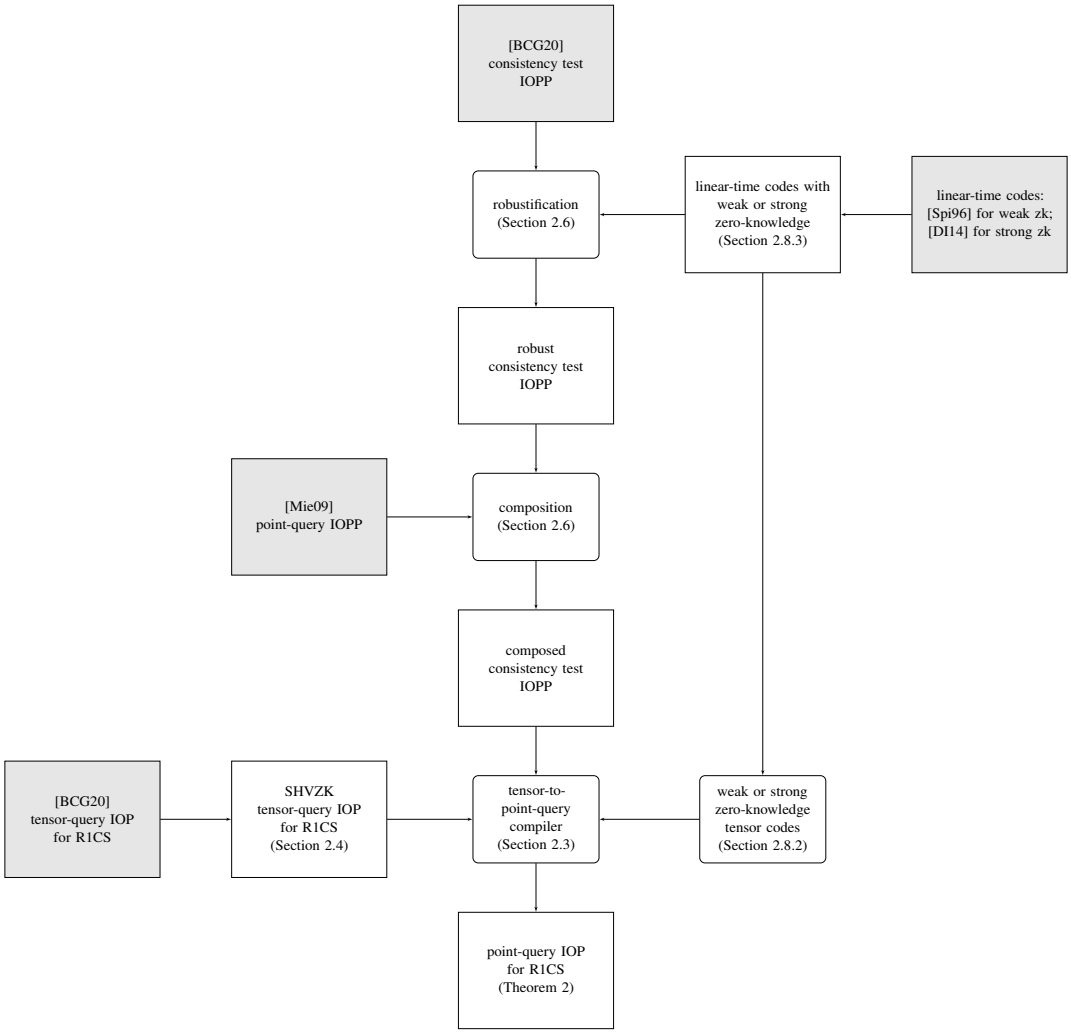

Figure 2: Our approach to proving Theorem 2.

do not reveal information on the underlying message even when the verifier makes queries to a constant fraction of codeword entries. This allows us to prove a variation of Theorem 2 with the stronger property of *bounded-query zero-knowledge*. We describe this in more detail in Section 2.8.

- *Interactive proof composition.* Proof composition [AS98] combines an outer proof system and an inner proof system to obtain a new proof system that (roughly) has the prover and verifier complexity of the outer proof system, and the query complexity of the inner proof system. The outer proof system must be *robust* and the inner proof system must be a *proximity proof*. We use interactive proof composition of IOPs [BCGRS17] to achieve the efficiency of our compiler: the outer proof system is a robustification of the tensor-query consistency check of [BCG20] and the inner proof system is the PCP of proximity of [Mie09].

2.3 From tensor-queries to point-queries in zero-knowledge

We generically transform any tensor-query IOP into a corresponding point-query IOP, *while preserving zero knowledge*. In more detail, the transformation is parametrized by a zero-knowledge linear code (a notion explained in more detail in Section 2.5) and outputs a point-query IOP that is bounded-query zero knowledge, meaning that malicious queries up to a fixed query bound do not leak any information. In contrast, the tensor-query IOP being transformed is required to only satisfy a weaker notion of zero knowledge, called *semi-honest*-verifier zero knowledge, that we describe further below.

Theorem 3 (informal)**.** *There is an efficient transformation that takes as input a tensor-query IOP and a linear code, and outputs a point-query IOP that has related complexity parameters, as summarized below.*

- Input IOP: *an $(\mathbb{F}, k, t)$-tensor IOP for a relation R with soundness error ϵ, round complexity* rc*, proof length* l*, query complexity* q*, prover arithmetic complexity* tp*, and verifier arithmetic complexity* tv*.*
- Input code: *a linear code $\mathcal{C}$ over $\mathbb{F}$ with rate $\rho = \frac{k}{n}$, relative distance $\delta = \frac{d}{n}$, encoding time $\theta(k) \cdot k$ and description size $|\mathcal{C}|$.*
- Output IOP: *a point-query IOP with soundness error $O_{\delta,t}(\epsilon) + O(d^t/|\mathbb{F}|)$, round complexity $O_t(\mathsf{rc})$, proof length $O_{\rho,t}(\mathsf{q} \cdot \mathsf{l})$, query complexity $O_t(\mathsf{rc})$, prover arithmetic complexity $\mathsf{tp} + O_{\rho,t}(\mathsf{q} \cdot \mathsf{l}) \cdot \theta(k) + \mathsf{poly}(|\mathcal{C}|, t, \mathsf{q}, k)$, and verifier arithmetic complexity $\mathsf{tv} + \mathsf{poly}(|\mathcal{C}|, t, \mathsf{q}, \log k)$.*

Moreover, when the tensor-query IOP is semi-honest-verifier zero knowledge

- *if the code $\mathcal{C}$ is 1-query zero-knowledge, then the point-query IOP is semi-honest-verifier zero knowledge;*
- *if the code $\mathcal{C}$ is b-query zero-knowledge, then the point-query IOP is b-query zero knowledge.*

Finally, the transformation preserves holography up to the multiplicative encoding overhead θ of $\mathcal{C}$ and terms that depend on ρ and t.

The formal statement and its proof are in Section 8.[9] We now explain the main ideas behind our compiler.

Starting point: an inefficient compiler that breaks zero knowledge. Our starting point is the code-based compiler of [BCG20], which takes as input a tensor-query IOP $(\mathbf{P}, \mathbf{V})$ and a linear error-correcting code $\mathcal{C}$ and produces a corresponding point-query IOP $(\hat{\mathbf{P}}, \hat{\mathbf{V}})$. We briefly summarize how the compiler works.

First, the point-query IOP simulates the tensor-query IOP with the modification that: (i) each proof oracle Π is replaced by its encoding $\hat{\Pi}$ using the tensor product code $\mathcal{C}^{\otimes t}$; (ii) instead of making tensor queries

[9] The result in Section 8 (Lemma 8.2) is stated with an additional generic input: an IOP of proximity for a certain tensor-query relation. The result stated here is obtained by using a new IOP of proximity obtained by applying proof composition techniques to results from [BCG20] (see Section 9.1 and Section 9.2).

to the proof oracles directly, the new verifier $\hat{\mathbf{V}}$ sends tensor queries $q^{(s)}$ to the provers, who replies with the answers v_s. Second, the new prover $\hat{\mathbf{P}}$ and new verifier $\hat{\mathbf{V}}$ engage in a *consistency test* subprotocol to ensure that the answers v_s (which may have been computed dishonestly) are consistent with the proofs Π. The consistency test incorporates a *proximity test* to make sure that each proof message $\hat{\Pi}$ is close to a valid encoding of some proof message Π (as a malicious prover may send messages which are far from $\mathcal{C}^{\otimes t}$). As part of the consistency check, the prover sends the verifier "folded" proof messages $c_j^{(s)} = \langle \otimes_{i \le j} q_i^{(s)}, \Pi \rangle$ encoded under lower-dimensional tensor codes $\mathcal{C}^{\otimes t-j}$. The proximity test works similarly, using random linear combinations v_i of length k sampled by the verifier instead of structured tensor queries. In both cases, the verifier checks linear relations between successive encodings c_j and c_{j+1} by making $O(k)$ point queries.

This compiler preserves prover time up to the encoding overhead $\theta(k)$ as in our Theorem 3, but has two major shortcomings. The compiler does not preserve zero-knowledge, even if the tensor IOP to be compiled is zero knowledge. Moreover, the output IOP has query complexity $O(k)$ and verifier complexity $O(k)$, which does not suffice for Theorem 3.

Below we elaborate on how we overcome these shortcomings. We begin by explaining semi-honest verifier zero knowledge, the property for the tensor IOP to achieve zero knowledge for the output IOP.

Semi-honest-verifier zero knowledge. Here, "semi-honest" means that there exists a simulator that (perfectly) simulates the honest verifier's view for any fixed choice of the honest verifier's randomness.[10] This requirement is stronger than honest-verifier zero-knowledge, where the simulator must simulate the honest verifier's view for a random choice of its randomness; also, this requirement is weaker than the standard definition of zero-knowledge for IOPs, in which the verifier may deviate from the protocol and make arbitrary queries to the received oracles up to some query bound. Nevertheless, this notion suffices for our compilation procedure, which will produce point-query IOPs with zero-knowledge against semi-honest verifiers and verifiers making a bounded number of point queries.

Approach for zero knowledge. We need to ensure that, in our compiler, if the tensor-query IOP given as input is semi-honest-verifier zero knowledge then, depending on the zero knowledge property of the code $\mathcal{C}$, the output point-query IOP is either semi-honest verifier zero knowledge or bounded-query zero knowledge. This implication does not hold for the compiler of [BCG20] because, when using a (non-zero-knowledge) linear code $\mathcal{C}$, a point query to any encoding $\hat{\Pi}$ or c_j leaks information about Π. We address the information leaked by the encoded proof messages $\hat{\Pi}$ and the folded proof messages $c_j^{(s)}$ in two ways.

We ensure that the folded proof messages $c_j^{(s)}$ do not leak any information by leveraging the fact that the consistency check of [BCG20] is about a linear relation, and thus can be invoked on a random shift of the instance of interest. In more detail, the usual approach to making the messages in such a subprotocol zero-knowledge is to mask the input message as $f = \gamma\Pi + \Xi$, where Ξ is a random message sent by the prover and γ is a random challenge sent by the verifier after that, and then run the consistency test on the *encoding* $c = \gamma\hat{\Pi} + \hat{\Xi}$. (The claimed tensor-query answers v_s need to be adjusted accordingly too to account for the contribution of Ξ). Informally, this enables the simulator to randomly sample c and honestly run the [BCG20] consistency check protocol. Queries on the resulting messages $c_j^{(s)}$ do not reveal any information, since they are derived from c, which is a random tensor codeword. Further, we do not require any zero-knowledge properties from the consistency check.

The simulator must still simulate the answers to point queries on Ξ by querying $\hat{\Pi}$ instead. To avoid information leakage from the encoded proofs $\hat{\Pi}$, we use a linear code $\mathcal{C}$ with bounded-query zero-knowledge. This is similar to the notion for IOPs, and means that queries to a codeword up to a fixed query bound do not leak any information. The [BCG20] compiler uses tensor products of codes, and to achieve semi-honest-

[10] This is related to special honest-verifier zero-knowledge for sigma protocols.

verifier zero knowledge for the output IOP, it is important that the tensor product code $\mathcal{C}^{\otimes(t-1)}$ is 1-query zero-knowledge. Furthermore, to achieve b-query zero knowledge for the output IOP, it is important that the tensor product code $\mathcal{C}^{\otimes t}$ is also zero-knowledge against b queries. This leads to the problem of finding a zero-knowledge code which is encodable in linear time,[11] which we discuss in Section 2.8.3, and showing that the zero-knowledge property of codes is preserved under tensor products, which we discuss in Section 2.8.2.

Approach for efficiency. Our modifications to the compiler of [BCG20] to preserve zero-knowledge do not affect its efficiency; in particular, if the zero-knowledge code $\mathcal{C}^{\otimes t}$ has a linear-time encoding, then compiler preserves linear arithmetic complexity of the prover. We are left to reduce the query complexity and the verifier complexity from $O(k)$ to polylogarithmic in k.

First, we discuss query complexity. By first strengthening the consistency test through robustification, and then using proof composition techniques, we can reduce the query complexity so that it is independent of k. Further details are given in Section 2.6.

Second, we discuss verifier complexity. After proof composition, the new verifier's arithmetic complexity is polynomial in the *description size* of the computations performed by the original consistency-test verifier. In this case, the description consists of the *verifier randomness* used in the protocol, and to generate the error-correcting code $\mathcal{C}$. The randomness complexity of the [BCG20] compiler is $O(\mathsf{q} \cdot k \cdot t)$, due to the random linear combinations v_i used in the proximity test. Fortunately, these linear combinations can be derandomized, as we now explain. The v_i are used in the soundness analysis of the [BCG20] proximity as part of a "distortion statement": if any member of a collection of messages is far (in Hamming distance) from a linear code, then a random linear combination of those messages is also far from the code, except with some small, bounded, failure probability. Recent work [BSCIKS20] proves distortion statements for linear combinations of the form $v = (\alpha_i^1, \alpha_i^2, \alpha_i^3, \ldots, \alpha_i^{|v_i|})$ for a uniformly random $\alpha_i \leftarrow \mathbb{F}$, at the cost of a tolerable increased in failure probability, and thus, in the soundness error of the proximity test. This allows us to dramatically reduce the number of random field elements used in the proximity test from $O(\mathsf{q} \cdot k \cdot t)$ to $O(\mathsf{q} \cdot t)$, which suffices for Theorem 3.

2.4 Tensor IOP for R1CS with semi-honest verifier zero knowledge

The input to the compiler in Section 2.3 is a tensor IOP for R1CS that is semi-honest-verifier zero knowledge.

Theorem 4 (informal)**.** *For every finite field $\mathbb{F}$ and positive integers $k, t \in \mathbb{N}$, there is a $(\mathbb{F}, k, t)$-tensor holographic IOP for the indexed relation R_{R1CS}, which is semi-honest-verifier zero-knowledge, that supports instances over $\mathbb{F}$ with $m = O(k^t)$, that has the following parameters: (1) soundness error is $O(\frac{m}{|\mathbb{F}|})$; (2) round complexity is $O(\log m)$; (3) proof length is $O(m)$ elements in $\mathbb{F}$; (4) query complexity is $O(1)$; (5) the indexer and prover use $O(m)$ field operations; (6) the verifier uses $O(|x| + \log m)$ field operations.*

We prove this theorem in Section 4, and below we summarize the proof.

Our starting point is the holographic tensor IOP for R1CS in [BCG20], which achieves the same parameters as in the above theorem[12] except that it is not zero knowledge. We apply re-randomization techniques to modify their construction to achieve zero knowledge against semi-honest verifiers, while preserving all efficiency parameters. We now elaborate on this: first we review the structure of the tensor IOP in [BCG20], and then explain how we modify it for zero knowledge.

The holographic tensor IOP of BCG. The holographic tensor IOP for R1CS in [BCG20] follows a standard blueprint for constructing protocols for R1CS [BCRSVW19], adapted to the case of tensor queries. The

[11] Note also that query bound b must be at least the number of queries that $\hat{\mathbf{V}}$ makes to the encoded proof $\hat{\Pi}$.

[12] We reduce the verifier complexity by generating the verifier's tensor queries using a short seed.

prover first sends oracles containing the full assignment $z = (x, w)$ and its linear combinations $z_A := Az$, $z_B := Bz$, and $z_C := Cz$. The verifier wishes to check that $z_A \circ z_B = z_C$ and that z_A, z_B, z_C are the correct linear combinations of z. To facilitate this, the verifier sends some randomness to the prover, which enables reducing the first condition (a Hadamard product) to a scalar-product condition. The verifier then engages with the prover in scalar-product subprotocols for checking the scalar products, and holographic "lincheck" subprotocols for checking the linear relations (given tensor-query access to suitable linear-time encodings of the matrices A, B, C). This leads the verifier to make a constant number of tensor queries to each of z, z_A, z_B, z_C for concluding the subprotocols and performing other consistency checks (e.g., consistency of z with the public input x).

This protocol is not zero knowledge even for an honest verifier because: (1) the answer to each tensor query to z, z_A, z_B, z_C reveals information about the secret input w (part of the full assignment z); (2) messages sent by the prover during the scalar-product and lincheck protocols reveal further information about z, z_A, z_B, z_C.

Approach for zero knowledge. We need to ensure that every prover message and the answer to every tensor query is simulatable. The fact that queries are linear combinations with a tensor structure would make this rather difficult if we had to deal with malicious verifiers.[13] Fortunately, we seek zero knowledge against semi-honest verifiers only, which means that it suffices to consider any valid execution of an honest verifier, and in particular we have the freedom to assume that the verifier's queries have a certain structure. While there are generic techniques for related settings (e.g., a transformation for linear PCPs with degree-2 verifiers in [BCIOP13]), they do not seem to be useful for our setting (tensor IOPs with linear-time proving). So our approach here will be to directly modify the protocol in [BCG20] by adapting ideas used in prior works.

We incorporate random values into the protocol in two different ways to address the two types of leakage above. This will enable us to make every prover message and query answer either uniformly random (independent of the witness) or uniquely determined by other prover messages or query answers. The simulator that we construct will then simply sample all the random values and derive the rest from them. We elaborate on this strategy in the paragraphs below.

(1) ZK against verifier queries. The answer to each verifier query is a linear combination (with tensor structure) of elements in the prover's oracle message. Intuitively, if we pad each oracle message with as many random values as the number of queries it receives, and also "force" the linear combination to have non-zero coefficients in the padded region, then all the query answers will be uniformly random and reveal no information. Padding each of z, z_A, z_B, z_C with independent randomness, however, does not preserve completeness because then the padded vectors would not satisfy the R1CS condition.

This naive strategy, however, can be fixed in a straightforward way. We rely on a small R1CS gadget, whose solutions can be efficiently sampled, for which we can control the amount of independent randomness. Then we augment the original R1CS instance with this gadget.[14] This means that, in the first step of the protocol, the prover samples a random solution to the R1CS gadget and appends it to the witness to obtain an augmented witness.[15] In the remainder of the protocol, the random solution acts as padding as described above, while preserving completeness. The choice of how much to pad is made depending on how many

[13] For example, constructing linear PCPs that are zero knowledge against malicious verifiers remains an open problem. Constructing tensor IOPs that are zero knowledge against malicious verifiers, while formally an easier question, appears similarly hard.

[14] This is distinct from how zero knowledge is achieved for prior IOPs for R1CS based on the Reed–Solomon code [BCRSVW19]. Instead, it is closer in spirit to how semi-honest-verifier zero knowledge was achieved for linear PCPs for circuits or quadratic arithmetic programs in [GGPR13; BCIOP13].

[15] We stress that this modification achieves zero knowledge only against semi-honest verifiers, because a malicious verifier could choose to query the padded vectors with a linear combination that leaves out the randomness and thereby learns information about the secret witness. Nevertheless, as discussed in Section 2.3, a tensor IOP that is merely semi-honest-verifier zero knowledge suffices for obtaining a point-query IOP with zero knowledge against bounded-query malicious verifiers.

independent queries each oracle receives.

(2) ZK for the subprotocols. Each lincheck subprotocol checks a linear relation $Uz = z_U$, and as with the point-query compiler, the usual approach to making the messages in such a subprotocol zero-knowledge is to run the subprotocol on the input vector $e = \gamma z + w$, where w is a random vector sent by the prover and γ is a random challenge sent by the verifier after that. (The claimed output vector z_U needs to be adjusted accordingly too). Informally, this enables the simulator to randomly sample e and honestly run the lincheck protocol, which reveals no information, since the honest verifier only makes queries to $e = \gamma z + w$, and never to v and w separately. Since the lincheck subprotocol is used as a black-box, the holographic properties of our protocol are unaffected by our modifications for zero-knowledge, and are inherited directly from the lincheck protocol of [BCG20].

The scalar-product subprotocol is a sumcheck protocol on a certain polynomial p. Sumcheck protocols are usually made zero knowledge by following a similar pattern and running the sumcheck protocol on the polynomial $u := \gamma p + q$, where q is a random polynomial [BCFGRS17]. The simulator can randomly sample u and honestly run the sumcheck protocol, while simulating answers to q by querying p instead.

Applying this idea in our setting of linear-time prover requires some care. In the protocol of [BCG20], the polynomial p is the product of two multilinear polynomials f and g, each with $\log n$ variables (and thus $O(n)$ coefficients). To achieve linear arithmetic complexity for the prover, it is crucial that the prover *does not compute the sumcheck directly on* p, which could have up to $O(n^2)$ coefficients, and works only with f and g following a certain linear-time algorithm [Tha13]. Thus the prover cannot simply sample a random q.

The solution is to re-randomize the multiplicands f and g separately to $\gamma f + r$ and $\gamma g + s$, and run the sumcheck protocol on their product $(\gamma f + r)(\gamma g + s)$. (If $p = f \cdot g$ sums to α then $(\gamma f + r)(\gamma g + s)$ sums to $\alpha\gamma^2 + \rho\gamma + \sigma$ for some ρ and σ derived from r and s alone.) The prover can then compute on polynomials with $O(n)$ coefficients, and the simulator can sample each factor of p at random and proceed similarly.

Efficiency. The resulting tensor IOP inherits all efficiency parameters of the non-ZK tensor IOP of [BCG20]: soundness error $O(m/|\mathbb{F}|)$; logarithmic round complexity; linear proof length; constant query complexity; linear-time indexer; linear-time prover; and logarithmic-time verifier.

2.5 Hiding properties of linear codes

Linear codes have been used to achieve hiding properties in many applications, including secret sharing, multi-party computation, and probabilistic proofs. Below we introduce useful notation and then review the properties of linear codes that we use, along with other ingredients, to achieve zero knowledge IOPs. Informally, we consider probabilistic encodings for linear codes with the property that a small number of locations of a codeword reveal no information about the underlying encoded message.

Randomized linear codes. Let $\mathcal{C}$ be a linear code over a field $\mathbb{F}$ with message length k and block length n, and let $\mathrm{Enc}\colon \mathbb{F}^k \to \mathbb{F}^n$ be an encoding function for $\mathcal{C}$ (that is, $\mathrm{Enc}(\mathbb{F}^k) = \mathcal{C}$). For a fixed choice of k_m and k_r such that $k_m + k_r = k$, we can derive from Enc the bivariate function $\tilde{\mathrm{Enc}}\colon \mathbb{F}^{k_m} \times \mathbb{F}^{k_r} \to \mathbb{F}^n$ defined as $\tilde{\mathrm{Enc}}(m; r) := \mathrm{Enc}(m \| r)$. In turn, this function naturally induces a probabilistic encoding: we define $\tilde{\mathrm{Enc}}(m)$ to be the random variable $\{\tilde{\mathrm{Enc}}(m; r)\}_{r \leftarrow \mathbb{F}^{k_r}}$. In other words, we have designated the first k_m inputs of Enc for the message and the remaining k_r inputs for encoding randomness. We shall refer to a code $\mathcal{C}$ specified via a bivariate function $\tilde{\mathrm{Enc}}$ as a *randomized linear code*.

Bounded-query zero knowledge. A randomized linear code is *b-query zero knowledge* if reading any b locations of a random encoding of a message does not reveal any information about the message. The locations may be chosen arbitrarily and adaptively. In more detail, we denote by $\mathrm{View}(\tilde{\mathrm{Enc}}(m; r), A)$ the view of an oracle algorithm A that is given query access to the codeword $\tilde{\mathrm{Enc}}(m; r)$. We say that $\mathcal{C}$ is b-query

zero knowledge if there exists a $\mathsf{poly}(n, \log|\mathbb{F}|)$-time simulator algorithm $\mathcal{S}$ such that, for every message $m \in \mathbb{F}^{k_m}$ and b-query algorithm A, the following random variables are identically distributed:

$$\left\{\mathrm{View}\big(\tilde{\mathrm{Enc}}(m;r), A\big)\right\}_{r \leftarrow \mathbb{F}^{k_r}} \quad \text{and} \quad \mathcal{S}^A \ .$$

To achieve even 1-query zero knowledge the random encoding cannot be systematic (as otherwise the algorithm A could learn any location of the message by querying the corresponding location in the codeword).

The above notion mirrors the standard notion of bounded-query zero knowledge for several models of probabilistic proofs (PCPs [KPT97; IMS12; IMSX15], IPCPs [GIMS10], and IOPs [BCGV16; BCFGRS17]). Moreover, it is equivalent, in the special case of codes with a polynomial-time encoding, to the message-indistinguishability definition of zero knowledge of [ISVW13] (which requires that the encodings of any two messages are equidistributed when restricted to any small-enough subset of coordinates); see Appendix C.

Bounded-query uniformity. In intermediate steps we also consider a stronger notion of zero knowledge: we say that $\mathcal{C}$ is *b-query uniform* if any b locations of $\{\tilde{\mathrm{Enc}}(m;r)\}_{r \leftarrow \mathbb{F}^{k_r}}$ are uniformly random and independent symbols. This is a strengthening over the prior notion because the simulator for this case is a simple fixed strategy: answer each query with a freshly sampled random symbol. We refer the reader to Example 1 for more intuition on the difference between the two notions; there we explain how code concatenation, a standard operation on codes, naturally leads to codes that are bounded-query zero knowledge but not bounded-query uniform, and in particular the simulator cannot employ the foregoing simple strategy.

Example: Reed–Solomon code. The Reed–Solomon code is a well-known code whose hiding properties are well understood. Namely, one can achieve b-query uniformity (and thus also b-query zero knowledge) by interpolating the given message padded with b random elements and then evaluating the resulting polynomial on a domain disjoint from the interpolation domain. We explore this example further in Section 5.3.1 to build intuition about algebraic statements proved in this paper (and discussed in Section 2.8.1). The Reed–Solomon code happens to also be a versatile tool to construct efficient IOPs, and indeed many IOPs rely on the Reed–Solomon code to additionally achieve bounded-query zero knowledge [BCGV16; BCFGRS17; BBHR19; AHIV17; BCRSVW19; COS20]. In this paper, however, we will not use the Reed–Solomon code in our constructions because its encoding function costs more than linear time.

2.6 Interactive proof composition

Our compiler from tensor-query to point-query IOPs uses the consistency check protocol from [BCG20]. When applied to our $(\mathbb{F}, k, t)$-tensor IOP with $n = \Theta(k^t)$, the consistency check has query complexity $O(n^{1/t})$, prover arithmetic complexity $O(n)$ and verifier arithmetic complexity $O(n^{1/t})$. Though the query complexity and verifier complexity can be improved by increasing the value of t, they remain sublinear in n. Thus, the [BCG20] consistency check does not suffice to prove Theorem 3.

We apply interactive proof composition for IOPs [BCGRS17] to reduce both the query complexity and the verifier complexity, without increasing the prover arithmetic complexity above linear time. (It remains an interesting question whether one can also achieve proof length that approaches witness length, the efficiency goal studied in [RR20] via related techniques.)

Interactive proof composition involves an "outer" IOP that is robust[16] and is for the desired relation, and an "inner" IOP of proximity that is for a relation about outer IOP's verifier. At a high level, we wish to apply this with the consistency check of [BCG20] as the outer IOP and the PCP of proximity of [Mie09] as the inner IOP. This requires some care, in part because the consistency check of [BCG20] is not robust, and also

[16]This means that with high probability over the verifier's randomness, a local view of the verifier is far from accepting.

because our target parameters do not leave much wiggle room. We elaborate on how we robustify the outer protocol, and how we perform the composition below.

Robustification. Any IOP can be generically robustified by encoding each proof symbol in every round via an appropriate error-correcting code: if the IOP has query complexity q then this transformation yields a robustness parameter $\alpha = O(1/\mathsf{q})$ (over the alphabet of the code). This is a straightforward generalization of robustifications for IPs (each prover message in each round is encoded) and for PCPs (each proof symbol of the PCP is encoded). This also extends to robustifying IOPPs, in which case each symbol of the witness whose proximity is being proved is also encoded (and this modifies the relation proved by the IOPP slightly). For convenience, we state and give a proof sketch for this generic robustification in Appendix A.

Superficially, this robustification seems insufficient to prove Theorem 3 because the IOPP of [BCG20] has sublinear query complexity $\mathsf{q} = O(n^{1/t})$, which would lead to a robustness parameter that is sub-constant. However, the queries in [BCG20] are *bundled*: the verifier always queries entire sets of $O(n^{1/t})$ locations, so the IOPP can be restated as a *constant-query* IOPP over the large alphabet $\mathbb{F}^{O(n^{1/t})}$. To robustify an IOPP over such a large alphabet, we need to use a code with a linear-time encoding such as [Spi96] (here knowledge codes are not essential) in order to preserve the linear complexity of the prover. This gives us an IOPP for tensor queries with prover complexity $O(n)$, verifier complexity $O(n^{1/t})$, query complexity $O(n^{1/t})$ over the alphabet $\mathbb{F}$, constant soundness error, and, most importantly, a *constant robustness parameter* α. Furthermore, the verifier randomness complexity, which affects the verifier arithmetic complexity in the composed proof, is unchanged by robustification. We are now ready for the next step, proof composition, which will enable us to reduce query complexity and also verifier complexity.

Composition. Interactive proof composition [BCGRS17] applies to any outer IOP that is robust and inner IOP that is a proof of proximity. If the outer IOP is a proof of proximity (as is the case when using the IOPP obtained above) then the composed IOP is also a proof of proximity; similarly, if the inner IOP is robust then the composed IOP is also robust. For convenience, we state this generic proof composition in Appendix B.

We apply proof composition as follows: (i) the outer proof system is the robust IOPP for tensor queries obtained above; (ii) the inner proof system is the PCP of proximity for $\mathsf{NTIME}(T)$ due to Mie [Mie09]. Recall that the latter achieves any desired constant soundness error and constant robustness parameter, with proof length $\tilde{O}(T(|\mathbb{x}|))$, query complexity $O(1)$, prover time $\mathsf{poly}(T(|\mathbb{x}|))$, and verifier time $\mathsf{poly}(|\mathbb{x}|, \log T(|\mathbb{x}|))$.

Informally, the new verifier in the composed proof system runs the interactive phase of the IOPP for tensor queries and then, rather than running the query phase of the outer IOPP, runs the PCPP verifier of [Mie09] to check that the witness is close to a tensor encoding of a message that is consistent with all the answers to the tensor queries. This reduces the query complexity from $O(n^{1/t})$ to $O(1)$ queries.

Next we discuss prover complexity and verifier complexity for the composed proof system.

The cost of the prover in the composed proof system is $O(n)$ field operations to run the prover of the robust IOPP for tensor queries plus $\mathsf{poly}(T(|\mathbb{x}|))$ bit operations to run the PCPP prover in [Mie09]. In our case, the $\mathsf{NTIME}(T)$ relation being checked is the decision predicate for the verifier in the robust IOPP, so that $T = O(n^{1/t})$ (times smaller factors depending on $\log |\mathbb{F}|$ since T refers to bit operations rather than field operations). If we take the tensor power $t \in \mathbb{N}$ to be a sufficiently large constant, then we can ensure that the cost of the prover in the composed proof system is dominated by $O(n)$ field operations.

The cost of the verifier in the composed proof system is dominated by $\mathsf{poly}(|\mathbb{x}|, \log T(|\mathbb{x}|))$ bit operations, the time to run the PCPP verifier in [Mie09]. From the previous paragraph, we know that $\log T(|\mathbb{x}|) = O(\log n)$. We are thus left to discuss $|\mathbb{x}|$. Here $\mathbb{x}$ is the state used to *describe* (not run) the computation of the decision predicate for the verifier in the robust IOPP. Hence, $|\mathbb{x}|$ is dominated by: (a) the description of the code $\mathcal{C}$ used for the tensor encoding; (b) the description of the tensor queries whose answers are being checked; and (c) the randomness complexity of the verifier for the robust IOPP. The last term can, via the

derandomization discussed in Section 2.3, be reduced to $O(t)$, a low-order term. The second term depends on the tensor queries, but for simplicity here we will ignore it because in our application all the tensor queries can be described via $O(t \log n)$ elements, again a low-order term. The first term depends on the choice of code $\mathcal{C}$, so we keep it as a parameter. In sum, the cost of the verifier in the composed system is $\mathsf{poly}(|\mathcal{C}|, \log n)$ bit operations.

Remark 2.1 (zero knowledge)**.** We do not require the ingredient discussed above (the IOPP for tensor queries) to be zero knowledge because we can invoke it on a random instance as explained in Section 2.3. Nevertheless, we believe that future improvements in zero knowledge IOPs (especially with a focus on concrete efficiency) will benefit from applying the transformations of robustification and proof composition to zero knowledge protocols. In such a case, it will be useful to understand how zero knowledge is affected by these transformations: *If the robustified IOP is required to be zero knowledge, what should we require of the given IOP and code used to encode each symbol? Also, if the composed IOP is required to be zero knowledge, what should we require of the outer IOP and inner IOP?* We took the opportunity to investigate this in our appendices, taking advantage of the fact that we already had to specify the relevant transformations.

We consider two target notions of zero knowledge: against semi-honest verifiers and against (malicious) bounded-query verifiers. Then we deduce natural conditions on the ingredients to robustification and proof composition that suffice to achieve the target notion of zero knowledge. See Lemma A.4 for the zero knowledge properties of robustification, and Lemma B.6 for the zero knowledge properties of proof composition. We view these results as an independent contribution to foundational transformations of IOPs.

2.7 Completing the proof of Theorem 2

We have described all of the ingredients for Theorem 2. We briefly summarise how they are combined.

To prove Theorem 2, one applies the compiler in Section 2.3, with the semi-honest-verifier zero-knowledge tensor-query IOP for relation R_{R1CS} from Theorem 4, and a family of *explicit* linear-time 1-query zero-knowledge codes as inputs. The family of codes are obtained from the explicit (deterministic) code construction of Spielman [Spi96]. Since the codes are encodable in linear-time, the compiler preserves the linear proof length and prover arithmetic complexity of the tensor IOP. The result is the point-query IOP for R_{R1CS} described in Theorem 2.

The precise details of the proof are given in Section 9.

2.8 On bounded-query zero knowledge

Theorem 2 guarantees zero-knowledge against *semi-honest* verifiers. However, we can achieve *bounded-query* zero-knowledge against a malicious verifier who makes at most $O(m^{\epsilon})$ queries, if we relax the verifier time in our construction to $\mathsf{poly}(|x|) + O(m^{\epsilon})$ field operations.

The reason behind this is as follows. By Theorem 3, the zero-knowledge property in Theorem 2 relies (among other things) on a family of *explicit* linear-time encodable error-correcting codes which themselves have a zero-knowledge property, whereby a single query to a codeword leaks no information about the encoded message. These codes suffice for semi-honest-verifier zero-knowledge because the honest verifier never learns more than one query of each codeword. By contrast, in the setting of bounded-query zero-knowledge, we require codes with a zero-knowledge property against a sublinear number of queries. The aforementioned codes do not satisfy this property. Instead, we show how to obtain suitable codes from a *probabilistic* construction of Druk and Ishai [DI14], leading to an IOP verifier whose randomness complexity is sublinear.

Obtaining an explicit construction of linear-time encodable zero-knowledge codes remains an interesting open problem, which would allow us to prove Theorem 2 with *both* polylogarithmic verifier complexity and bounded-query zero-knowledge.

2.8.1 Algebraic reformulation of zero knowledge

We provide algebraic reformulations for the properties of bounded-query zero knowledge and bounded-query uniformity. We use these throughout this work, and deem them to be of independent interest.

Let $\mathcal{C}$ be a randomized linear code with encoding function $\tilde{\mathrm{Enc}}\colon \mathbb{F}^{k_m} \times \mathbb{F}^{k_r} \to \mathbb{F}^n$. We can split the generator matrix $G \in \mathbb{F}^{n \times k}$ associated to $\tilde{\mathrm{Enc}}$ into two parts $G = \begin{bmatrix} G_m & G_r \end{bmatrix}$ so that $\tilde{\mathrm{Enc}}(m; r) = G_m m + G_r r$ is the sum of a *message part* $G_m m$ and a *randomness part* $G_r r$, which acts as "mask".

The lemma below involves two codes associated to $\mathcal{C}$:

- $\mathcal{C}^\perp := \{z \in \mathbb{F}^n \mid z^\intercal G = 0\}$ is the dual code of $\mathcal{C}$, which consists of all linear combinations z that "eliminate" the message part and the randomness part of a codeword, regardless of the choice of message and randomness. I.e., if $z \in \mathcal{C}^\perp$ then $z^\intercal(G_m m + G_r r) = 0$ for every $m \in \mathbb{F}^{k_m}$ and $r \in \mathbb{F}^{k_r}$.
- $\mathcal{D}(\mathcal{C}) := \{z \in \mathbb{F}^n \mid z^\intercal G_r = 0\}$ is the code consisting of all linear combinations that eliminate the randomness part. I.e., if $z \in \mathcal{D}(\mathcal{C})$ then $z^\intercal(G_m m + G_r r) = z^\intercal G_m m$ for every $m \in \mathbb{F}^{k_m}$ and $r \in \mathbb{F}^{k_r}$. In particular, the linear combination z "leaks" information about the encoded message if $z^\intercal G_m m$ is non-zero.

Note that $\mathcal{C}^\perp \subseteq \mathcal{D}(\mathcal{C})$.

Lemma 2.2. *For every $b \in \mathbb{N}$ the following equivalences hold:*

1. *$\mathcal{C}$ is b-query zero-knowledge if and only if the minimum weight of codewords in $\mathcal{D}(\mathcal{C}) \setminus \mathcal{C}^\perp$ is at least $b+1$;*
2. *$\mathcal{C}$ is b-query uniform if and only if $\mathcal{D}(\mathcal{C})$'s minimum absolute distance is at least $b+1$.*

The difference between the two equivalences is that for the weaker condition (bounded-query zero knowledge) the minimum-weight requirement is imposed only on codewords in $\mathcal{D}(\mathcal{C}) \setminus \mathcal{C}^\perp$, rather than on all non-zero codewords in $\mathcal{D}(\mathcal{C})$. Below we provide intuition about the equivalences, first discussing bounded-query uniformity and then bounded-query zero knowledge. Technical details are in Section 5.

Intuition for Equivalence 2. Consider the random variable $c := \{G_m m + G_r r\}_{r \leftarrow \mathbb{F}^{k_r}}$, which is a random encoding of the message $m \in \mathbb{F}^{k_m}$. If $\mathcal{D}(\mathcal{C})$ has minimum absolute distance $b+1$, then there is a linear combination $z \in \mathbb{F}^n$ of weight $b+1$ that eliminates the randomness part $G_r r$. The support of z gives $b+1$ entries of c that are correlated (via the non-zero coefficients of z), showing that $\mathcal{C}$ cannot be $(b+1)$-query uniform. On the other hand, if no linear combination $z \in \mathbb{F}^n$ of weight at most b eliminates the randomness part $G_r r$, then every linear combination of b entries of c is uniformly random. By an XOR Lemma (Lemma 5.4) this can only happen if each set of b entries of c is uniformly distributed.

Intuition for Equivalence 1. Again consider the random variable $c := \{G_m m + G_r r\}_{r \leftarrow \mathbb{F}^{k_r}}$. If the minimum weight of codewords in the set $\mathcal{D}(\mathcal{C}) \setminus \mathcal{C}^\perp$ is $b+1$, then there is a linear combination z of weight $b+1$ that eliminates the randomness part $G_r r$ ($z^\intercal G_r = 0$) but does not eliminate the message part $G_m m$ ($z^\intercal G_m \neq 0$). The support of z gives $b+1$ entries of c that can be used to distinguish between different messages m according to the value of $z^\intercal G_m m$ (using the non-zero coefficients of z). Therefore, $\mathcal{C}$ cannot be $(b+1)$-query zero-knowledge. On the other hand, suppose that no linear combination z of weight at most b can be used to distinguish between encodings of different messages. Then every low-weight linear combination of entries of c is either random (if z does not eliminate $G_r r$) or zero (if z eliminates both $G_r r$ and $G_m m$). Thus, no such low-weight z belongs to the set $\mathcal{D}(\mathcal{C}) \setminus \mathcal{C}^\perp$.

2.8.2 Tensor products of zero knowledge codes

As part of the tensor-query to point-query compiler (see Section 2.4), the prover sends to the verifier proof messages $\hat{\Pi}$ consisting of tensor-IOP proof messages Π encoded under a tensor code $\mathcal{C}^{\otimes t}$. The verifier has point-query access to the encoded messages $\hat{\Pi}$. To ensure that these queries do not leak information (up to a certain number of queries), we require the tensor code $\mathcal{C}^{\otimes t}$ to be zero-knowledge. To this end, we prove that the tensor product operation preserves the property of bounded-query zero-knowledge (and bounded-query uniformity). In particular, for $\mathcal{C}^{\otimes t}$ to be zero knowledge it will suffice for $\mathcal{C}$ to be zero knowledge. (We discuss how to obtain a zero-knowledge code that is linear-time encodable after this, in Section 2.8.3.)

Theorem 5. *Let $\mathcal{C}$ and $\mathcal{C}'$ be randomized linear codes.*

1. *If $\mathcal{C}$ is b-query zero-knowledge and $\mathcal{C}'$ is b'-query zero-knowledge, then $\mathcal{C} \otimes \mathcal{C}'$ is $\min(b, b')$-query zero-knowledge.*
2. *If $\mathcal{C}$ is b-query uniform and $\mathcal{C}'$ is b'-query uniform, then $\mathcal{C} \otimes \mathcal{C}'$ is $\min(b, b')$-query uniform.*

The formal statement and proof are provided in Section 6. Below we describe the main ideas behind the two items in Theorem 5.

For Item 2 (the simpler case), using the algebraic reformulation given in Lemma 2.2, it suffices to show that the minimum distance of $\mathcal{D}(\mathcal{C} \otimes \mathcal{C}')$ is at least $\min(b, b')$. Unfortunately, it is difficult to directly analyze $\mathcal{D}(\mathcal{C} \otimes \mathcal{C}')$. Although $\mathcal{D}(\mathcal{C} \otimes \mathcal{C}')$ can be expressed as a sum of linear-spaces each of which has a known minimum distance (see Figure 3), this cannot be used to prove any useful bounds. Instead, we analyze the space $R := \mathcal{D}(\mathcal{C}) \otimes \mathcal{D}(\mathcal{C}')^{\perp} \oplus \mathbb{F}^n \otimes \mathcal{D}(\mathcal{C}')$ (also shown in Figure 3). Since R contains $\mathcal{D}(\mathcal{C} \otimes \mathcal{C}')$, the minimum distance of R is a lower bound for the minimum distance of $\mathcal{D}(\mathcal{C} \otimes \mathcal{C}')$. Further, R is defined by a direct sum of two tensor-product spaces (whereas $\mathcal{D}(\mathcal{C} \otimes \mathcal{C}')$ seems to require three). This gives a decomposition of any element of R into two elements with strong restrictions on their rows and columns (when we view tensors of rank two as matrices), which belong to different spaces and yet must be equal in almost all of their entries for a low-weight element of R. This allows us to bound the minimum distance of R.

Item 1 (which is the harder case) follows in a similar fashion, using a different definition of R and accounting for the fact that the algebraic characterization of zero-knowledge in Lemma 2.2 uses the set $\mathcal{D}(\mathcal{C}) \setminus \mathcal{C}^{\perp}$, rather than the linear space $\mathcal{D}(\mathcal{C})$.

Note that one cannot hope to improve Item 2 to prove that $\mathcal{C} \otimes \mathcal{C}'$ is even $\max(b, b')$-query uniform in general. We know that the rows of any codeword of $\mathcal{C} \otimes \mathcal{C}'$ belong to $\mathcal{C}$, and must therefore satisfy various parity checks. Therefore, if b' is greater than the block length n of $\mathcal{C}$, then $\mathcal{C} \otimes \mathcal{C}'$ cannot be $\max(b, b')$-query uniform. However, one might hope that Item 1 could be improved to $O(bb')$-query zero knowledge, which would imply zero-knowledge against a verifier making a *linear* number of queries in Theorem 2.

2.8.3 Zero-knowledge codes with linear-time encoding

To prove our main theorem, Theorem 2, we require an explicit construction of a randomized linear code, that must be both linear-time encodable and 1-query zero-knowledge. Prior works such as [BCGGHJ17; Cer19] achieve this by applying a 1-out-of-2 secret sharing scheme to every element of the output of an explicit (non zero-knowledge) linear-time encodable code, such as [Spi96]. Given the encoding function Enc for a linear-time encodable code, the new code is defined by $\widetilde{\mathrm{Enc}}(m; r) := (\mathrm{Enc}(m) + r, r)$.

Investigating bounded-query zero-knowledge. To prove the variation on our main theorem, Theorem 2, with bounded-query zero-knowledge, we require randomized linear codes which are linear-time encodable as above, but with a stronger zero-knowledge property. In this case, the code must be b-query zero-knowledge where b is not only greater than 1, but may even be a constant fraction of the block length.

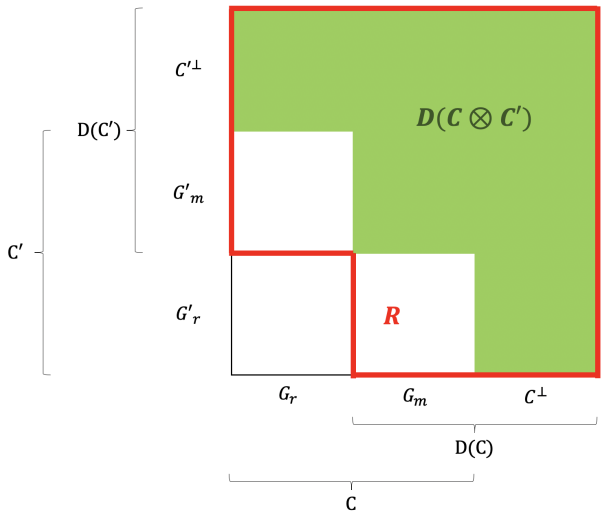

Figure 3: The linear spaces $\mathcal{D}(\mathcal{C} \otimes \mathcal{C}')$ and R as subspaces of $\mathbb{F}^n \otimes \mathbb{F}^{n'}$.

Prior works achieved these properties separately. For example, it is well-known that the Reed–Solomon code can be made b-query zero-knowledge by using b elements of encoding randomness, but their encoding functions incur costs quasilinear in the message length. On the other hand, the zero-knowledge properties of linear-time encodable codes, such as the explicit family by Spielman [Spi96] or the probabilistic family by Druk and Ishai [DI14], have not been investigated.

We prove the existence of codes satisfying both requirements, via a probabilistic construction. In the statement below, $H_q \colon [0,1] \to [0,1]$ denotes the q-ary entropy function (see Definition 7.2).

Theorem 6. *For every finite field $\mathbb{F}$, every $\epsilon \in (0,1)$, and every function $\beta \colon \mathbb{N} \to (0,1)$ bounded away from 1, letting $q := |\mathbb{F}|$, there is a circuit family $\{E_{k_m} : \mathbb{F}^{k_m} \times \mathbb{F}^{(H_q(\beta(k_m))+\epsilon)\cdot O(k_m)} \times \mathbb{F}^{O(k_m)} \to \mathbb{F}^{O(k_m)}\}_{k_m \in \mathbb{N}}$ such that: (1) E_{k_m} has size $O(k_m)$; (2) with probability at least $1 - q^{-\Omega_\epsilon(k_m)}$ over $R \in \mathbb{F}^{O(k)}$, the randomized linear code $\mathcal{C}_{k_m}$ whose encoding function is $\tilde{\mathrm{Enc}}_{k_m}(m;r) := E_{k_m}(m,r,R)$ has constant relative distance and is $O(\beta(k_m) \cdot k_m)$-query uniform.*

The precise statement of the theorem and its proof are provided in Section 7.

Below we provide intuition about the theorem statement by comparing the parameters to those achievable by the Reed–Solomon code; then provide an overview of the proof; and finally discuss related constructions and analyses. Derandomizing Theorem 6, namely the goal of obtaining an explicit family of codes that are both zero knowledge and linear-time encodable, remains an open problem.

Comparison with RS code. The relation between encoding randomness and bounded-query uniformity is simple for the Reed–Solomon code: b elements of encoding randomness ensure b-query uniformity. This relation is more complex for the code in Theorem 6, but we can understand its qualitative behavior by considering two regimes, depending on whether the desired b is small or large.

- *b is small.* If $\beta = O(q^{-\sigma})$ for $\sigma \in (0,1)$, then $H_q(\beta) \cdot k_m$ is bounded by $O(-\frac{\beta}{\sigma} \cdot \log \beta) \cdot k_m$, which itself is bounded by $O(\frac{\log k_m}{\sigma}) \cdot b$. This tells us that $O(\frac{\log k_m}{\sigma}) \cdot b$ elements of encoding randomness suffice for b-query uniformity, which in this regime is a factor of $O(\frac{\log k_m}{\sigma})$ more than for the Reed–Solomon code.

- *b is large.* If b is linear in the block length of the code (which is linear in k_m), then β is constant and so $H_q(\beta) \cdot k_m = O(b)$. This tells us that $O(b)$ elements of encoding randomness suffice for b-query uniformity, which in this regime is within a constant factor of the Reed–Solomon code.

The regime that we use in this paper is when b is large (linear in k_m).

Overview of proof of Theorem 6. We use the same code as in [DI14], which is a probabilistic construction (which we inherit). Our contribution is to show that their construction additionally satisfies the strong requirement of b-query uniformity, by using Lemma 2.2 and ideas from the analysis of [DI14].

Informally, Druk and Ishai [DI14] construct a family of distributions $\mathcal{G} = \{\mathcal{G}_k\}_{k\in\mathbb{N}}$ such that, for every $k \in \mathbb{N}$, $\mathcal{G}_k = \{G_R \in \mathbb{F}^{O(k)\times k}\}_{R\in\mathbb{F}^{O(k)}}$ is a distribution over generator matrices such that: (1) matrix-vector multiplication is computable in linear time; (2) for any fixed non-zero vector x, when G_R is sampled at random from the distribution, Gx is uniformly distributed. This latter property is known as *linear uniform output*, and implies that, with high probability over R, G_R has constant relative distance and dual distance.

We are interested in analyzing what happens if we split the message space of a generator matrix $G \in \mathcal{G}_k$ into two parts, one of length k_m for the actual message and another of length k_r for the encoding randomness, for $k_m+k_r = k$. As in Section 2.5, this induces a corresponding split in the generator matrix: $G = \begin{bmatrix} G_m & G_r \end{bmatrix}$. By Lemma 2.2, the probability that this code is b-query uniform is bounded from below by the probability that $(G_r)^\perp$, the dual of G_r, has minimum (absolute) distance at least $b + 1$.

Druk and Ishai [DI14] show that $G^\perp$ has constant relative distance with high probability. We observe that G_r inherits the linear-uniform output property from G, and then adapt their analysis to $(G_r)^\perp$. We now summarize the main ideas of their analysis when it is applied to our setting of b-query uniformity. Similarly, to the standard probabilistic proof of the Gilbert–Varshamov bound, requiring that $(G_r)^\perp$ has distance at least $b + 1$ is equivalent to showing that each non-zero vector $z \in \mathbb{F}^n$ with Hamming weight at most b is not in $(G_r)^\perp$, i.e., $zG_r \neq 0$. The linear-uniform output property of G implies that zG_r is uniformly distributed, over a random choice of G_r; therefore the probability that $z \in \mathbb{F}^n$ is in $(G_r)^\perp$ is at most q^{-k_r}. Taking a union bound over all vectors of weight at most b, of which there are at most $q^{H_q(b/n)\cdot n}$, gives an upper bound on the probability that the distance of $(G_r)^\perp$ is at most b.

We can choose parameters so that $n = O(k_m)$ and $k_r = O(H_q(b/n) \cdot n)$ with suitable constants so that $q^{-k_r} \cdot q^{H_q(b/n)\cdot n} = q^{-\Omega(k_m)}$. In combination with results on the distance of G, this yields Theorem 6.

In sum, we obtain a trade-off between the fraction of the message space allocated to the encoding randomness and the b-query uniformity of the code. For example, if (as we use in this paper) b is linear in n then $H_q(b/n)$ is constant and the encoding randomness is a constant fraction of the input.

Comparison with related work. Chen et al. [CCGHV07] show a result analogous to Theorem 6 for random codes: with high probability over a choice of random code with block length $O(k_m)$, using $H_q(\beta(k_m))\cdot O(k_m)$ elements of encoding randomness ensures $O(\beta(k_m) \cdot k_m)$-query zero-knowledge. Random codes, however, are not linear-time encodable. Theorem 6 can be viewed as strengthening the result for random codes in [CCGHV07, Theorem 11] to apply to the linear-time codes of [DI14] (and to proving the stronger property of bounded-query uniformity). The proofs of both results follow the standard template of the existence proof of codes meeting the Gilbert–Varshamov bound, except that the analyzed code family changes.

Druk and Ishai [DI14] give a linear-time secret sharing scheme for message vectors of constant length, based on the same code family used to prove Theorem 6. Their construction generalizes to a randomized encoding scheme with b-query zero-knowledge (and most likely b-query uniformity), where b is determined by the distance of the dual code. For this code family, the dual distance is linear in k_m, giving b-query zero-knowledge for $b = \Theta(k_m)$. However, encoding requires solving a system of linear equations whose dimension is the same length as the message, and so fails to be linear-time.

Ishai et al. [ISVW13; Wei16] give a generic construction of zero-knowledge codes from any linear code, which works by randomizing a generator matrix for the code, but this does not preserve linear-time encoding.

3 Preliminaries

3.1 Interactive oracle proofs with special queries

Definition 3.1. *An* **indexed relation** R *is a set of triples* $(\mathbb{i}, \mathbb{x}, \mathbb{w})$ *where* $\mathbb{i}$ *is the index,* $\mathbb{x}$ *the instance, and* $\mathbb{w}$ *the witness. The corresponding* **indexed language** $L(R)$ *is the set of pairs* $(\mathbb{i}, \mathbb{x})$ *for which there exists a witness* $\mathbb{w}$ *such that* $(\mathbb{i}, \mathbb{x}, \mathbb{w}) \in R$*.*

Definition 3.2. *A* **holographic IOP with query class** $\mathcal{Q}$ *(some set of functions) is a tuple* $\mathsf{IOP} = (\mathbf{I}, \mathbf{P}, \mathbf{V})$*, where* $\mathbf{I}$ *is the* indexer*,* $\mathbf{P}$ *the* prover*, and* $\mathbf{V}$ *the* verifier*. The indexer is a deterministic polynomial-time algorithm, while the prover and verifier are probabilistic polynomial-time interactive algorithms.*

In an offline phase, the indexer $\mathbf{I}$ *is given an index* $\mathbb{i}$ *and outputs an encoding* Π_0 *of* $\mathbb{i}$*.*

In an online phase, the prover $\mathbf{P}$ *receives as input an index-instance-witness tuple* $(\mathbb{i}, \mathbb{x}, \mathbb{w})$ *and the verifier* $\mathbf{V}$ *receives as input the instance* $\mathbb{x}$*; in addition, the verifier* $\mathbf{V}$ *has query access to* Π_0 *(in a precise sense specified below), which we denote as* $\mathbf{V}^{\Pi_0}(\mathbb{x})$*. The online phase consists of multiple rounds, and in each round the prover* $\mathbf{P}$ *sends a proof message* Π_i *and the verifier* $\mathbf{V}$ *replies with a challenge message* ρ_i*.*

The prover $\mathbf{P}$ *may compute its proof message* Π_i *based on its input* $(\mathbb{i}, \mathbb{x}, \mathbb{w})$ *and all the verifier challenges received thus far (none if* $i = 1$ *or* $\rho_1, \ldots, \rho_{i-1}$ *if* $i > 1$*). In contrast, the verifier* $\mathbf{V}$ *may compute its challenge message* ρ_i *based on its input* $\mathbb{x}$ *and on answers obtained by querying* $(\Pi_0, \Pi_1, \ldots, \Pi_i)$ *via queries in* $\mathcal{Q}$*. In more detail, the answer of a query* $q \in \mathcal{Q}$ *to* $(\Pi_0, \Pi_1, \ldots, \Pi_i)$ *is* $v := q(\mathbb{x}, \Pi_0, \Pi_1, \ldots, \Pi_i)$ *(this answer could also be a special error value in case the proof messages are not according to an expected format).*

After the interaction and all queries are concluded, the verifier $\mathbf{V}$ *accepts or rejects.*

Remark 3.3 (non-oracle messages)**.** We allow the prover in an IOP to also send, at any point in the interaction, arbitrary messages that the verifier will simply read in full (without making any queries) as in a usual interactive proof. We refer to such messages as *non-oracle messages*, to differentiate them from the *oracle messages* to which the verifier has query access. These non-oracle messages can typically be viewed as degenerate cases of oracle messages, and we use them in protocol descriptions for convenience of exposition.

A holographic interactive oracle proof $\mathsf{IOP} = (\mathbf{I}, \mathbf{P}, \mathbf{V})$ for an indexed relation R has completeness 1 and soundness error ϵ if the following holds.

- **Completeness.** For every index-instance-witness tuple $(\mathbb{i}, \mathbb{x}, \mathbb{w}) \in R$, the probability that $\mathbf{P}(\mathbb{i}, \mathbb{x}, \mathbb{w})$ convinces $\mathbf{V}^{\mathbf{I}(\mathbb{i})}(\mathbb{x})$ to accept is 1.
- **Soundness.** For every index-instance tuple $(\mathbb{i}, \mathbb{x}) \notin L(R)$ and malicious prover $\tilde{\mathbf{P}}$, the probability that $\tilde{\mathbf{P}}$ convinces $\mathbf{V}^{\mathbf{I}(\mathbb{i})}(\mathbb{x})$ to accept is at most ϵ.

Public coins. A holographic IOP is *public-coin* if each verifier message to the prover is a random string. This means that the verifier's randomness is its challenge messages $\rho_1, \ldots, \rho_{\mathsf{rc}}$. All verifier queries can be postponed, without loss of generality, to a query phase that occurs after the interactive phase with the prover.

Non-adaptive queries. A holographic IOP is *non-adaptive* if all verifier queries depend solely on the input instance $\mathbb{x}$ and the verifier's randomness, as opposed to some queries depending on answers to prior queries. For non-adaptive IOPs, the verifier $\mathbf{V}$ can be written as a pair of algorithms $(\mathbf{V}_{\mathrm{q}}, \mathbf{V}_{\mathrm{d}})$ where: (a) $\mathbf{V}_{\mathrm{q}}$ is probabilistic, takes as input the instance $\mathbb{x}$, interacts with the prover $\mathbf{P}$, and outputs a decision state σ and a query set I (for the index oracle and proof oracles); and (b) $\mathbf{V}_{\mathrm{d}}$ is deterministic, takes as input the decision state σ and query answers $\mathbf{v} \in \Sigma^I$, and outputs a decision bit. For this case, we define the relation $R(\mathbf{V})$ to be all pairs $(\sigma, \mathbf{v})$ such that there exist $\mathbb{x}$ and I with (σ, I) in the support of $\mathbf{V}_{\mathrm{q}}(\mathbb{x})$ and $\mathbf{V}_{\mathrm{d}}(\sigma, \mathbf{v}) = 1$.

Complexity measures. We consider several complexity measures:

– *round complexity* rc is the number of back-and-forth message exchanges between the prover and verifier;
– *answer alphabet* Σ is the alphabet over which oracle messages are defined;
– *proof length* $\mathsf{l} = \mathsf{li} + \mathsf{lp} + \mathsf{lc}$ where $\mathsf{li} := |\Pi_0|$ is the number of alphabet symbols output by the indexer, $\mathsf{lp} := |\Pi_1| + \cdots + |\Pi_{\mathsf{rc}}|$ is the total number of alphabet symbols sent in oracle messages by the prover, and lc is the total number of alphabet symbols sent in non-oracle messages by the prover;
– *randomness* r is the number of random bits used by the verifier;
– *query complexity* q is the total number of queries made by the verifier (to any oracle);
– *running time* ti is the running time of $\mathbf{I}$, tp is the running time of $\mathbf{P}$, and tv is the running time of $\mathbf{V}$. In the non-adaptive case, tq and td are the running times of the query and decision components of $\mathbf{V}$ respectively.

3.2 Point queries and tensor queries

We define the two query classes that we use in this paper, point queries and tensor queries.

Definition 3.4. *A* **holographic IOP with point queries** *is an IOP with the query class $\mathcal{Q}_{\mathrm{point}}$ defined as follows: $\mathcal{Q}_{\mathrm{point}}$ is all functions of the form $q(\mathbb{x}, \Pi_0, \Pi_1, \ldots, \Pi_i) = \Pi_j[k]$ for some $j \in \{0, 1, \ldots, i\}$ and location k. (If the location k does not exist, the answer is an error.) Namely, each query in the class $\mathcal{Q}_{\mathrm{point}}$ returns the symbol at a location of the encoded index ($j = 0$) or of a specified prover message ($j > 0$).*

Definition 3.5. *Given a finite field $\mathbb{F}$ and positive integers k, t, a* **holographic IOP with $(\mathbb{F}, k, t)$-tensor queries** *is an IOP with the query class $\mathcal{Q}_{\mathrm{tensor}}(\mathbb{F}, k, t)$ defined as follows: $\mathcal{Q}_{\mathrm{tensor}}(\mathbb{F}, k, t)$ contains all functions of the form $q(\mathbb{x}, \Pi_0, \Pi_1, \ldots, \Pi_i) = \langle q_0 \otimes q_1 \otimes \cdots \otimes q_t, \Pi_j \rangle$ for some $j \in \{0, 1, \ldots, i\}$ and vectors $q_0 \in \mathbb{F}^*$ and $q_1, \ldots, q_t \in \mathbb{F}^k$. (If the lengths of the linear combination $q_0 \otimes q_1 \otimes \cdots \otimes q_t$ and proof string Π_j do not match, the answer is an error.) Namely, each query in the class $\mathcal{Q}_{\mathrm{tensor}}$ returns the scalar product of a certain tensor vector and the encoded index ($j = 0$) or of a specified prover message ($j > 0$).*

Remark 3.6. In the context of tensor IOPs, we often view a proof message $\Pi \in \mathbb{F}^{\ell \cdot k^t}$ as consisting of ℓ "sub-messages" $\pi_1, \ldots, \pi_\ell$ each in $\mathbb{F}^{k^t}$. In this case, when describing a protocol, we may make a tensor query $q_1 \otimes \cdots \otimes q_t$ to one of the sub-messages π, with the understanding that one can specify an indicator vector $q_0 \in \mathbb{F}^\ell$ so that $\langle q_0 \otimes q_1 \otimes \cdots \otimes q_t, \Pi \rangle = \langle q_1 \otimes \cdots \otimes q_t, \pi \rangle$.

3.3 Robust proofs

We sometimes require, for the case of non-adaptive verifiers, a stronger notion of soundness, *robust soundness*. Informally this means that, up to a soundness error ϵ, the information read by the verifier is not only rejecting but also cannot be modified in few locations to make the verifier accept.

Definition 3.7. *A holographic interactive oracle proof $(\mathbf{I}, \mathbf{P}, \mathbf{V})$ for an indexed relation R has* **soundness error ϵ with robustness parameter α relative to distance Δ** *if $\mathbf{V}$'s queries are non-adaptive and, for every index-instance tuple $(\mathbb{i}, \mathbb{x}) \notin L(R)$ and malicious prover $\tilde{\mathbf{P}}$,*

$$\Pr\left[\Delta\left(\mathbf{v}, R(\mathbf{V})|_\sigma\right) \leq \alpha \;\middle|\; \begin{array}{r} [\tilde{\Pi}, (\sigma, I)] \leftarrow \langle \tilde{\mathbf{P}}, \mathbf{V}_{\mathrm{q}}(\mathbb{x}) \rangle \\ \mathbf{v} := (\mathbf{I}(\mathbb{i}), \tilde{\Pi})|_I \end{array}\right] \leq \epsilon \ .$$

Above, (σ, I) is the output of the query sampler algorithm $\mathbf{V}_{\mathrm{q}}(\mathbb{x})$ when interacting with $\tilde{\mathbf{P}}$ and $\tilde{\Pi}$ consists of all proof oracles sent by $\tilde{\mathbf{P}}$ during this interaction. The probability is taken over $\mathbf{V}_{\mathrm{q}}$'s randomness.

3.4 Proximity proofs

A *holographic interactive oracle proof of proximity* $\mathsf{IOPP} = (\mathbf{I}, \mathbf{P}, \mathbf{V})$ for an indexed relation R has completeness 1 and soundness error ϵ with distance function Δ if the following holds.

- **Completeness.** For every index-instance-witness tuple $(\mathbb{i}, \mathbb{x}, \mathbb{w}) \in R$, the probability that $\mathbf{P}(\mathbb{i}, \mathbb{x}, \mathbb{w})$ convinces $\mathbf{V}^{\mathbf{I}(\mathbb{i}),\mathbb{w}}(\mathbb{x})$ to accept is 1.
- **Proximity soundness.** For every index-instance tuple $(\mathbb{i}, \mathbb{x})$ and malicious prover $\tilde{\mathbf{P}}$, the probability that $\tilde{\mathbf{P}}$ convinces $\mathbf{V}^{\mathbf{I}(\mathbb{i}),\mathbb{w}}(\mathbb{x})$ to accept is at most $\epsilon(\Delta(\mathbb{w}, R|_{(\mathbb{i},\mathbb{x})}))$. Here the soundness error ϵ is a function of the Δ-distance of $\mathbb{w}$ to the set of valid witnesses $R|_{(\mathbb{i},\mathbb{x})} := \{\mathbb{w}' \mid (\mathbb{i}, \mathbb{x}, \mathbb{w}') \in R\}$. (Intuitively, one expects the soundness error to go down as the distance of $\mathbb{w}$ to valid witnesses increases.) If the set of valid witnesses $R|_{(\mathbb{i},\mathbb{x})}$ is empty (this happens when $(\mathbb{i}, \mathbb{x}) \notin L(R)$) then we use the convention $\Delta(\mathbb{w}, \emptyset) := 1$.

We remark that, while not explicit above, each verifier's query $q \in \mathcal{Q}$ is to $(\mathbf{I}(\mathbb{i}), \mathbb{w}, \Pi_1, \ldots, \Pi_i)$ and its answer is $q(\mathbb{x}, \mathbf{I}(\mathbb{i}), \mathbb{w}, \Pi_1, \ldots, \Pi_i)$; namely, the candidate witness $\mathbb{w}$ is also an input to the query function q. The definition of IOPs with particular query classes (e.g., point-query IOPs) extend naturally to this case.

Exact proximity proofs. We are sometimes in the position where if $\Delta(\mathbb{w}, R|_{(\mathbb{i},\mathbb{x})}) > 0$ then $\epsilon(\Delta(\mathbb{w}, R|_{(\mathbb{i},\mathbb{x})})) = \epsilon$ where $\epsilon \in (0, 1)$ does not depend on the distance. In other words, whenever $(\mathbb{i}, \mathbb{x}) \notin L(R)$, then for any malicious prover $\tilde{\mathbf{P}}$, the probability that $\tilde{\mathbf{P}}$ convinces $\mathbf{V}^{\mathbf{I}(\mathbb{i})}(\mathbb{x})$ to accept is at most ϵ. In such cases, we do not mention the distance function Δ, and use the terminology *exact interactive oracle proof of proximity* (exact IOPP), analogous to that used in [IW14] for probabilistically-checkable proofs.

3.5 Zero knowledge

The main notion of zero-knowledge that we achieve in this paper is (perfect) zero-knowledge against semi-honest verifiers. This means that for any choice of verifier randomness, an honest verifier's queries do not leak any information about the prover's witness. We also investigate (perfect) zero knowledge against malicious verifiers that make a bounded number of queries to the proof oracles. This means that the malicious verifier can send to the honest prover arbitrary messages and also query any proof oracle sent by the prover in arbitrary, possibly adaptive, ways as long as the number of queried locations does not exceed a given query bound, which we usually denote by b; the query bound can be a function of other parameters, e.g., input size.

Below we define the notion of view and then the notions of zero knowledge that we use.

Definition 3.8. *We denote by* $\mathrm{View}(\mathbf{B}, \mathbf{A}(a))$ *the* **view** *of* $\mathbf{A}$ *in an interactive protocol with* $\mathbf{B}$*. Namely, it is the random variable* $(r, b_1, \ldots, b_n, v_1, \ldots, v_m)$ *where* r *is* $\mathbf{A}$*'s randomness,* $b_1, \ldots, b_n$ *are non-oracle messages sent from* $\mathbf{B}$ *to* $\mathbf{A}$*, and* $v_1, \ldots, v_m$ *are the answers to* $\mathbf{A}$*'s queries to* $\mathbf{B}$*'s oracle messages. (There is no need to include any messages from* $\mathbf{A}$ *to* $\mathbf{B}$ *as these are implied by the other information in the view.)*

Definition 3.9. *Let* A *be an algorithm with adaptive query access to oracles* $\mathcal{O}_1, \ldots, \mathcal{O}_n$*. Let* $\mathbf{Q}$ *be a stateful query-checker algorithm which receives the adaptive queries of* A *and may output* $\perp$ *at any point. We say that* A *is a* $\mathbf{Q}$*-query algorithm if* $\mathbf{Q}$ *never outputs* $\perp$*.*

Definition 3.10. *A holographic IOP* $(\mathbf{I}, \mathbf{P}, \mathbf{V})$ *for an indexed relation* R *is* **(perfect) zero-knowledge with query-checker** $\mathbf{Q}$ *if there exists a polynomial-time simulator algorithm* $\mathbf{S}$ *such that, for every* $(\mathbb{i}, \mathbb{x}, \mathbb{w}) \in R$ *and* $\mathbf{Q}$*-query algorithm* $\tilde{\mathbf{V}}$*, the random variables* $\mathbf{S}^{\tilde{\mathbf{V}}}(\mathbb{i}, \mathbb{x})$ *and* $\mathrm{View}(\mathbf{P}(\mathbb{i}, \mathbb{x}, \mathbb{w}), \tilde{\mathbf{V}})$ *are identically distributed.*

Definition 3.11. *Let $\mathbf{Q}_b$ be a query-checker algorithm which outputs $\perp$ if more than b queries are made. For the query-checker $\mathbf{Q}_b$, we refer to A from Definition 3.9 as a b-query algorithm, and say that $(\mathbf{I}, \mathbf{P}, \mathbf{V})$ from Definition 3.10 has* **(perfect) zero-knowledge with query bound** *b.*

All results in this paper achieve the definition above via *straightline* simulators, which means that the simulator samples the view of the verifier by running through a single execution of the malicious verifier, simulating any (non-oracle) messages from the prover or the answers to any of the malicious verifier's queries.

For intermediate building blocks we will also consider a relaxation of the above definition where we require zero knowledge to hold only against *semi-honest* verifiers, which are malicious verifiers whose messages and queries are consistent with some execution of an honest verifier though they may be distributed differently than those of an honest verifier.

Definition 3.12. *A holographic interactive oracle proof $(\mathbf{I}, \mathbf{P}, \mathbf{V})$ for an indexed relation R is* **(perfect) zero knowledge against semi-honest verifiers** *if there exists a polynomial-time simulator algorithm $\mathbf{S}$ such that, for every $(\mathbb{i}, \mathbb{x}, \mathbb{w}) \in R$ and choice of verifier randomness ρ, the random variables $\mathbf{S}^{\mathbf{V}(\mathbb{i},\mathbb{x};\rho)}(\mathbb{i}, \mathbb{x})$ and $\mathrm{View}(\mathbf{P}(\mathbb{i}, \mathbb{x}, \mathbb{w}), \mathbf{V}(\mathbb{i}, \mathbb{x}; \rho))$ are identically distributed.*

Remark 3.13. If a holographic IOP $(\mathbf{I}, \mathbf{P}, \mathbf{V})$ is zero-knowledge with query-checker $\mathbf{Q}$, and for choice of verifier randomness ρ, the verifier $\mathbf{V}$ is a $\mathbf{Q}$-query algorithm, then $(\mathbf{I}, \mathbf{P}, \mathbf{V})$ is zero knowledge against semi-honest verifiers.

Finally, we also consider zero knowledge for proximity proofs, which we use towards our main result. The notion of (perfect) zero knowledge in the definition below is closely related to prior ones used for proximity proofs, such as for PCPPs in [IW14] and for IOPPs in [BCFGRS17]. The main technical difference is that we consider a query function f that determines how many queries the simulator makes to the witness as a function of the number of queries made to the witness *and any proof oracles sent by the prover*.

Definition 3.14. *A holographic interactive oracle proof of proximity $\mathsf{IOPP} = (\mathbf{I}, \mathbf{P}, \mathbf{V})$ for an indexed relation R is* **(perfect) zero knowledge with query function** *f if there exists a polynomial-time simulator algorithm $\mathbf{S}$ such that, for every $(\mathbb{i}, \mathbb{x}, \mathbb{w}) \in R$, query checker $\mathbf{Q}$, and $\mathbf{Q}$-query algorithm $\tilde{\mathbf{V}}$, the random variables $\mathbf{S}^{\tilde{\mathbf{V}},\mathbb{w}}(\mathbb{i}, \mathbb{x})$ and $\mathrm{View}(\mathbf{P}(\mathbb{i}, \mathbb{x}, \mathbb{w}), \tilde{\mathbf{V}})$ are identically distributed, and $\mathbf{S}$ is $f(\mathbf{Q})$-query with respect to $\mathbb{w}$.*

We conclude by remarking that all of the zero knowledge definitions above are sensitive to the choice of alphabet for the proof oracles because they all refer to the query complexity of certain verifiers (and a query to a proof oracle returns an entire alphabet symbol). In particular, when we establish that a certain construction achieves zero knowledge, this is proved for a specific choice of alphabet and there are guarantees for, e.g., if the alphabet is changed a smaller one and the query bound is increased correspondingly.

Remark 3.15. In the above definitions we required the simulator to output a verifier view, which in particular includes the verifier's randomness. However, as we do not restrict the efficiency of verifiers, the verifier's randomness could have super-polynomial size, in which case the (polynomial-time) simulator cannot sample it on the verifier's behalf. This is a minor technicality that is resolved, as in prior work, as follows: we assume that the verifier is initialized with its own randomness and the simulator is only required to produce the rest of the view (simulated messages and query answers) by having access to the verifier but not to its randomness.

3.6 Error-correcting codes

We introduce preliminaries on error-correcting codes.

We consider functions $c\colon [n] \to \mathbb{F}$ for a given domain $[n]$ and finite field $\mathbb{F}$. The (absolute) distance $d(c, c')$ between two functions $c, c'\colon [n] \to \mathbb{F}$ is the number of inputs for which c and c' differ; the relative distance is $\delta(c, c') := d(c, c')/n$. The weight of c is the number of inputs for which c is non-zero.

A linear error-correcting code $\mathcal{C}$ over $\mathbb{F}$ is a set of functions $c\colon [n] \to \mathbb{F}$ that form an $\mathbb{F}$-linear space. The block length is n and the message length is the rank k of the linear space $\mathcal{C}$; the rate is $\rho := k/n$. The minimum (absolute) distance $d := d(\mathcal{C})$ is the minimal (absolute) distance between any two distinct codewords c and c' in $\mathcal{C}$. The minimum relative distance $\delta := \delta(\mathcal{C})$ of the code is defined similarly.

For a given function $c\colon [n] \to \mathbb{F}$, if $d(c, \mathcal{C})$ is smaller than the unique decoding radius of $\mathcal{C}$ (which is $d(\mathcal{C})/2$), there is a unique codeword $\overline{c} \in \mathcal{C}$ closest to c; otherwise we default to setting $\overline{c} := 0$.

Codewords can also be viewed as vectors in $\mathbb{F}^n$, and we can associate to them messages in $\mathbb{F}^k$. This leads to two fundamental notions associated to a linear codes.

- *Generator matrix.* A generator matrix for $\mathcal{C}$ is a matrix $G \in \mathbb{F}^{n \times k}$ whose rows generate $\mathcal{C}$ as a linear subspace of $\mathbb{F}^n$. In particular, G is an injective map from $\mathbb{F}^k$ to $\mathcal{C}$, and thus associates to each vector $f \in \mathbb{F}^k$ the corresponding codeword $c := Gf \in \mathcal{C}$. The cost of encoding a message this way is $O(k \cdot n)$ operations (or, more generally, is linear in the number of non-zero entries in G). There are multiple generator matrices for the same code (they are in one-to-one correspondence with ordered bases of $\mathcal{C}$).

- *Parity-check matrix.* A parity-check matrix for $\mathcal{C}$ is a generator matrix $H \in \mathbb{F}^{n \times (n-k)}$ for the dual code $\mathcal{C}^\perp$ or equivalently, a matrix H such that, for every $c \in \mathbb{F}^n$, it holds that $c \in \mathcal{C}$ if and only if $H^\intercal c = 0$. The cost of checking whether a function c is a codeword in this way is $O(n \cdot (n-k))$ operations (or, more generally, is linear in the number of non-zero entries in H). If G is any generator matrix for $\mathcal{C}$, then it holds that $H^\intercal G = O_{(n-k)\times k}$, where $O_{(n-k)\times k}$ is the $(n-k) \times k$ matrix of all zeros.

  A parity-check matrix can be efficiently derived from G in $O(n \cdot k^2)$ operations by writing G in column-echelon form $\begin{bmatrix} I_k \\ A \end{bmatrix}$ for some $A \in \mathbb{F}^{(n-k)\times k}$ and setting $H := \begin{bmatrix} -A^\intercal \\ I_{n-k} \end{bmatrix}$. That H is a parity-check matrix follows from the fact that $H \cdot G^\intercal = 0$, which shows that H generates a sub-code of $\mathcal{C}^\perp$, and that $\mathcal{C}^\perp$ has the same rank $n-k$ as H, which then shows that H generates all of $\mathcal{C}^\perp$.

The mapping from a message to its codeword need not be performed as a matrix-vector product (a generator matrix times the message), and so we will use the notation $\mathrm{Enc}\colon \mathbb{F}^k \to \mathcal{C}$ to denote an injective function that performs this encoding possibly some other way (e.g., via an arithmetic circuit) for the sake of efficiency. Given the encoding function, one can always recover a generator matrix G by evaluating Enc at each unit vector of $\mathbb{F}^k$ in some canonical order and using the evaluations as the columns of G. We say that this is the generator matrix G associated with Enc.

Another common operation is *error-free decoding*, which is the task of computing the (unique) message that corresponds to a given codeword. This can be done via a left inverse G^+ of G (a matrix such that $G^+ G = I_k$); as G has full rank, the left-inverse G^+ can be computed as $(G^\intercal G)^{-1} G^\intercal$ in $O(n \cdot k^2)$ operations.

We will represent a linear code via (some representation of) all of these quantities, as follows.

Definition 3.16. *Let $\mathcal{C}$ be a linear code over a field $\mathbb{F}$. The* **public parameters** $(\mathrm{Enc}, G, H, G^+)$ *of $\mathcal{C}$ consist of an injective encoding function $\mathrm{Enc}\colon \mathbb{F}^k \to \mathbb{F}^n$ such that $\mathcal{C} = \mathrm{Enc}(\mathbb{F}^k)$, a generator matrix $G \in \mathbb{F}^{n\times k}$, a parity-check matrix $H \in \mathbb{F}^{n\times(n-k)}$, and a left-inverse $G^+ \in \mathbb{F}^{k\times n}$ of G.*

Interleaved codes. We say that c is an *interleaved codeword* if, for some $\ell \in \mathbb{N}$, we have $c \in \mathcal{C}^\ell$. We equivalently view c as a function from $[\ell] \times [n]$ to $\mathbb{F}$ or as a matrix in $\mathbb{F}^{\ell \times n}$.

We consider a single symbol of a function $c\colon [\ell] \times [n] \to \mathbb{F}$ to be a column of the $\ell \times n$ matrix that represents c (the column thus contains ℓ elements of $\mathbb{F}$). Accordingly, we define the block-wise (absolute) distance between two functions $c, c'\colon [\ell] \times [n] \to \mathbb{F}$ to be the number of columns in which c and c' differ. If c is within the unique decoding radius of $\mathcal{C}^\ell$, we denote by $\overline{c}$ the unique codeword in $\mathcal{C}^\ell$ that is closest to c (and otherwise we default to setting $\overline{c} := 0$); note that $\overline{c}$ is obtained by replacing each row of c with the corresponding closest codeword in $\mathcal{C}$.

Moreover, we define the block-wise relative distance of a set of functions $\{c^{(s)}\colon [\ell] \times [n] \to \mathbb{F}\}_s$ to $\mathcal{C}$ to be the fraction of columns where at least one of the functions deviates from a codeword, as follows:

$$\Delta(\{c^{(s)}\}_s, \mathcal{C}) := \frac{\left|\left\{ j \in [n] \;\middle|\; \exists\, (s,i) \text{ s.t. } c^{(s)}(i,j) \neq \overline{c}^{(s)}(i,j) \right\}\right|}{n} \ .$$

Tensor-product codes. The tensor product code $\mathcal{C}^{\otimes t}$ is the linear code in $\mathbb{F}^{n^t}$ with message length k^t, block length n^t, and distance d^t that comprises all functions $c\colon [n]^t \to \mathbb{F}$ whose restriction to any axis-parallel line is in $\mathcal{C}$. Namely, for every $j \in [t]$ and $a_1, \ldots, a_{j-1}, a_{j+1}, \ldots, a_t \in [n]$, the function $c'\colon [n] \to \mathbb{F}$ defined by $c'(i) := c(a_1, \ldots, a_{j-1}, i, a_{j+1}, \ldots, a_t)$ is in $\mathcal{C}$. The encoding function associated to $\mathcal{C}^{\otimes t}$ is defined below.

Definition 3.17. *Let $G\colon [n] \times [k] \to \mathbb{F}$ be the generator matrix associated with the encoding function* Enc *of a linear code $\mathcal{C}$ in $\mathbb{F}^n$, and $f\colon [k]^t \to \mathbb{F}$ a message function with inputs indexed by $(i_1, \ldots, i_t)$. The* **$\mathcal{C}^{\otimes t}$-encoding** *of f is the function $\mathrm{Enc}_{1,\ldots,t}(f)\colon [n]^t \to \mathbb{F}$, with inputs indexed by $(j_1, \ldots, j_t)$, defined as follows:*

$$\mathrm{Enc}_{1,\ldots,t}(f)(j_1, \ldots, j_t) := \sum_{i_1,\ldots,i_t \in [k]} G(j_1, i_1) \cdots G(j_t, i_t) f(i_1, \ldots, i_t) \ .$$

While the encoding function of $\mathcal{C}^{\otimes t}$ is most conveniently defined in terms of the generator matrix G of Enc, it may be significantly cheaper to compute it using $\mathrm{Enc}_{\mathcal{C}}$.

Lemma 3.18. *If the encoding function $\mathrm{Enc}\colon \mathbb{F}^k \to \mathcal{C}$ of a code $\mathcal{C}$ has arithmetic complexity $\theta(k) \cdot k$, then the encoding function $\mathrm{Enc}_{1,\ldots,t}\colon \mathbb{F}^k \to \mathcal{C}^{\otimes t}$ of the tensor code $\mathcal{C}^{\otimes t}$ has arithmetic complexity $\frac{\rho^{-t}-1}{\rho^{-1}-1}\theta(k) \cdot k^t$. (In particular, if $\mathcal{C}$ is linear-time encodable then so is the tensor code $\mathcal{C}^{\otimes t}$.)*

Proof sketch. View a message in $\mathbb{F}^{k^t}$ as k^{t-1} vectors in $\mathbb{F}^k$. Encode each of the vectors to get a partial encoding with $k^{t-1}n$ elements. View it as $k^{t-2}n$ vectors in $\mathbb{F}^k$ and encode each of them to obtain a partial encoding with $k^{t-2}n^2$ elements. Compute t partial encodings to obtain the full encoding.

After performing r partial encodings, there are $k^{t-r}n^r = k^t\rho^{-r}$ elements of $\mathbb{F}$, which are then viewed as $k^{t-1}\rho^{-r}$ vectors of $\mathbb{F}^k$ before computing the next partial encoding. In this way, the final encoding can be obtained via $\theta k^t + \theta k^t\rho^{-1} + \cdots + \theta k^t\rho^{-t+1} = (1 + \rho^{-1} + \cdots + \rho^{-t+1})\theta k^t$ operations. Summing the geometric series yields the claimed arithmetic complexity. □

In later sections we will also need more general notions of encodings, which consider *partial encodings* of a high-dimensional array along different axis-parallel lines. We provide these definitions below.

Definition 3.19. *Let $G\colon [n] \times [k] \to \mathbb{F}$ be the generator matrix of a linear code $\mathcal{C}$ in $\mathbb{F}^n$. The* **encoding** *of a function $c\colon [k] \times [a_1] \times \cdots \times [a_h] \to \mathbb{F}$ is the function $\mathrm{Enc}(c)\colon [n] \times [a_1] \times \cdots \times [a_h] \to \mathbb{F}$ defined as follows:*

$$\mathrm{Enc}(c)(j, \cdot, \ldots, \cdot) := \sum_{i \in [k]} G(j,i) c(i, \cdot, \ldots, \cdot) \ .$$

More generally, we write Enc_r to indicate an encoding operation that is applied to the r-th coordinate.

3.7 Zero knowledge codes

Let $\mathcal{C}$ be a linear code over a field $\mathbb{F}$ with message length k and block length n, and let $(\mathrm{Enc}, G, H, G^+)$ be public parameters associated to the code. For any choice of $k_m, k_r \in \mathbb{N}$ such that $k = k_m + k_r$, we can split the domain of $\mathcal{C}$'s encoding function Enc into two parts, by considering the bivariate function $\tilde{\mathrm{Enc}}\colon \mathbb{F}^{k_m} \times \mathbb{F}^{k_r} \to \mathbb{F}^n$ defined as $\tilde{\mathrm{Enc}}(m; r) := \mathrm{Enc}(m \| r)$. This split naturally induces a *probabilistic encoding*: $\tilde{\mathrm{Enc}}(m)$ is defined to be the random variable $\{\tilde{\mathrm{Enc}}(m; r)\}_{r \leftarrow \mathbb{F}^{k_r}}$.

We are interested in *hiding properties* of the foregoing encoding, for given values of k_m and k_r. In light of this, we use the term *randomized linear code* to refer to a code $\mathcal{C}$ specified via the tuple $(\tilde{\mathrm{Enc}}, G, G^+, H)$, where the bivariate function $\tilde{\mathrm{Enc}}$ is given explicitly as part of the code description. Observe that, since $\tilde{\mathrm{Enc}}$ is linear in both the message and the randomness, we can uniquely split the generator matrix G in two parts: $G = \begin{bmatrix} G_m & G_r \end{bmatrix}$, where $G_m \in \mathbb{F}^{n \times k_m}$ corresponds to the message space of Enc, and $G_r \in \mathbb{F}^{n \times k_r}$ corresponds to the randomness space of Enc. We will rely on this notation throughout this paper.

Common notions associated to a code change accordingly. For example, the message space is now $\mathbb{F}^{k_m}$, and so the message length is k_m and the rate is $\rho_m = k_m/n$. We similarly refer to $\mathbb{F}^{k_r}$ as the randomness space, and call k_r the randomness length and $\rho_r = k_r/n$ the randomness rate. The distance of the code is now the worst-case distance across all possible randomness choices of two encodings.

Below we define the hiding properties that we consider for randomized linear codes, (i) $\mathbf{Q}$*-query zero knowledge*, which requires that certain sets of coordinates leak no information about the encoded message; (ii) $\mathbf{Q}$*-query uniformity*, which strengthens the prior notion to require that any such set of coordinates is uniformly random. The following definition uses the same definition of a $\mathbf{Q}$-query algorithm is as in Definition 3.9.

Definition 3.20. *A randomized linear code $\mathcal{C}$ is* **$\mathbf{Q}$-query zero-knowledge** *for a query-checker $\mathbf{Q}$ if there exists a probabilistic polynomial-time simulator $\mathcal{S}$ such that, for every message $m \in \mathbb{F}^{k_m}$ and $\mathbf{Q}$-query algorithm A, the following two random variables are equidistributed:*
- $\{\mathrm{View}(\tilde{\mathrm{Enc}}(m; r), A)\}_{r \leftarrow \mathbb{F}^{k_r}}$*, which is the view of A when making adaptive queries to* $\tilde{\mathrm{Enc}}(m; r)$*;*
- $\mathcal{S}^A$*, which is the output of the simulator $\mathcal{S}$ given black-box access to A.*

Moreover, if the answer to each query made by A is simply uniformly distributed, then we say that $\mathcal{C}$ is **$\mathbf{Q}$-query uniform**.

As a special case of Definition 3.20, we also consider: (i) *bounded-query zero knowledge*, which mirrors the corresponding standard notion for PCPs and IOPs (and, as shown in Appendix C, is essentially equivalent to the zero knowledge notion considered in [ISVW13; Wei16]); (ii) *bounded-query uniformity*, which strengthens the prior notion to require that any sufficiently small set of coordinates is uniformly random.

Definition 3.21. *Let $\mathbf{Q}_b$ be a query-checker algorithm which outputs $\perp$ if more than b queries are made, as in Definition 3.11. For the query-checker $\mathbf{Q}_b$, we refer to A from Definition 3.9 as a b-query algorithm, and $\mathcal{C}$ from Definition 3.20 as a b-query zero-knowledge or b-query uniform code.*

Next, we define some simple randomised linear codes, starting with an encoding scheme used in [BCGGHJ17].

Definition 3.22 (split encodings)**.** *Let $\mathcal{C}$ be a linear code with encoding function $\mathrm{Enc}\colon \mathbb{F}^{k_m} \to \mathbb{F}^n$. Then, the randomised linear code $\tilde{\mathcal{C}}$ has encoding function $\tilde{\mathrm{Enc}}\colon \mathbb{F}^{k_m} \times \mathbb{F}^n \to \mathbb{F}^{2n}$ defined as follows:*

$$\tilde{\mathrm{Enc}}(m; r) := \Big(\mathrm{Enc}(m) + r, r\Big) \ .$$

Let $\mathbf{Q}$ *be a query-checker for* $\tilde{\mathrm{Enc}}(m, r)$ *which outputs* 0 *if queries are made to both the* j*-th symbol of* $\mathrm{Enc}(m) + r$ *and* r*. It is easy to see that* $\tilde{\mathcal{C}}$ *is* $\mathbf{Q}$*-query uniform, and in particular, is* 1*-query uniform.*

If $\mathcal{C}$ *has rate* ρ*, relative distance* δ*, and encoding time* $\theta(k_m) \cdot k_m$*, then* $\tilde{\mathcal{C}}$ *has rate* $\rho/2$*, relative distance* $\delta/2$*, and encoding time* $\theta(k_m) \cdot k_m + n$*.*

Example 1 (hiding in code concatenation)**.** We illustrate the differences between the notions of bounded-query zero knowledge and bounded-query uniformity via a natural code operation: *code concatenation*.

Let $\mathcal{C}_{\mathsf{in}}$ be a b_{in}-query zero knowledge code with encoding function $\tilde{\mathrm{Enc}}_{\mathsf{in}} \colon \mathbb{F}^{k_m} \times \mathbb{F}^{k_r} \to \mathbb{F}^n$, and let $\mathcal{C}_{\mathsf{out}}$ be a b_{out}-query zero knowledge code with encoding function $\tilde{\mathrm{Enc}}_{\mathsf{out}} \colon \mathbb{F} \times \mathbb{F}^{k'_r} \to \{0,1\}^{n'}$. The concatenated code $\mathcal{C}$ has encoding function $\tilde{\mathrm{Enc}} \colon \mathbb{F}^{k_m} \times \mathbb{F}^{k_r + k'_r \cdot n} \to \{0,1\}^{n' \cdot n}$ defined as follows:

$$\tilde{\mathrm{Enc}}\big(m; (r, r_1, \ldots, r_n)\big) := \Big(\tilde{\mathrm{Enc}}_{\mathsf{out}}\big(\tilde{\mathrm{Enc}}_{\mathsf{in}}(m; r)_1; r_1\big), \ldots, \tilde{\mathrm{Enc}}_{\mathsf{out}}\big(\tilde{\mathrm{Enc}}_{\mathsf{in}}(m; r)_n; r_n\big)\Big) \ .$$

Note that the encoding functions use independent encoding randomness each time they are used: first for encoding the message (via the inner code), and then for each symbol of the resulting encoding (via the outer code).

The concatenated code $\mathcal{C}$ is b-query zero knowledge for $b := b_{\mathsf{in}} b_{\mathsf{out}} + b_{\mathsf{in}} + b_{\mathsf{out}}$. Indeed, reading any b symbols of $\tilde{\mathrm{Enc}}(m)$ reveals at most $\lfloor \frac{b}{b_{\mathsf{out}}+1} \rfloor = \lfloor \frac{b_{\mathsf{in}} b_{\mathsf{out}} + b_{\mathsf{in}} + b_{\mathsf{out}}}{b_{\mathsf{out}}+1} \rfloor = b_{\mathsf{in}}$ symbols of $\tilde{\mathrm{Enc}}_{\mathsf{in}}(m)$, and thus reveals no information of m. (As we shall see in Section 2.6 and Appendix A, the same computation tells us how zero knowledge is preserved when robustifying an IOP.)

However, if we assume that $\mathcal{C}_{\mathsf{in}}$ is b_{in}-query uniform and $\mathcal{C}_{\mathsf{out}}$ is b_{out}-query uniform then we *cannot* conclude that $\mathcal{C}$ is b-query uniform. (Naturally, we can still conclude that $\mathcal{C}$ is b-query zero knowledge as we have only strengthened the assumptions on $\mathcal{C}_{\mathsf{in}}$ and $\mathcal{C}_{\mathsf{out}}$.) Indeed, $\tilde{\mathrm{Enc}}(m)$ is always the concatenation of codewords in $\mathcal{C}_{\mathsf{out}}$, and therefore many substrings of the codeword are not independent.

In sum, code concatenation amplifies bounded-query zero knowledge but not bounded-query uniformity, and in particular a simulator for a zero knowledge concatenated code may have to do more than "sample a random symbol and return it".

4 Tensor IOP for R1CS with (semi)honest-verifier zero knowledge

In this section, we give a holographic tensor IOP for the relation R_{R1CS} that is zero knowledge against (semi-)honest verifiers.

Definition 4.1 (R1CS)**.** *The indexed relation R_{R1CS} is the set of all triples*

$$(\mathtt{i}, \mathtt{x}, \mathtt{w}) = \big((\mathbb{F}, m, n, A, B, C), (\ell, x), w\big)$$

where $\mathbb{F}$ is a finite field, A, B, C are matrices in $\mathbb{F}^{m \times n}$, each with at most M non-zero entries, $x \in \mathbb{F}^{\ell}$, $w \in \mathbb{F}^{n-\ell}$, and $z := (x, w) \in \mathbb{F}^n$ is a vector such that $Az \circ Bz = Cz$. (Here "$\circ$" denotes the entry-wise product between two vectors.)

Theorem 4.2. *For every finite field $\mathbb{F}$ and positive integers k, t, there is a $(\mathbb{F}, k, t)$-tensor holographic IOP, with non-adaptive queries, for the indexed relation R_{R1CS} that supports instances over $\mathbb{F}$ with $M = \ell \cdot k^t$, $m = k^t - 2$,$n = k^t - 6$ and $n_{\mathrm{in}} = \ell_{\mathrm{in}} \cdot k^{t_{\mathrm{in}}}$, where $\ell_{\mathrm{in}} \in [k-1]$, and has the following parameters:*
- *soundness error is $O((M + k^t)/|\mathbb{F}|)$;*
- *round complexity is $O(\log(M + k^t))$;*
- *proof length is $O(M + k^t)$ elements in $\mathbb{F}$;*
- *query complexity is $O(1)$;*
- *the prover sends $O(\log \ell + t \log k)$ non-oracle messages;*
- *the indexer uses $O(M)$ field operations;*
- *the prover uses $O(k^t + M)$ field operations;*
- *the verifier uses $O(\ell_{\mathrm{in}} \cdot k^{t_{\mathrm{in}}} + \log \ell + t \log k)$ field operations;*
- *the verifier has randomness complexity $O(t \log k)$.*

Moreover, the tensor IOP is semi-honest-verifier zero-knowledge.

4.1 Preliminaries

Zero-knowledge gadgets. To help achieve zero-knowledge in our construction, we will augment the R1CS instance with the following matrix gadgets.

Claim 4.3. *Let I_s denote the $s \times s$ identity matrix. Define the following vectors and matrices:*

$$\begin{aligned}
&\vec{e}_1 := \begin{bmatrix}1 & 0 & 0\end{bmatrix}, && \vec{e}_2 := \begin{bmatrix}0 & 1 & 0\end{bmatrix}, && \vec{e}_3 := \begin{bmatrix}0 & 0 & 1\end{bmatrix} && \in \mathbb{F}^3 \ , \\
&A_s := \vec{e}_1 \otimes I_s \ , && B_s := \vec{e}_2 \otimes I_s, && C_s := \vec{e}_3 \otimes I_s && \in \mathbb{F}^{s \times 3s} \ .
\end{aligned}$$

The solutions to the equation $A_s z \circ B_s z = C_s z$ are all vectors $z \in \mathbb{F}^{3s}$ of the form $\begin{pmatrix}\vec{p} & \vec{q} & \vec{p} \circ \vec{q}\end{pmatrix}$ where $\vec{p}, \vec{q} \in \mathbb{F}^s$.

During the R1CS protocol, the prover and verifier engage in a *twisted scalar-product sub-protocol* and three executions of a *holographic lincheck sub-protocol* which are specified in [BCG20]. We note that their results can be generalised using the notion of ϵ-biased generators, which allows the verifier's randomness complexity to be improved. First, we define ϵ-biased generators. Then, we define the scalar-product and holographic lincheck relations, along with theorems specifying tensor IOPs for both relations.

ϵ-biased generators. We define ϵ-biased generators over arbitrary fields, using a notion of absolute bias instead of using the deviation from uniform probabilities.

Definition 4.4 (ϵ-biased generators)**.** *Let $G\colon (\mathbb{F}\setminus\{0\})^s \to (\mathbb{F}\setminus\{0\})^n$ be a function. We say that G is ϵ-biased if*

$$\max_{w\in\mathbb{F}^n\setminus\{0\}} \Pr_{x\in\mathbb{F}^s}[\langle w, G(x)\rangle = 0] \le \epsilon \ .$$

Lemma 4.5. *Let $G\colon (\mathbb{F}\setminus\{0\})^s \to (\mathbb{F}\setminus\{0\})^n$ be an ϵ-biased generator and let $G'\colon (\mathbb{F}\setminus\{0\})^{s'} \to (\mathbb{F}\setminus\{0\})^{n'}$ be an ϵ'-biased generator. Then, the function $G\otimes G'\colon (x,x') \mapsto G(x)\otimes G'(x')$ is an $(\epsilon+\epsilon')$-biased generator.*

Proof. Let w be a non-zero vector in $\mathbb{F}^{n\cdot n'}$. Choose $x \leftarrow (\mathbb{F}\setminus\{0\})^s$ and $x' \leftarrow (\mathbb{F}\setminus\{0\})^{s'}$ uniformly at random. Writing

$$\langle w, G(x)\otimes G'(x')\rangle = w^{\intercal}\cdot(G(x)\otimes I_{n'})\cdot(I_n\otimes G'(x')) \ ,$$

it is easy to see that $w^{\intercal}\cdot(G(x)\otimes I_{n'})$ is zero with probability at most ϵ over the choice of x, in which case $\langle w, G(x)\otimes G'(x')\rangle$ is also zero. If $w^{\intercal}\cdot(G(x)\otimes I_{n'})$ is not zero, then $w^{\intercal}\cdot(G(x)\otimes I_{n'})\cdot(I_n\otimes G'(x'))$ is zero with probability at most ϵ' over the choice of x'. □

To achieve the parameters stated in Theorem 4.13, we use the following generator.

Definition 4.6. *Let $G\colon (\mathbb{F}\setminus\{0\})^{\log k} \to (\mathbb{F}\setminus\{0\})^k$ be defined by $G(x_1,\ldots,x_{\log k}) = \left(\prod_{i\in S} x_i\right)_{S\subseteq[\log k]}$*

By the Schwarz–Zippel lemma, Definition 4.6 gives a $\frac{\log k}{|\mathbb{F}|-1}$-biased generator with arithmetic complexity $O(k)$.

Relations and subprotocols. The *twisted scalar-product sub-protocol* and the *holographic lincheck sub-protocol* are specified by the following definitions and theorems. (Note that if we set $y := 1^n$ in the definition below then we recover "standard" scalar products.)

Definition 4.7. *The* **twisted scalar product** *relation R_{TSP} is the set of tuples $(\mathtt{i},\mathtt{x},\mathtt{w}) = (\bot,(\mathbb{F},n,y,\tau),(a,b))$ where $a,b,y\in\mathbb{F}^n$, $\tau\in\mathbb{F}$, and $\langle a\circ y, b\rangle = \tau$.*

Theorem 4.8 ([BCG20])**.** *For every finite field $\mathbb{F}$ and positive integers k,t, there is a $(\mathbb{F},k,t)$-tensor exact IOPP for the relation R_{TSP} that supports instances over $\mathbb{F}$ with $n=\ell\cdot k^t$ and $y = y_0\otimes y_1\otimes\ldots\otimes y_t$ for $y_0\in\mathbb{F}^\ell$, $y_1,\ldots,y_t\in\mathbb{F}^k$ and has the following parameters: soundness error is $O(\frac{\log n}{|\mathbb{F}|})$; round complexity is $O(\log n)$; proof length is $O(n)$ elements in $\mathbb{F}$; query complexity is $O(1)$; the prover uses $O(n)$ field operations; the verifier uses $O(\log\ell + t\log k)$ field operations; and the verifier has randomness complexity $O(t\log k)$.*

Definition 4.9. *Let $U\in\mathbb{F}^{m\times n}$ be a matrix with M non-zero entries. The* **sparse representation** *of U consists of $\mathrm{val}_U\in\mathbb{F}^M$, $\mathrm{row}_U\in[m]^M$ and $\mathrm{col}_U\in[n]^M$ such that val_U contains the M non-zero entries of $U\in\mathbb{F}^{m\times n}$ and, for all $\kappa\in[M]$, $\mathrm{val}_U(\kappa) = U(\mathrm{row}_U(\kappa),\mathrm{col}_U(\kappa))$.*

Definition 4.10. *The* **lincheck** *indexed relation R_{LC} is the set of all triples*

$$(\mathtt{i},\mathtt{x},\mathtt{w}) = \big(U,(\mathbb{F},M,m,n),(v,v_{\mathrm{U}})\big)$$

where $U\in\mathbb{F}^{m\times n}$ is a matrix with M non-zero entries given in sparse representation, $v\in\mathbb{F}^n, v_{\mathrm{U}}\in\mathbb{F}^m, \mathrm{val}_U\in\mathbb{F}^M, \mathrm{row}_U\in[m]^M, \mathrm{col}_U\in[n]^M$ and $Uv = v_{\mathrm{U}}$.

Lemma 4.11 ([BCG20])**.** *For every finite field $\mathbb{F}$ and positive integers k,t, given an ϵ-biased generator $G\colon\mathbb{F}^s\to\mathbb{F}^k$ with circuit complexity $O(k)$, there is an exact $(\mathbb{F},k,t)$-tensor holographic IOPP for the indexed relation R_{LC} that supports instances over $\mathbb{F}$ with $M=\ell\cdot k^t$, $m,n=k^t$ and has the following parameters:*

soundness error is $O((M+k^t)/|\mathbb{F}|+t\epsilon)$; round complexity is $O(\log(M+k^t))$; proof length is $O(M+k^t)$ elements in $\mathbb{F}$; query complexity is $O(1)$; the prover uses $O(M+k^t)$ field operations; the verifier uses $O(\log(M/k^t)+t\log k)$ field operations; and the verifier has randomness complexity $O(t(s+\log k))$. The index $\mathbb{i}$ consists of $(A,B,C,[m],[n])$, where A, B and C are given in their sparse representations. Moreover, the indexer algorithm is degenerate: the encoding of an index $\mathbb{i}$ is the index itself.

Remark 4.12. The queries used in our tensor IOP construction and those [BCG20] may be highly structured. For example, a query $q \in (\mathbb{F}^k)^t$ in which each component is produced by the generator from Definition 4.6 can be described using only $t \log k$ field elements. As such, we allow the verifier to produce a description of each tensor query rather than computing the entire query explicitly, which gives the stated verification time.

4.2 Our construction

The holographic tensor IOP is given in Construction 4.14. In the following theorem, the performance parameters of the tensor IOP are described in terms of the soundness error, communication complexity, and round complexity of the sub-protocols. Later, in Section 9, this allows us to demonstrate how our final result depends critically on these two subprotocols.

Write $k_{\mathrm{SP}}(N)/|\mathbb{F}|$ for the soundness error of the twisted scalar-product protocol, $c_{\mathrm{SP}}(N)$ for its communication complexity and $r_{\mathrm{SP}}(N)$ for its round complexity. Write $k_{\mathrm{LC}}(N)/|\mathbb{F}|$ for the soundness error of the holographic lincheck protocol, $c_{\mathrm{LC}}(N)$ for its communication complexity and $r_{\mathrm{LC}}(N)$ for its round complexity.

Theorem 4.13. *For every finite field $\mathbb{F}$ and positive integers k, t, given an ϵ-biased generator $G\colon (\mathbb{F}\setminus\{0\})^s \to (\mathbb{F}\setminus\{0\})^k$ with arithmetic complexity $O(k)$, there is a $(\mathbb{F},k,t)$-tensor holographic IOP for the indexed relation R_{R1CS} that supports instances over $\mathbb{F}$ with $M=\ell\cdot k^t$, $m=k^t-2$,$n=k^t-6$ and $n_{\mathrm{in}}=\ell_{\mathrm{in}}\cdot k^{t_{\mathrm{in}}}$, where $\ell_{\mathrm{in}}\in[k-1]$, and has the following parameters:*

- *soundness error is $O((k_{\mathrm{LC}}+k_{\mathrm{SP}})/|\mathbb{F}|+t\epsilon)$;*
- *round complexity is $O(r_{\mathrm{LC}}+r_{\mathrm{SP}})$;*
- *proof length is $O(M+k^t)$ elements in $\mathbb{F}$;*
- *query complexity is $O(1)$;*
- *the prover sends $O(c_{\mathrm{LC}}+c_{\mathrm{SP}})$ non-oracle messages;*
- *the indexer uses $O(M)$ field operations;*
- *the prover uses $O(k^t+M)$ field operations;*
- *the verifier uses $O(\ell_{\mathrm{in}}\cdot k^{t_{\mathrm{in}}}+\log\ell+t\log k)$ field operations;*
- *the verifier has randomness complexity $O(t(s+\log k))$.*

Moreover, the tensor IOP is semi-honest-verifier zero-knowledge.

Construction 4.14 (tensor IOP for R1CS)**.** We construct a holographic interactive oracle proof $\mathsf{IOP}=(\mathbf{I},\mathbf{P},\mathbf{V})$ with tensor queries for the indexed relation R_{R1CS}. Given as input an index $\mathbb{i}=(\mathbb{F},m,n,A,B,C)$, the indexer $\mathbf{I}$ runs the indexer for the lincheck protocol (implied by Lemma 4.11) on input U to obtain $(\mathrm{row}_U,\mathrm{col}_U,[m],[n])$ for each $U\in\left\{\begin{bmatrix}A&0\\0&A_2\end{bmatrix},\begin{bmatrix}B&0\\0&B_2\end{bmatrix},\begin{bmatrix}C&0\\0&C_2\end{bmatrix}\right\}$, and outputs the oracle message

$$\Pi_0 := \left(\left\{\begin{bmatrix}A&0\\0&A_2\end{bmatrix},\begin{bmatrix}B&0\\0&B_2\end{bmatrix},\begin{bmatrix}C&0\\0&C_2\end{bmatrix}\right\},[m],[n]\right) \quad .$$

The prover $\mathbf{P}$ takes as input the index $\mathbb{i}$, instance $\mathbb{x}=(\ell,x)$, and witness $\mathbb{w}=w$, while the verifier $\mathbf{V}$ has query access to the index $\mathbb{i}$ and takes as input the instance $\mathbb{x}$. Let $G\colon (\mathbb{F}\setminus\{0\})^s\to(\mathbb{F}\setminus\{0\})^k$ be an ϵ-biased generator with arithmetic complexity $O(k)$.

- The prover $\mathbf{P}$ relies on Claim 4.3 to sample a random solution $u \in \mathbb{F}^6$ to the equation $A_2u \circ B_2u = C_2u$ such that the first two entries of u are equal to $(1, 0)$.

- The prover $\mathbf{P}$ constructs the full assignment $z := (x, w, u) \in \mathbb{F}^{k^t}$. The prover samples vectors y, y_A and $y_B \in \mathbb{F}^{k^t}$ uniformly at random, subject to the condition that the final 2 entries of y_A are $(0, 1)$. The prover computes the vectors

$$z_A := \begin{bmatrix} A & 0 \\ 0 & A_2 \end{bmatrix} z, \qquad z_B := \begin{bmatrix} B & 0 \\ 0 & B_2 \end{bmatrix} z, \qquad z_C := \begin{bmatrix} C & 0 \\ 0 & C_2 \end{bmatrix} z,$$
$$w_A := \begin{bmatrix} A & 0 \\ 0 & A_2 \end{bmatrix} y, \qquad w_B := \begin{bmatrix} B & 0 \\ 0 & B_2 \end{bmatrix} y, \qquad w_C := \begin{bmatrix} C & 0 \\ 0 & C_2 \end{bmatrix} y \quad \in \mathbb{F}^{k^t} \ .$$

  The prover sends the oracle message $\Pi_1 := (z, z_A, z_B, z_C, y, y_A, y_B, w_A, w_B, w_C) \in \mathbb{F}^{10 \cdot k^t}$.

- The verifier $\mathbf{V}$ sends uniformly random seeds $\rho_1, \ldots, \rho_t \in (\mathbb{F} \setminus \{0\})^s$ and the uniformly random challenge $\alpha \in \mathbb{F} \setminus \{0\}$.

- The prover $\mathbf{P}$ computes the query vector $r := G(\rho_1) \otimes \cdots \otimes G(\rho_t)$, and the field elements

$$\nu_C := \langle r, z_C \rangle, \ \mu_1 := \langle z_A \circ r, y_B \rangle + \langle y_A \circ r, z_B \rangle, \text{ and } \mu_0 := \langle y_A \circ r, y_B \rangle.$$

  The prover $\mathbf{P}$ sends the non-oracle message $(\nu_C, \mu_1, \mu_0) \in \mathbb{F}^3$.

- The verifier $\mathbf{V}$ performs consistency checks. First, $\mathbf{V}$ queries z_C in Π_1 at $r = G(\rho_1) \otimes \cdots \otimes G(\rho_t)$ in order to obtain the answer $\langle r, z_C \rangle$. Then, $\mathbf{V}$ checks that $\nu_C = \langle r, z_C \rangle$, which shows that ν_C is the correct answer to the query on z_C.

  Moreover, $\mathbf{V}$ checks that the (claimed) satisfying assignment z is consistent with the partial assignment x as follows: sample uniformly random seeds $\sigma_1, \ldots, \sigma_{t_{\text{in}}+1} \in \mathbb{F}^s$; compute vectors $s_i = G(\sigma_i)$ for $i \in [t_{\text{in}}]$, and compute $s_{t_{\text{in}}+1}$ by computing $G(\sigma_{t_{\text{in}}+1})$ and changing all but the first ℓ_{in} entries to zero. Set $s'_{t_{\text{in}}+1}$ to be the first ℓ_{in} entries. Then set each of the vectors $s_{t_{\text{in}}+2}, \ldots, s_t \in \mathbb{F}^k$ to equal $(1, 0, \ldots, 0) \in \mathbb{F}^k$; query z in Π_1 at the tensor $s := s_1 \otimes \cdots \otimes s_t$ in order to obtain the answer $\langle s, z \rangle$; and check that $\langle s, z \rangle = \langle s_1 \otimes \cdots \otimes s_{t_{\text{in}}} \otimes s'_{t_{\text{in}}+1}, x \rangle$.

- The prover $\mathbf{P}$ and verifier $\mathbf{V}$ engage in several sub-protocols, which make use of the masking vectors y, y_A and y_B, and the images w_A, w_B, w_C of y under A, B and C, to check that $z_A \circ z_B = z_C$, $Az = z_A$, $Bz = z_B$ and $Cz = z_C$ while ensuring zero-knowledge. Note that $\mathbf{V}$ can query, for example, $\alpha z_A + y_A$ with query q by applying the query $(\alpha, 1) \otimes q$ to (z_A, y_A).
  - A twisted scalar-product protocol with instance $\mathtt{x} = (\mathbb{F}, m, r, \alpha^2\nu_C + \alpha\mu_1 + \mu_0)$ and witness $\mathtt{w} = (\alpha z_A + y_A, \alpha z_B + y_B)$ to show that $\langle (\alpha z_A + y_A) \circ r, \alpha z_B + y_B \rangle = \alpha^2\nu_C + \alpha\mu_1 + \mu_0$.
  - A lincheck protocol with $\mathtt{i} = A$, $\mathtt{x} = (\mathbb{F}, M, m, n)$, and $\mathtt{w} = (\alpha z + y, \alpha z_A + w_A)$ to show that $\alpha z_A + w_A = A(\alpha z + y)$.
  - A lincheck protocol with $\mathtt{i} = B$, $\mathtt{x} = (\mathbb{F}, M, m, n)$, and $\mathtt{w} = (\alpha z + y, \alpha z_B + w_B)$ to show that $\alpha z_B + w_B = B(\alpha z + y)$.
  - A lincheck protocol with $\mathtt{i} = C$, $\mathtt{x} = (\mathbb{F}, M, m, n)$, and $\mathtt{w} = (\alpha z + y, \alpha z_C + w_C)$ to show that $\alpha z_C + w_C = C(\alpha z + y)$.

Lemma 4.15 (completeness)**.** *Construction 4.14 has perfect completeness.*

Sketch of proof. Let (x, w) be a solution to the R1CS instance. Since $A_2 u \circ B_2 u = C_2 u$, we know that $z := (x, w, u)$ satisfies the augmented R1CS instance in which A, B, C are combined with A_2, B_2, C_2. The prover $\mathbf{P}$ computes $z_A = \begin{bmatrix} A & 0 \\ 0 & A_2 \end{bmatrix} z$ and similarly for z_B and z_C, so the equation $z_A \circ z_B = z_C$ holds.

This means that, for every choice of $r = G(\rho_1) \otimes \cdots \otimes G(\rho_t)$, we have $\langle z_A \circ r, z_B \rangle = \langle r, z_C \rangle$. Expanding the expression $\langle (\alpha z_A + y_A) \circ r, \alpha z_B + y_B \rangle$, we see that

$$\begin{aligned} \langle (\alpha z_A + y_A) \circ r, \alpha z_B + y_B \rangle &= \alpha^2 \langle z_A \circ r, z_B \rangle + \alpha(\langle z_A \circ r, y_B \rangle + \langle y_A \circ r, z_B \rangle) + \langle y_A \circ r, y_B \rangle \\ &= \alpha^2 \langle r, z_C \rangle + \alpha \mu_1 + \alpha \mu_0 \\ &= \alpha^2 \nu_C + \alpha \mu_1 + \mu_0 \ . \end{aligned}$$

This means that the twisted scalar-product protocol succeeds.

Moreover, for every choice of α, we have $A(\alpha z + y) = \alpha A z + A y = \alpha z_A + w_A$, which means that the lincheck protocol with A succeeds. Similar reasoning holds for B and C. So the three scalar product sub-protocols succeed.

Finally, for every choice of $s_1, \ldots, s_{t_{\text{in}}+1}$ and setting $s_{t_{\text{in}}+2} = \cdots = s_t = (1, 0, \ldots, 0) \in \mathbb{F}^k$ it holds that $\langle s_1 \otimes \cdots \otimes s_t, z \rangle = \langle s_1 \otimes \cdots \otimes s_{t_{\text{in}}} \otimes s'_{t_{\text{in}}+1}, x \rangle$, which is the equation checked by the verifier. □

Lemma 4.16 (soundness)**.** *Let ϵ_{TSP} and ϵ_{LC} be the soundness errors of the (twisted) scalar-product and lincheck protocols. Construction 4.14 has soundness error*

$$\epsilon := \max \left\{ \epsilon_{\text{TSP}} + t\epsilon + \frac{2}{|\mathbb{F}| - 1}, \epsilon_{\text{LC}} + \frac{1}{|\mathbb{F}| - 1}, (t_{\text{in}} + 1)\epsilon \right\} \ .$$

Proof. Let $(\mathbb{i}, \mathbb{x}) = \big((\mathbb{F}, m, n, A, B, C), (\ell, x)\big) \notin L(R)$ and fix a malicious prover. Let

$$\Pi_1 = (z, z_A, z_B, z_C, y, y_A, y_B, w_A, w_B, w_C)$$

be the first message sent by the malicious prover. At least one of the following cases must hold.

- *The Hadamard product is incorrect: $z_A \circ z_B \neq z_C$.* By Lemma 4.5, the vector $r = G(\rho_1) \otimes \ldots \otimes G(\rho_t)$ is the output of a $t\epsilon$-biased generator. Since $\langle z_A \circ z_B - z_C, r \rangle = 0$ if and only if $\langle z_A \circ r, z_B \rangle = \langle z_C, r \rangle$, we have $\Pr_r[\langle z_A \circ r, z_B \rangle = \langle z_C, r \rangle] \leq t\epsilon$. Suppose that $\langle z_A \circ r, z_B \rangle \neq \langle z_C, r \rangle$ and let $\nu_C \in \mathbb{F}$ be the claimed value of the inner product sent by the malicious prover in the second message. Either $\langle z_C, r \rangle \neq \nu_C$ or $\langle z_A \circ r, z_B \rangle \neq \nu_C$ (or both). In the former case, the verifier will reject due to the check that $\nu_C = \langle z_C, r \rangle$. In the latter case, we apply the Schwartz–Zippel lemma to the non-zero polynomial $h(\alpha) := \langle (\alpha z_A + y_A) \circ r, \alpha z_B + y_B \rangle - \alpha^2 \nu_C - \alpha \mu_1 - \mu_0$ of degree 2, concluding that $\Pr_\alpha[\langle (\alpha z_A + y_A) \circ r, \alpha z_B + y_B \rangle = \alpha^2 \nu_C + \alpha \mu_1 + \mu_0] \leq \frac{2}{|\mathbb{F}|-1}$. By the soundness of the twisted scalar-product protocol, the verifier accepts with probability at most ϵ_{TSP}. By a union bound, the probability of accepting is at most $\epsilon_{\text{TSP}} + t\epsilon + \frac{2}{|\mathbb{F}|-1}$.
- *A linear combination is incorrect: $z_A \neq Az$ or $z_B \neq Bz$ or $z_C \neq Cz$.* Suppose that $z_A \neq Az$, without loss of generality. Then, there exists some $i \in [k^t]$ such that $(z_A)_i \neq (Az)_i$. We apply the Schwartz–Zippel lemma to the non-zero polynomial $P(\alpha) := \alpha\,[(z_A)_i - (Az)_i] + [(w_A)_i - (Ay)_i]$ of degree 1, concluding that $P(\alpha) = 0$ with probability at most $\frac{1}{|\mathbb{F}|-1}$, and hence $\Pr_\alpha[A(\alpha z + e) = \alpha z_A + w_A] \leq \frac{1}{|\mathbb{F}|-1}$. If $A(\alpha z + e) \neq \alpha z_A + w_A$, then by the soundness of the lincheck protocol, the verifier accepts with probability at most ϵ_{LC}. By a union bound, the acceptance probability is at most $\epsilon_{\text{LC}} + \frac{1}{|\mathbb{F}|-1}$.

- *Inconsistency with the partial assignment: $z \neq (x, w)$ for some w.* Note that after setting the final entries of $s_{t_{\text{in}}+1}$ to zero, it is still the output of an ϵ-biased generator. By Lemma 4.5, the vector $s_1 \otimes \cdots \otimes s_{t_{\text{in}}} \otimes s'_{t_{\text{in}}+1}$ is the output of a $(t_{\text{in}}+1)\epsilon$-biased generator. This implies that $\Pr_{s_1,\ldots,s_{t_{\text{in}}+1}}[\langle s_1 \otimes \cdots \otimes s_t, z\rangle = \langle s_1 \otimes \cdots \otimes s_{t_{\text{in}}+1}, x\rangle] \leq (t_{\text{in}}+1)\epsilon$. The acceptance probability is at most $(t_{\text{in}}+1)\epsilon$.

□

Lemma 4.17 (zero-knowledge)**.** *Construction 4.14 is (semi)honest-verifier zero-knowledge.*

Proof. We give an efficient simulator $\mathbf{S}$ for Construction 4.14, such that for every index-instance tuple $(\mathbb{i}, \mathbb{x}) \in L(R)$ and verifier randomness r, the random variables $\mathbf{S}^{\mathbf{V}}(\mathbb{i}, \mathbb{x}, r)$ and $\text{View}(\mathbf{P}(\mathbb{i}, \mathbb{x}, \mathbb{w}), \mathbf{V}(\mathbb{i}, \mathbb{x}, r))$ are identically distributed.

1. Start simulating $\mathbf{V}$.
2. Sample (ν_C, μ_1, μ_0) uniformly at random from $\mathbb{F}^3$, and send to $\mathbf{V}$ as non-oracle messages.
3. Receive the challenge α from $\mathbf{V}$. Receive the seeds $\rho_1, \ldots, \rho_t$ from $\mathbf{V}$ and compute $r = G(\rho_1) \otimes \cdots \otimes G(\rho_t)$.
4. Sample uniformly random $e, e_A, e_B \in \mathbb{F}^{k^t}$ conditioned on $\langle e_A \circ r, e_B\rangle = \alpha^2\nu_C + \alpha\mu_1 + \mu_0$, and such that the entries of e_A corresponding to the $(1,0)$ in z_A are equal to $(\alpha, 1)$. Compute
$$f_A := \begin{bmatrix} A & 0 \\ 0 & A_2 \end{bmatrix} e, \qquad f_B := \begin{bmatrix} B & 0 \\ 0 & B_2 \end{bmatrix} e, \qquad f_C := \begin{bmatrix} C & 0 \\ 0 & C_2 \end{bmatrix} e \quad \in \mathbb{F}^{k^t} \ .$$
5. Engage in a twisted scalar-product subprotocol with $\mathbf{V}$ to prove that $\langle e_A \circ r, e_B\rangle = \alpha^2\nu_C + \alpha\mu_1 + \mu_0$, and lincheck subprotocols to prove that $Ae = f_A$, $Be = f_B$ and $Ce = f_C$.
6. Answer the verifier's tensor query $\langle r, z_C\rangle$ with ν_C. Receive the seeds $\sigma_1, \ldots, \sigma_{t_{\text{in}}+1}$, compute $s_1, \ldots, s_{t_{\text{in}}+1}$, and answer the verifier's tensor query $\langle s_1 \otimes \cdots \otimes s_t, z\rangle$ with $\langle s_1 \otimes \cdots \otimes s_{t_{\text{in}}}, x\rangle$.

Efficiency. We show that $\mathbf{S}$ runs in polynomial time. This is clear apart from Item 4. To see that Item 4 is also efficient, note that the condition $\langle e_A \circ r, e_B\rangle = \alpha^2\nu_C + \alpha\mu_1 + \mu_0$ from Item 4 imposes a quadratic constraint on e_A, e_B. The simulator samples e_A first uniformly at random. Now, e_A enforces a linear constraint on e_B. Since one entry of e_A is α which is non-zero, e_B is sampled by sampling all entries except the entry corresponding to α uniformly at random, and then choosing the entry corresponding to α to satisfy the linear constraint.

Finally, we argue that $\mathbf{S}$ simulates the verifier's view perfectly. In a simulated view, all of the values $(e_A, e_B, e, \nu_C, \mu_1, \mu_0)$ are uniformly random (except for three entries of e_A being fixed to $(\alpha, 1)$) conditioned on satisfying the relation checked by the twisted scalar-product sub-protocol. The vectors f_A, f_B and f_C are then fully determined by e. The values of any queries made as part of the sub-protocols are determined by e, f_A f_B and f_C.

Correctness. To see that $\mathbf{S}$ simulates the view of $\mathbf{V}$ perfectly, we will display all of the randomisers used by the prover in a real proof, and then examine the distribution of oracle and non-oracle messages and queries.

First, we display the randomizers in the last 2 entries of z_A, z_B, z_C, y_A, y_B, and the last 6 entries of z, y. All values denoted by letters and asterisks are sampled uniformly at random in a real execution by the honest prover:

$$\begin{array}{c|cc} z_A[-2:] & 1 & 0 \\ y_A[-2:] & 0 & 1 \\ \hline z_B[-2:] & a_1 & a_2 \\ y_B[-2:] & b_1 & b_2 \\ \hline z_C[-2:] & a_1 & 0 \end{array}$$

	$z_A[-2:]$	$z_B[-2:]$	$z_C[-2:]$
$z[-6:]$			
$y[-6:]$	$*$	$*$	$*$

We now examine the distribution of oracle and non-oracle messages and queries.

- *Claim: The vector $\alpha z + y$ is uniformly random.* This is obvious due to the values of z and y as displayed above.
- *Claim: the distribution of $\alpha z_A + y_A$ is uniformly random, except for the first three entries which are $(\alpha, 1)$.* This is obvious due to the values of z_A and y_A as displayed above.
- *Claim: The messages $\mu_1 = \langle z_A \circ r, y_B \rangle + \langle y_A \circ r, z_B \rangle$ and $\mu_0 = \langle y_A \circ r, y_B \rangle$, and the vector $\alpha z_B + y_B$ are uniformly random.* These values can be written

$$\begin{bmatrix} \mu_1 \\ \mu_0 \\ \alpha z_B + y_B \end{bmatrix} = \begin{bmatrix} (y_A \circ r)^\intercal & (z_A \circ r)^\intercal \\ 0^{k^t} & (y_A \circ r)^\intercal \\ \alpha I_{k^t} & I_{k^t} \end{bmatrix} \begin{bmatrix} z_B \\ y_B \end{bmatrix} \ .$$

  Consider the sub-matrix obtained by dropping the columns corresponding to all but the final 2 entries of z_B and y_B. By choice of z_A and y_A, and since all entries of r are non-zero, this sub-matrix has full row-rank. The matrix equation above can be rewritten as the sum of a vector which depends on the witness, and the aforementioned sub-matrix multiplied by the random input vector (a_1, a_2, b_1, b_2). The claim follows.
- *Claim: The query answer $\nu_C = \langle r, z_C \rangle$ is uniformly distributed.* This follows from the fact that the following equation holds, in which every term except ν_C is known to be uniformly distributed.

$$\langle (\alpha z_A + y_A) \circ r, \alpha z_B + y_B \rangle = \alpha^2 \nu_C + \alpha \mu_1 + \mu_0 \ .$$

- *Claim: The answers to the queries made as part of the three lincheck protocols are fully determined by $\alpha z + y$.* As described in Construction 4.14, the verifier makes, for example, query $(\alpha, 1) \otimes q$ to (z, y) which is the same as query q to $\alpha z + y$, and queries $(\alpha, 1) \otimes q'$ to (z_A, w_A) which is the same as query q' to $\alpha z_A + w_A$, and similarly for B and C. This means that these query answers are determined by the vectors $\alpha z + y$, $\alpha z_A + w_A$, $\alpha z_B + w_B$ and $\alpha z_C + w_C$, and hence all by determined by $\alpha z + y$, which is a random vector.
- The answer to the query $\langle s_1 \otimes \cdots \otimes s_t, z \rangle$ is always equal to $\langle s_1 \otimes \cdots \otimes s_{t_{\text{in}}}, x \rangle$ which is independent of the witness.

□

5 Algebraic reformulation of zero knowledge codes

We provide algebraic reformulations of the conditions in Definition 3.21 that we use in this work. Throughout this section we fix a randomized linear code $\mathcal{C} = (\mathrm{Enc}, G, H, G^+)$ over a field $\mathbb{F}$, and state results that involve the following vector space: $\mathcal{D}(\mathcal{C}) := \{z \in \mathbb{F}^n \mid z^\intercal G_r = 0\}$.

Lemma 5.1. *The following are equivalent:*
1. *$\mathcal{C}$ is b-query zero-knowledge for exactly the integers $b \in [B]$.*
2. *The minimum weight of codewords in $\mathcal{D}(\mathcal{C}) \setminus \mathcal{C}^\perp$ is $B+1$.*

Lemma 5.2. *The following conditions are equivalent:*
1. *$\mathcal{C}$ is b-query uniform for exactly the integers $b \in [B]$.*
2. *$\mathcal{D}(\mathcal{C})$ has minimum (absolute) distance $B+1$.*

We first prove Lemma 5.2 in Section 5.1 as it is simpler, and then prove Lemma 5.1 in Section 5.2.

Remark 5.3. The second condition in Lemma 5.2 is closely related to the condition on the rows of G given in [Wei16, Lemma 6.4.1], which essentially states that $\mathcal{D}(\mathcal{C})$ has no codeword of weight less than $B+1$. That this condition implies b-query zero-knowledge is proved in [Wei16, Lemma 6.4.1]. Here, we strengthen that result to an *equivalence* by making two modifications:
- we consider the stronger notion of b-query uniformity rather than b-query zero-knowledge;
- we consider a condition on $\mathcal{D}(\mathcal{C})$, namely, its minimum distance (there is no codeword of weight less than $B+1$ but there does exist a codeword of weight $B+1$).

In this section we use several times the following version of the XOR Lemma.

Lemma 5.4 (XOR Lemma)**.** *Two random variables $X = (X_1, \ldots, X_n)$ and $Y = (Y_1, \ldots, Y_n)$ over $\mathbb{F}^n$ are equidistributed if and only if, for every $z \in \mathbb{F}^n$, the random variables $z^\intercal X$ and $z^\intercal Y$ over $\mathbb{F}$ are equidistributed. In particular, a random variable $X = (X_1, \ldots, X_n) \in \mathbb{F}^n$ is uniformly distributed if and only if, for every non-zero $z \in \mathbb{F}^n$, the random variable $z^\intercal X \in \mathbb{F}$ is uniformly distributed.*

5.1 Proof of Lemma 5.2

We prove each of the two implications in turn.

$1 \Rightarrow 2$. By hypothesis, $\mathcal{C}$ is B-query uniform but not $(B+1)$-query uniform. This implies that there exists a message $m^* \in \mathbb{F}^{k_m}$ and a subset $J^* \subseteq [n]$ with $B+1$ entries such that $\mathrm{Enc}(m^*, r)_{J^*}$ is not uniformly distributed. By Lemma 5.4, there exists non-zero $z^* \in \mathbb{F}^J$ such that $\langle e|_{J^*}, z^* \rangle$ is not uniformly distributed.

Let $w^* := (w^*_m, w^*_r)$, where $((w^*_m)^\intercal)_i := \sum_{j \in J} (z^*)^\intercal_j (G_m)_{i,j}$ for each $i \in [k_m]$, and $((w^*_r)^\intercal)_i := \sum_{j \in J} (z^*)^\intercal_j (G_r)_{i,j}$ for each $i \in [k_r]$. Then, w^*_r ***must*** be the zero vector, otherwise $\langle e|_{J^*}, z^* \rangle$ would be uniformly distributed. Therefore, z^* can be padded with zeroes to obtain a codeword in $\mathcal{D}(\mathcal{C})$ of weight $B+1$. Further, any codeword in $\mathcal{D}(\mathcal{C})$ with weight at most B contradicts the B-query uniformity of $\mathcal{C}$ via Lemma 5.4. Therefore, the minimum distance of $\mathcal{D}(\mathcal{C})$ is exactly $B+1$.

$2 \Rightarrow 1$. Fix a message $m \in \mathbb{F}^{k_m}$ and a subset $J \subseteq [n]$ of at most B locations. We argue that, for a random $r \in \mathbb{F}^{k_r}$, the entries in J of $e := \mathrm{Enc}(m, r)$ are uniformly distributed. By Lemma 5.4, it suffices to check that for every non-zero vector $z \in \mathbb{F}^J$ the random variable $\langle z, e|_J \rangle$ is uniformly distributed. Let $G = \begin{bmatrix} G_m & G_r \end{bmatrix}$ be the generator matrix of $\mathcal{C}$, so that $e = G_m m + G_r r$. Setting $w := (w_m, w_r)$ defined similarly to w^*_m and w^*_r above. Then, writing e in terms of m, r and G, we have

$$\langle z, e|_J \rangle = \sum_{j \in J} z_j e_j = \sum_{j \in J} z_j (G_m m + G_r r)_j = \langle w_m, m \rangle + \langle w_r, r \rangle \ .$$

By our hypothesis, w_r *cannot* be the zero vector, otherwise, z can be padded with zeroes to obtain a codeword in $\mathcal{D}(\mathcal{C})$ of weight B, which is less than the minimum distance of $\mathcal{D}(\mathcal{C})$. Hence w_r is non-zero, so $\langle r, w_r \rangle$ is uniformly distributed since r is uniformly distributed over $\mathbb{F}^{k_r}$, and so $\langle z, e|_J \rangle$ is also uniformly distributed. The non-zero vector $z \in \mathbb{F}^J$ was arbitrary, and by Lemma 5.4 we conclude that $e|_J$ is uniformly distributed. Further, m and J were also arbitrary, which proves that $\mathcal{C}$ is b-query uniform. Finally, it is clear that $\mathcal{C}$ is not $(B+1)$-query uniform because the codeword in $\mathcal{D}(\mathcal{C})$ of weight $B+1$ that exists by hypothesis gives a linear relation between $B+1$ entries of e. This shows that if $|J| = b+1$ then $e|_J$ is not uniformly distributed.

5.2 Proof of Lemma 5.1

We prove each of the two implications in turn, using the definition of b-query zero-knowledge given by Definition C.1.

$\mathbf{1 \Rightarrow 2}$. By hypothesis, $\mathcal{C}$ is B-query zero-knowledge but not $(B+1)$-query zero-knowledge. This implies that there exist distinct messages $m, m' \in \mathbb{F}^{k_m}$ and a subset $J \subseteq [n]$ of at most $B+1$ locations such that the distributions $\{e|_J \mid e = \mathrm{Enc}(m; r), r \leftarrow \mathbb{F}^{k_r}\}$ and $\{e'|_J \mid e' = \mathrm{Enc}(m'; r'), r' \leftarrow \mathbb{F}^{k_r}\}$ are not identical.

By Lemma 5.4, there exists a vector $z \in \mathbb{F}^J$ such that the distributions of $\langle z, e|_J \rangle$ and $\langle z, e'|_J \rangle$ are not identical. Let $G = \begin{bmatrix} G_m & G_r \end{bmatrix}$ be the generator matrix of $\mathcal{C}$, so that $e = G_m m + G_r r$ and similarly for e'. Let $w_m \in \mathbb{F}^{k_m}$ and $w_r \in \mathbb{F}^{k_r}$ be defined by $(w_m^\intercal)_i := \sum_{j \in J} z_j^\intercal (G_m)_{i,j}$ for each $i \in [k_m]$, and $(w_r^\intercal)_i := \sum_{j \in J} z_j^\intercal (G_r)_{i,j}$ for each $i \in [k_r]$. Then, as in the proof of Lemma 5.2, we have

$$\langle z, e|_J \rangle = \langle w_m, m \rangle + \langle w_r, r \rangle \ , \qquad \langle z, e'|_J \rangle = \langle w_m, m' \rangle + \langle w_r, r' \rangle \ .$$

By our hypothesis, w_r must be the zero vector, otherwise $\langle z, e|_J \rangle$ and $\langle z, e'|_J \rangle$ are both uniformly random over $\mathbb{F}$. Therefore, z can be padded with zeroes to obtain a codeword $z^* \in \mathcal{D}(\mathcal{C})$ with at most $B+1$ entries. Further, w_m must be non-zero, otherwise both $\langle z, e|_J \rangle$ and $\langle z, e'|_J \rangle$ would be equal to zero. Hence $z^* G_m \neq 0$ and z^* is not an element of $\mathcal{C}^\perp$, which can be written $\{x \mid xG_m = 0 \wedge xG_r = 0\}$. Finally, note that z^* must have *exactly* $B+1$ non-zero entries; a z^* with weight at most B would give a linear distinguisher for some set of B entries of e and e' and contradicts the B-query zero-knowledge property of $\mathcal{C}$.

$\mathbf{2 \Rightarrow 1}$. We show that for every set $J \subseteq [n]$ of size at most B, and every message pair $m, m' \in \mathbb{F}^{k_m}$, the distributions $\{e|_J \mid e = \mathrm{Enc}(m; r), r \leftarrow \mathbb{F}^{k_r}\}$ and $\{e'|_J \mid e' = \mathrm{Enc}(m'; r'), r' \leftarrow \mathbb{F}^{k_r}\}$ are identical. By Lemma 5.4, it suffices to show that for all vectors $z \in \mathbb{F}^J$, the random variables $\langle z, e|_J \rangle$ and $\langle z, e'|_J \rangle$ are identically distributed. We may pad $z \in \mathbb{F}^J$ with zeroes to obtain $z^* \in \mathbb{F}^n$ of weight at most B. By our hypothesis, the minimum weight codeword of $\mathcal{D}(\mathcal{C}) \setminus \mathcal{C}^\perp$ is $B+1$. Hence, any such z^*, which has weight at most B, is either not an element of $\mathcal{D}(\mathcal{C})$, or is an element of $\mathcal{C}^\perp$. In the first case, $z^* G_r \neq 0$. Setting $w^* = (w_m^*, w_r^*)$, defined similarly to w_m and w_r above, we see that $w_r^* \neq 0$ and that $\langle z, e|_J \rangle$ and $\langle z, e'|_J \rangle$ are both uniformly distributed over $\mathbb{F}$. In the second case, $w_r^* = z^* G_r = 0$ and $w_m^* = z^* G_m = 0$. We see that $w_r^* \ \langle z, e|_J \rangle$ and $\langle z, e'|_J \rangle$ are both equal to zero. Since m,m' and J were arbitrary, we conclude that $\mathcal{C}$ has B-query zero-knowledge.

Finally, it is clear that $\mathcal{C}$ is not $(B+1)$-query zero-knowledge because the weight $B+1$ codeword z^* of $\mathcal{D}(\mathcal{C}) \setminus \mathcal{C}^\perp$ that exists by hypothesis yields a relation $\langle z, e|_J \rangle = \langle w_m, m \rangle$ for J of size $B+1$ and non-zero w_m. This shows that one can distinguish between messages that have different values of $\langle w_m, m \rangle$.

5.3 Examples

We illustrate how the algebraic reformulations in this section relate to two well-known codes: the Reed–Solomon code (Section 5.3.1) and the Reed–Muller code (Section 5.3.2).

5.3.1 Example: Reed–Solomon code

Let k_m, k_r, k and n be positive integers such that $k = k_m + k_r$. Consider the randomized linear code $\mathcal{C}$ derived from the Reed–Solomon code $\mathcal{C} = \mathrm{RS}[n, k, n-k+1]$, whose encoding function $\mathrm{Enc}\colon \mathbb{F}^{k_m} \times \mathbb{F}^{k_r} \to \mathbb{F}^n$ is defined by generator matrix

$$G = \begin{bmatrix} G_r & G_m \end{bmatrix} = \left[\begin{array}{ccccc|ccc} 1 & \omega_1 & \omega_1^2 & \cdots & \omega_1^{k_r-1} & \omega_1^{k_r} & \cdots & \omega_1^{k-1} \\ 1 & \omega_2 & \omega_2^2 & \cdots & \omega_2^{k_r-1} & \omega_2^{k_r} & \cdots & \omega_2^{k-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega_n & \omega_n^2 & \cdots & \omega_n^{k_r-1} & \omega_n^{k_r} & \cdots & \omega_n^{k-1} \end{array}\right]$$

for (pre-determined) distinct points $\omega_1, \ldots, \omega_n \in \mathbb{F}$. Here, we have split the input to the encoding function into a k_r-element randomizer followed by a k_m-element message (for any desired k_m and k_r such that $k_m + k_r = k$). One can verify that, for every message $m \in \mathbb{F}^{k_m}$, if $r \in \mathbb{F}^{k_r}$ is random then any subset of k_r entries of $\mathrm{Enc}(m, r)$ are uniformly distributed. We now explain how to alternatively understand this through the lens of Lemma 5.2 and the code $\mathcal{D}(\mathcal{C})$.

Observe that G_r generates the Reed–Solomon code $\mathrm{RS}[n, k_r, n-k_r+1]$. Next observe that $\mathcal{D}(\mathcal{C})$ is its dual code and is straightforward to understand: the dual of an MDS code is also an MDS code (and so the distance of the dual is easy to compute) and, in particular, the dual of a Reed–Solomon code is also a Reed–Solomon code. One can thus verify that $\mathcal{D}(\mathcal{C}) = \mathrm{RS}[n, n-k_r, k_r+1]$. Since $\mathcal{D}(\mathcal{C})$ has minimum distance $k_r + 1$, Lemma 5.2 shows that $\mathcal{C}$ is k_r-query uniform. This agrees with our prior conclusion.

Furthermore, any subset of $k_r + 1$ entries of $\mathrm{Enc}(m, r)$ leak information about m. One can understand this via Lemma 5.1 and the set $\mathcal{D}(\mathcal{C}) \setminus \mathcal{C}^\perp$. Observe that $\mathcal{C}^\perp$ is the Reed-Solomon code $\mathrm{RS}[n, n-k, k+1]$ which has minimum distance greater than $k_r + 1$. This means that removing $\mathcal{C}^\perp$ from $\mathcal{D}(\mathcal{C})$ does not change the weight $k_r + 1$ of the minimum weight codeword, and so $\mathcal{C}$ is zero-knowledge for at most k_r queries.

5.3.2 Example: Reed–Muller code

Let ℓ, ℓ', m be positive integers such that $\ell < \ell' \leq m$. Let H be a subgroup of the multiplicative group of $\mathbb{F}_q$ with $|H| = h$. Consider the Reed-Muller code $\mathrm{RM}[H, m, \ell']$ over $\mathbb{F}_q$. That is, $\mathrm{RM}[H, m, \ell']$ is defined to be

$$\mathrm{RM}[H, m, \ell'] = \{(f(\vec{\omega}))_{\vec{\omega} \in H^m} \mid f \in \mathbb{F}_q[Z_1, \ldots, Z_m] \text{ with } \deg(f) \leq \ell' \text{ and } \forall i \in [m], \deg_{Z_i}(f) \leq h-1\}$$

which has block length $n = h^m$. The code $\mathrm{RM}[H, m, \ell']$ has a generator matrix G where column corresponds to a distinct monomial in $Z_1, \ldots, Z_m$, evaluated at all points in H^m, and the columns are ordered by graded lexicographic order of monomials. We may derive a randomised linear code $\mathcal{C}$ by assigning the random part of the input to the monomials of degree at most ℓ. This means that the matrix G_r is a submatrix of G which generates $\mathrm{RM}[H, m, \ell]$.

Let s, t be the unique non-negative integers such that $t < h-1$ and $\ell = s(h-1) + t$. The code $\mathrm{RM}[H, m, \ell]$ has minimum distance $(h-t) \cdot h^{m-s-1}$. The dual of the Reed–Muller code $\mathrm{RM}[H, m, \ell]$ is $\mathrm{RM}[H, m, m(h-1)-\ell-1]$, which is $\mathcal{D}(\mathcal{C})$. Further, $\mathcal{D}(\mathcal{C})$ has minimum distance $(t+2) \cdot h^s$. By Lemma 5.2, $\mathcal{C}$ is b-query uniform for $b < (t+2) \cdot h^s$.

Now we analyse the zero-knowledge property of $\mathcal{C}$ via Lemma 5.1 and the set $\mathcal{D}(\mathcal{C}) \setminus \mathcal{C}^\perp$. Let s', t' be the unique non-negative integers such that $t' < h-1$ and $\ell' = s'(h-1) + t'$. Observe that $\mathcal{C}^\perp$ is the Reed–Muller code $\mathrm{RM}[H, m, m(h-1)-\ell'-1]$ which has minimum distance $(t'+2) \cdot h^{s'}$. This is always strictly greater than the minimum distance $(t+2) \cdot h^s$ of $\mathcal{D}(\mathcal{C})$. Thus, removing $\mathcal{C}^\perp$ from $\mathcal{D}(\mathcal{C})$ does not change the weight of the minimum weight codeword, and so $\mathcal{C}$ is b-query zero-knowledge for $b < (t+2) \cdot h^s$ queries.

6 Tensor products of zero-knowledge codes

We define the tensor product $\mathcal{C} \otimes \mathcal{C}'$ of two randomized linear codes $\mathcal{C}$ and $\mathcal{C}'$. The definition matches the standard notion of tensor products in that the vector space $\mathcal{C} \otimes \mathcal{C}'$ is the tensor product of the vector spaces $\mathcal{C}$ and $\mathcal{C}'$. However, we additionally need to specify how the randomized encoding function of $\mathcal{C} \otimes \mathcal{C}'$ uses encoding randomness.

Definition 6.1. *Let $\mathcal{C}$ be a $[n, k_m, k_r, d]$ randomized linear code with encoding function $\mathrm{Enc}_{\mathcal{C}} \colon \mathbb{F}^{k_m} \times \mathbb{F}^{k_r} \to \mathbb{F}^n$. Let $\mathcal{C}'$ be a $[n', k'_m, k'_r, d']$ randomized linear code with encoding function $\mathrm{Enc}_{\mathcal{C}'} \colon \mathbb{F}^{k'_m} \times \mathbb{F}^{k'_r} \to \mathbb{F}^{n'}$. The encoding function $\mathrm{Enc}_{\mathcal{C}\otimes\mathcal{C}'}$ for the $[nn', k_m k'_m, k_m k'_r + k_r k'_m + k_r k'_r, dd']$ randomized linear code $\mathcal{C} \otimes \mathcal{C}'$ is defined by the following procedure:*

- *Extend the input message $m \in \mathbb{F}^{k_m \times k'_m}$ to the matrix $\tilde{m} \in \mathbb{F}^{(k_m+k_r)\times(k'_m+k'_r)}$ by padding with entries sampled independently and uniformly at random from $\mathbb{F}$.*
- *Output the codeword $c \in \mathbb{F}^{n \times n'}$ obtained by applying $\tilde{\mathrm{Enc}}_{\mathcal{C}}$ to each row of $\tilde{m}$ and then applying $\tilde{\mathrm{Enc}}_{\mathcal{C}'}$ to each column of the result.*

If $\mathcal{C}$ has generator matrix $\begin{bmatrix} G_m & G_r \end{bmatrix}$ and $\mathcal{C}'$ has generator matrix $\begin{bmatrix} G'_m & G'_r \end{bmatrix}$, then $\mathcal{C} \otimes \mathcal{C}'$ has generator matrix $\left[\begin{array}{c|ccc} G_m \otimes G'_m & G_m \otimes G'_r & G_r \otimes G'_m & G_r \otimes G'_r \end{array}\right]$, where the first part corresponds to the message and the last three parts correspond to the encoding randomness.

In this section we prove that the properties of bounded-query zero knowledge and bounded-query uniformity are preserved under the tensor operation.

Theorem 6.2. *Let $\mathcal{C}$ and $\mathcal{C}'$ be randomized linear codes.*

1. *If $\mathcal{C}$ is b-query zero-knowledge and $\mathcal{C}'$ is b'-query zero-knowledge, then $\mathcal{C} \otimes \mathcal{C}'$ is $\min(b, b')$-query zero-knowledge.*
2. *If $\mathcal{C}$ is b-query uniform and $\mathcal{C}'$ is b'-query uniform, then $\mathcal{C} \otimes \mathcal{C}'$ is $\min(b, b')$-query uniform.*

If we iteratively apply the above result to $\mathcal{C}$ and $\mathcal{C}' = \mathcal{C}^{\otimes i}$ for each $i \in [t-1]$ then we obtain this corollary:

Corollary 6.3. *Let $\mathcal{C}$ be a randomized linear code.*
1. *If $\mathcal{C}$ is b-query zero-knowledge, then, for any $t \in \mathbb{N}$, $\mathcal{C}^{\otimes t}$ is b-query zero-knowledge.*
2. *If $\mathcal{C}$ is b-query uniform, then, for any $t \in \mathbb{N}$, $\mathcal{C}^{\otimes t}$ is b-query uniform.*

We now proceed to the proof of Theorem 6.2. We start with some preliminaries and a structural result (Proposition 6.7), and then prove each item of the theorem.

Definition 6.4. *Let $\mathbb{F}$ be a field and $M \in \mathbb{F}^{m \times n}$ a matrix. We define Null(M) to be the left null-space of M:*

$$\textit{Null}(M) := \{x \in \mathbb{F}^m : x^{\intercal} M = 0\} \quad .$$

Lemma 6.5. *Let $f \colon V \to W$ and $f' \colon V' \to W'$ be surjective linear maps. Then*

$$\ker(f \otimes f') = \ker(f) \otimes V' + V \otimes \ker(f')$$

Proof sketch. We have $V = I \oplus K$, where $I \simeq W$ and $K = \ker f$, and similarly for V'. Thus, $f \otimes f'$ is a linear mapping from $V \otimes V'$ to $W \otimes W'$, where $V \otimes V'$ is equal to $(I \otimes I') \otimes (I \otimes K') \otimes (K \otimes I') \otimes (K \otimes K')$. Now, $f \otimes f'$ maps $I \otimes I'$ to $W \otimes W'$, and has kernel $(I \otimes K') \otimes (K \otimes I') \otimes (K \otimes K')$, which can be rewritten as $(K \otimes V') + (V \otimes K')$. □

Corollary 6.6. *For randomized linear codes $\mathcal{C}$ and $\mathcal{C}'$, $(\mathcal{C} \otimes \mathcal{C}')^\perp = \mathcal{C} \otimes \mathbb{F}^{n'} + \mathbb{F}^n \otimes \mathcal{C}'$.*

Proof. Setting $f(x) = x^\intercal G$ and $f'(x) = x^\intercal G'$ gives the result. The surjectivity of f and f' are guaranteed by the full rank of G and G'. □

Proposition 6.7. *For randomized linear codes $\mathcal{C}$ and $\mathcal{C}'$,*

$$\mathcal{D}(\mathcal{C} \otimes \mathcal{C}') = (\mathcal{C} \otimes \mathcal{C}')^\perp + \mathcal{D}(\mathcal{C}) \otimes \mathcal{D}(\mathcal{C}') = \mathcal{C} \otimes \mathbb{F}^{n'} + \mathbb{F}^n \otimes \mathcal{C}' + \mathcal{D}(\mathcal{C}) \otimes \mathcal{D}(\mathcal{C}')$$

Proof. The spaces $\mathcal{D}(\mathcal{C} \otimes \mathcal{C}')$ and $(\mathcal{C} \otimes \mathcal{C}')^\perp$ are characterised explicitly as follows.

$$\mathcal{D}(\mathcal{C} \otimes \mathcal{C}') = \left\{ z \in \mathbb{F}^{n \cdot n'} : z^\intercal \cdot (G_r \otimes G'_m) = 0 \ \wedge \ z^\intercal \cdot (G_m \otimes G'_r) = 0 \ \wedge \ z^\intercal \cdot (G_r \otimes G'_r) = 0 \right\},$$

$$(\mathcal{C} \otimes \mathcal{C}')^\perp = \left\{ z \in \mathbb{F}^{n \cdot n'} : z^\intercal \cdot (G \otimes G') = 0 \right\} \ .$$

It is clear that $\mathcal{D}(\mathcal{C} \otimes \mathcal{C}')$ contains both $(\mathcal{C} \otimes \mathcal{C}')^\perp$ and $\mathcal{D}(\mathcal{C}) \otimes \mathcal{D}(\mathcal{C}')$, and so $(\mathcal{C} \otimes \mathcal{C}')^\perp + \mathcal{D}(\mathcal{C}) \otimes \mathcal{D}(\mathcal{C}') \subseteq \mathcal{D}(\mathcal{C} \otimes \mathcal{C}')$.

Now we show that $\mathcal{D}(\mathcal{C} \otimes \mathcal{C}') \subseteq (\mathcal{C} \otimes \mathcal{C}')^\perp + \mathcal{D}(\mathcal{C}) \otimes \mathcal{D}(\mathcal{C}')$.

By definition, $\mathcal{C}^\perp \subseteq \mathcal{D}(\mathcal{C}) \subseteq \mathbb{F}^n$. Given a basis $A = \{a_i\}_{i \in I}$ for $\mathcal{C}^\perp$, we may extend it to a basis $A \cup B$ for $\mathcal{D}(\mathcal{C})$ by adding $B = \{b_j\}_{j \in J}$ such that no element b_j is contained in $\mathcal{C}^\perp$, and then again to a basis $A \cup B \cup C$ for $\mathbb{F}^n$ by adding $C = \{c_k\}_{k \in K}$ such that no element c_k is contained in $\mathcal{D}(\mathcal{C})$. The same applies to $(\mathcal{C}')^\perp \subseteq \mathcal{D}(\mathcal{C}') \subseteq \mathbb{F}^{n'}$. Considering the pairwise tensor products of basis elements for $\mathbb{F}^n$ and $\mathbb{F}^{n'}$ gives a basis of rank-2 tensors for $\mathbb{F}^{n \cdot n'}$.

It suffices to show that any rank-2 basis element $z \otimes z' \in \mathcal{D}(\mathcal{C} \otimes \mathcal{C}')$ is in $(\mathcal{C} \otimes \mathcal{C}')^\perp + \mathcal{D}(\mathcal{C}) \otimes \mathcal{D}(\mathcal{C}')$. If $z \otimes z' \in \mathcal{D}(\mathcal{C} \otimes \mathcal{C}')$, then by definition of $\mathcal{D}(\mathcal{C} \otimes \mathcal{C}')$, we have $(z^\intercal G_r) \otimes (z'^\intercal G'_m) = 0$, $(z^\intercal G_m) \otimes (z'^\intercal G'_r) = 0$ and $(z^\intercal G_r) \otimes (z'^\intercal G'_r) = 0$. Thus, at least one of $z^\intercal G_r$ and $z'^\intercal G'_r$ must be equal to zero. If both are zero, then $z \otimes z' \in \mathcal{D}(\mathcal{C}) \otimes \mathcal{D}(\mathcal{C}')$.

On the other hand, if, without loss of generality, $z^\intercal G_r = 0$ and $z'^\intercal G'_r \neq 0$, then since $(z^\intercal G_m) \otimes (z'^\intercal G'_r) = 0$, we must have $z^\intercal G_m = 0$. Then, as $z^\intercal G_m = 0$ and $z^\intercal G_r = 0$, we have $z \in \mathcal{C}^\perp$, and so $z \otimes z' \in \mathcal{C}^\perp \otimes \mathbb{F}^{n'}$. By Corollary 6.6, we have $(\mathcal{C} \otimes \mathcal{C}')^\perp = \mathcal{C}^\perp \otimes \mathbb{F}^{n'} + \mathbb{F}^n \otimes (\mathcal{C}')^\perp$, and so $z \otimes z' \in (\mathcal{C} \otimes \mathcal{C}')^\perp$.

Therefore, $\mathcal{D}(\mathcal{C} \otimes \mathcal{C}') \subseteq (\mathcal{C} \otimes \mathcal{C}')^\perp + \mathcal{D}(\mathcal{C}) \otimes \mathcal{D}(\mathcal{C}')$, which completes the proof. □

Proof of Item 2 in Theorem 6.2. Using the algebraic characterisation of b-query uniformity from Lemma 5.2, it suffices to prove that if $\mathcal{D}(\mathcal{C})$ has minimum distance at least $b+1$, and $\mathcal{D}(\mathcal{C}')$ has minimum distance at least $b'+1$, then $\mathcal{D}(\mathcal{C} \otimes \mathcal{C}')$ has minimum distance at least $\min(b, b') + 1$. Without loss of generality, suppose that $b \leq b'$. We show that $\mathcal{D}(\mathcal{C} \otimes \mathcal{C}')$ has minimum distance at least $b+1$.

Let $R := \mathcal{D}(\mathcal{C}) \otimes \mathbb{F}^{n'} + \mathbb{F}^n \otimes \mathcal{D}(\mathcal{C}')$. By Proposition 6.7, we have $\mathcal{D}(\mathcal{C} \otimes \mathcal{C}') = (\mathcal{C} \otimes \mathcal{C}')^\perp + \mathcal{D}(\mathcal{C}) \otimes \mathcal{D}(\mathcal{C}')$. Since $\mathcal{C}^\perp \subseteq \mathcal{D}(\mathcal{C})$ and $\mathcal{C}'^\perp \subseteq \mathcal{D}(\mathcal{C}')$, we have $\mathcal{D}(\mathcal{C} \otimes \mathcal{C}') \subseteq R$. This means that the minimum distance of R is a lower bound for the minimum distance of $\mathcal{D}(\mathcal{C} \otimes \mathcal{C}')$, so it suffices to show that the minimum distance of R is at least $b+1$. We will demonstrate this by showing that each element of R with at most b non-zero entries must be equal to zero. This implies that each non-zero element $z \in R$ has weight at least $b+1$.

Writing $\mathbb{F}^{n'}$ as a direct sum $\mathbb{F}^{n'} = \mathcal{D}(\mathcal{C}') \oplus I(\mathcal{C}')$, we have $R = \mathcal{D}(\mathcal{C}) \otimes I(\mathcal{C}') + \mathbb{F}^n \otimes \mathcal{D}(\mathcal{C}')$. We may therefore express any $z \in R$ as the difference $z = z_1 - z_2$, for some $z_1 \in \mathcal{D}(\mathcal{C}) \otimes I(\mathcal{C}')$ and $z_2 \in \mathbb{F}^n \otimes \mathcal{D}(\mathcal{C}')$. Note that each row of z_1 lies in $\mathcal{D}(\mathcal{C})$, each column of z_1 lies in $I(\mathcal{C}')$, and each column of z_2 lies in $\mathcal{D}(\mathcal{C}')$.

Suppose that $z \in R$ has at most b non-zero elements. Write $z = z_1 - z_2$ as above. At least $n - b$ columns of z are equal to zero. Hence, z_1 and z_2 are equal on these $n - b$ columns. Since each column of z_2 lies in $\mathcal{D}(\mathcal{C}')$ then these $n - b$ columns of z_1 lie in $\mathcal{D}(\mathcal{C}')$, and hence in $\mathcal{D}(\mathcal{C}') \cap I(\mathcal{C}')$, which is $\{0\}$. This means that

$n - b$ columns of z_1 are equal to zero, and that each row of z_1 contains at most b non-zero entries. Since the minimum distance of $\mathcal{D}(\mathcal{C})$ is at least $b+1$, each row of z_1 is zero, so z_1 is zero. Thus, $z = z_2 \in \mathbb{F}^n \otimes \mathcal{D}(\mathcal{C}')$. But since the minimum distance of $\mathbb{F}^n \otimes \mathcal{D}(\mathcal{C}')$ is at least $b' + 1$, which is at least $b + 1$, we see that $z = 0$.

This implies that each non-zero element $z \in R$ has weight at least $b + 1$, completing the proof. □

Proof of Item 1 in Theorem 6.2. Using the algebraic characterisation of b-query zero-knowledge from Lemma 5.1, it suffices to prove that if $\mathcal{D}(\mathcal{C}) \setminus \mathcal{C}^\perp$ has minimum weight at least $b + 1$, and $\mathcal{D}(\mathcal{C}') \setminus (\mathcal{C}')^\perp$ has minimum weight at least $b' + 1$, then $\mathcal{D}(\mathcal{C} \otimes \mathcal{C}') \setminus (\mathcal{C} \otimes \mathcal{C}')^\perp$ has minimum weight at least $\min(b, b') + 1$. Without loss of generality, suppose that $b \leq b'$. We show that $\mathcal{D}(\mathcal{C} \otimes \mathcal{C}') \setminus (\mathcal{C} \otimes \mathcal{C}')^\perp$ has minimum weight at least $b + 1$.

Use $N(\mathcal{C})$ to denote the linear subspace orthogonal to $\mathcal{C}^\perp$ such that $\mathcal{D}(\mathcal{C}) = \mathcal{C}^\perp \oplus N(\mathcal{C})$. We observe that since $\mathbb{F}^n = \mathcal{C}^\perp \oplus \mathcal{C}$ and $\mathcal{D}(\mathcal{C}) \subset \mathbb{F}^n$, so $N(\mathcal{C}) \subset \mathcal{C}$. Define a new linear space

$$R := \left(\mathcal{C}^\perp \otimes \mathbb{F}^{n'} + \mathcal{C} \otimes (\mathcal{C}')^\perp\right) + \mathcal{C} \otimes N(\mathcal{C}') \setminus \left(\mathcal{C}^\perp \otimes \mathbb{F}^{n'} + \mathcal{C} \otimes (\mathcal{C}')^\perp\right) \ .$$

By Proposition 6.7, we have $\mathcal{D}(\mathcal{C} \otimes \mathcal{C}') = (\mathcal{C} \otimes \mathcal{C}')^\perp + \mathcal{D}(\mathcal{C}) \otimes \mathcal{D}(\mathcal{C}')$. Since $\mathcal{D}(\mathcal{C}) = \mathcal{C}^\perp \oplus N(\mathcal{C})$ and $\mathcal{D}(\mathcal{C}') = \mathcal{C}'^\perp \oplus N(\mathcal{C}')$, we have $\mathcal{D}(\mathcal{C} \otimes \mathcal{C}') = \mathcal{C}^\perp \otimes \mathbb{F}^{n'} + \mathcal{C} \otimes \mathcal{C}'^\perp + N(\mathcal{C}) \otimes N(\mathcal{C}')$. Additionally since $N(\mathcal{C}) \subset \mathcal{C}$, $\mathcal{D}(\mathcal{C} \otimes \mathcal{C}') \setminus (\mathcal{C} \otimes \mathcal{C}')^\perp \subset R$.

This means that the minimum weight of R is a lower bound for the minimum weight of $\mathcal{D}(\mathcal{C} \otimes \mathcal{C}') \setminus (\mathcal{C} \otimes \mathcal{C}')^\perp$, so it suffices to show that the minimum weight of R is at least $b + 1$.

We may therefore express any $z \in R$ as the difference $z = z_1 - z_2$, for some $z_1 \in \mathcal{C}^\perp \otimes \mathbb{F}^{n'}$ and $z_2 \in \mathcal{C} \otimes \mathcal{D}(\mathcal{C}') \setminus \mathcal{C} \otimes (\mathcal{C}')^\perp$. Note that each row of z_1 lies in $\mathcal{C}^\perp$, each row of z_2 lies in $\mathcal{C}$, and there exists a column of z_2 that lies in $\mathcal{D}(\mathcal{C}') \setminus (\mathcal{C}')^\perp$.

Suppose that $z \in R$ has at most b non-zero elements. Write $z = z_1 - z_2$ as above. At least $n - b$ rows of z are equal to zero. Hence, z_1 and z_2 are equal on these $n - b$ rows. Since each row of z_1 lies in $\mathcal{C}^\perp$ then these $n - b$ rows of z_2 lie in $\mathcal{C}^\perp$, and hence in $\mathcal{C}^\perp \cap \mathcal{C}$, which is $\{0\}$. This means that $n - b$ rows of z_2 are equal to zero, and that each column of z_2 contains at most b non-zero entries. Since the minimum distance of $\mathcal{D}(\mathcal{C}') \setminus \mathcal{C}'^\perp$ is at least $b + 1$, no column of z_2 lies in $\mathcal{D}(\mathcal{C}') \setminus \mathcal{C}'^\perp$, so $z = z_1 - z_2$ cannot be in R. We derive a contradiction.

This implies that each element $z \in R$ has weight at least $b + 1$, completing the proof. □

7 Zero-knowledge codes with linear-time encoding

We state and prove our result about the existence of linear codes that are both linear-time encodable and b-query zero knowledge (in fact, b-query uniform).

Definition 7.1. *A* **field-agnostic circuit** *over variables $x_1, \ldots, x_n$ is a directed acyclic graph such that (i) every node with in-degree 0 is labeled by a variable x_i; (ii) every other gate is labelled by either $+$ or $\times$.*[17]

Definition 7.2. $H_q(x) := x \log_q(q-1) - x \log_q(x) - (1-x)\log_q(1-x)$ *is the q-ary entropy function.*

Theorem 7.3. *For every choice of rate parameters $\rho, \rho_m, \rho_r \in (0,1)$ with $\rho = \rho_m + \rho_r$, there is a polynomial-time constructible field-agnostic circuit family $\{E_k\}_{k\in\mathbb{N}}$ where E_k has size $O(k)$, $k + \nu(k)$ inputs and n outputs, with $\nu(k) = O(k)$ and $k = \lfloor \rho n \rfloor$ such that for any finite field $\mathbb{F}$ (of size q) the following holds:*

- *for every $R \in \mathbb{F}^{\nu(k)}$, $E_k(\cdot, R)$ is an injective linear encoding function with message length k and block length n, which defines a randomised linear code $\mathcal{C}_k$ with $k_r = \lceil \rho_r n \rceil$ and $k_m = k - k_r$;*
- *for every $\epsilon > 0$,*

$$\Pr_{R \leftarrow \mathbb{F}^{\nu(k)}} \Big[\text{the relative distance of } \mathcal{C}_k \text{ is greater than } H_q^{-1}(1-\rho-\epsilon)\Big] \geq 1 - q^{1-\epsilon n} \ .$$

- *for every $\epsilon > 0$,*

$$\Pr_{R \leftarrow \mathbb{F}^{\nu(k)}} \Big[\mathcal{C}_k \text{ is } b\text{-query uniform with } (b+1)/n > H_q^{-1}(\rho_r - \epsilon)\Big] \geq 1 - q^{-\epsilon n} \ .$$

In Section 7.1 we introduce preliminaries and then in Section 7.2 we prove the theorem.

7.1 Preliminaries

We recall several definitions and results about linear uniform output families (LUO families) from Druk and Ishai [DI14]. Let $\mathbb{F}$ be a finite field with q elements.

Definition 7.4. *A $(\mathbb{F}, k, n)$-***LUO family** *is a distribution $\mathcal{A}$ over linear functions $A\colon \mathbb{F}^k \to \mathbb{F}^n$ such that for every non-zero $x \in \mathbb{F}^k$ and every $y \in \mathbb{F}^n$ it holds that $\Pr_{A\leftarrow\mathcal{A}}[Ax = y] = q^{-n}$. Equivalently, for every fixed non-zero $x \in \mathbb{F}^k$, the distribution of Ax induced by a choice of A according to $\mathcal{A}$ is uniform over $\mathbb{F}^n$.*

Lemma 7.5 (transposition)**.** *If $\mathcal{A}$ is a $(\mathbb{F}, k, n)$-LUO family then $\mathcal{A}^\intercal := \{A^\intercal \mid A \leftarrow \mathcal{A}\}$ is a $(\mathbb{F}, n, k)$-LUO family. Namely, for every non-zero $y \in \mathbb{F}^n$ and every $x \in \mathbb{F}^k$ it holds that $\Pr_{A\leftarrow\mathcal{A}}[A^\intercal y = x] = q^{-k}$.*

Theorem 7.6. *For every parameter $c \in \mathbb{N}$ there is a polynomial-time constructible family $\{E_k\}_{k\in\mathbb{N}}$ of field-agnostic circuits where E_k has size $O(k)$, $k + \nu(k)$ inputs and ck outputs, such that for any field $\mathbb{F}$, viewing E_k as a function $E_k\colon \mathbb{F}^k \times \mathbb{F}^{\nu(k)} \to \mathbb{F}^{ck}$, the function family $\{A_R(x) := E_k(x,R)\}_{R\in\mathbb{F}^{\nu(k)}}$ is a $(\mathbb{F}, k, ck)$-LUO family when R is chosen uniformly in $\mathbb{F}^{\nu(k)}$.*

Proof. The circuit family $E_k\colon \mathbb{F}^k \times \mathbb{F}^{\nu(k)} \to \mathbb{F}^{ck}$ of [DI14] provides a suitable LUO family. It remains to prove that the circuits E_k are field-agnostic. The E_k are given by a composition of circuits as $E_k(x,R) = f_k \circ M_R \circ g_k$. The circuit g_k is the encoding function for a recursive construction of a variant of Spielman codes, and for a suitable choice of base code, g_k uses only addition and fan-out gates. The circuit f_k is the

[17] A standard arithmetic circuit is defined over a field $\mathbb{F}$, and nodes with in-degree 0 can be labelled by elements of $\mathbb{F}$ as well as variables. A field-agnostic circuit can be viewed as a circuit over any field $\mathbb{F}$.

transpose (see e.g. [KKB88]) of g_k obtained by reversing the direction of the circuit for g_k and thus also consists only of addition and fan-out gates. Finally, M_R is left-multiplication by a block-diagonal matrix with entries specified by R, and it is clear that this circuit can be implemented as stated since R is an input to the circuit. □

Theorem 7.7 (good codes from LUO)**.** *Let $\mathbb{F}$ be a finite field, $\delta \in [0, 1 - 1/q)$ a distance parameter, $\epsilon > 0$ an error parameter, and $n \in \mathbb{N}$ a block length; define the message length $k := \lceil (1 - H_q(\delta) - \epsilon) \cdot n \rceil$. If $\mathcal{A}$ is a $(\mathbb{F}, k, n)$-LUO family then the image of $A \colon \mathbb{F}^k \to \mathbb{F}^n$ sampled from $\mathcal{A}$ is a linear code with rate k/n and relative distance strictly greater than δ, except with probability at most $q^{1-\epsilon n}$ over the choice of A.*

Theorem 7.8 (good dual codes from LUO)**.** *Let $\mathbb{F}$ be a finite field, $\epsilon > 0$ an error parameter, k a message length and $n \in \mathbb{N}$ a block length. Let $\mathcal{A}$ be a $(\mathbb{F}, k, n)$-LUO family. If $A \colon \mathbb{F}^k \to \mathbb{F}^n$ sampled from $\mathcal{A}$ defines a code $\mathcal{C}$, then the kernel of A is the dual code $\mathcal{C}^\perp$ with rate $1 - k/n$ and relative distance strictly greater than $H_q^{-1}(k/n - \epsilon)$, except with probability at most $q^{-\epsilon n}$ over the choice of A.*

7.2 Proof of Theorem 7.3

Lemma 7.9. *Let $\mathbb{F}$ be a finite field with q elements. Let $\mathcal{A}$ be a $(\mathbb{F}, k, n)$-LUO family. Let $\epsilon > 0$ be an error parameter, $\rho_r \in (0, 1)$ be a randomness rate and $b \in \mathbb{N}$ be a bound such that $\rho_r \leq k/n$ and $\rho_r \geq H_q((b+1)/n) + \epsilon$. Then, for $A \leftarrow \mathcal{A}$, the associated randomized linear code $\mathcal{C} = (A, G, G^+, H)$ with $k_r = \lceil \rho_r n \rceil$ is b-query uniform, except with probability at most $q^{-\epsilon n}$ over the choice of A.*

Proof. The code $\mathcal{C}$ has generator matrix $G = \begin{bmatrix} G_m & G_r \end{bmatrix}$. We give a lower bound on the minimum distance of $\mathcal{D}(\mathcal{C})$ that holds with high probability over the random choice of $A \in \mathcal{A}$. Recall that $\mathcal{D}(\mathcal{C}) = \{z \in \mathbb{F}^n \mid z^\intercal G_r = 0\}$, so if $\mathcal{C}$ is induced by the LUO family $\mathcal{A}$ then $\mathcal{D}(\mathcal{C})$ is induced by the LUO family $\mathcal{A}^\intercal|_{[k]\setminus[k_m]} := \{A^\intercal|_{[k]\setminus[k_m]} \,|\, A \leftarrow \mathcal{A}\}$. This is indeed a LUO family by Lemma 7.5 and the fact that truncating the output preserves the LUO property. By Theorem 7.8, $\mathcal{D}(\mathcal{C})$ has relative distance strictly greater than $(b+1)/n = H_q^{-1}(\rho_r - \epsilon)$ except with probability at most $1 - q^{-\epsilon n}$. Hence, by Lemma 5.2, $\mathcal{C}$ is b-query uniform. □

Remark 7.10. Lemma 7.9 gives a lower bound on the amount of randomness needed for a randomized linear code to achieve b-query uniformity. This lower bound is essentially the same as the bound appearing in the probabilistic construction of zero-knowledge codes from [Wei16, Theorem 6.3], rewritten using the q-ary entropy function rather than the binary entropy function. This is because both results make use of the same condition on $\mathcal{D}(\mathcal{C})$ appearing in Lemma 5.2.

Our code construction is, however, different. The construction of [Wei16] takes any fixed code $\mathcal{C}$ with generator matrix G, and constructs a new generator matrix G by sampling a matrix M which satisfies certain constraints, and setting $G' := GM^{-1}$. When encoding is performed using the generator matrix G', then the result is a randomized linear code that is b-query uniform. This does not guarantee any linear-time encoding properties. For this reason, our construction in Lemma 7.9 shows that the condition of Lemma 5.2 is satisfied with high probability over a random code induced by a LUO family $\mathcal{A}$.

More concretely, the construction of [Wei16] analyzes $\mathcal{D}(\mathcal{C})$ defined by the kernel of G_r^* for $GM^{-1} = \begin{bmatrix} G_m^* & G_r^* \end{bmatrix}$ for a randomly generated M, and G fixed, whereas we analyze the kernel of G_r over the choice of G induced by a random choice of function from a LUO family.

Remark 7.11. Besides a probabilistic construction, [Wei16] also gives a fully explicit construction of zero-knowledge codes. It is an interesting open problem to give a fully explicit construction of linear-time encodable codes with comparable b-query uniformity.

Proof of Theorem 7.3. We discuss each of the items in the theorem in turn.

- By Theorem 7.6, there exists a polynomial-time constructible (q, k, n)-LUO family with $k = \lfloor \rho n \rfloor$ (after truncating and padding the input and output if necessary) given by $E_k \colon \mathbb{F}^k \times \mathbb{F}^{\nu(k)} \to \mathbb{F}^n$ (so that each member of the family is given by $E(\cdot; R)$ for some $R \in \mathbb{F}^{\nu(k)}$).
- By Theorem 7.7, we know that setting $\mathcal{C} = E_k(\cdot, R)$ for random R gives a linear code with relative distance at least $H_q^{-1}(1-\rho) - \epsilon$, except with probability at most $q^{1-\epsilon n}$. Given E_k, one can efficiently compute the public parameters for $\mathcal{C}$.
- By Lemma 5.2 and Lemma 7.9, $\mathcal{C}$ is b-query uniform with b at least $H_q^{-1}(\rho_r - \epsilon)$ except with probability at most $q^{-\epsilon n}$.

□

7.3 Setting parameters

Performance parameters. We now explain how to choose parameters in Theorem 7.3 to prove Theorem 6.

Fix a field $\mathbb{F}$ of size q, a constant $\epsilon \in (0,1)$ and a constant K bounded away from 1, and consider the problem of constructing a family of random encoding circuits whose input size is denoted by k_m. Find constants $\delta, \epsilon_r \in (0,1)$ that satisfy

$$1 - H_q(\delta) - H_q(K) - \epsilon_r - \epsilon > 0 \ . \tag{1}$$

Set the randomness rate $\rho_r \coloneqq H_q(K) + \epsilon_r$ and the total rate $\rho \coloneqq 1 - H_q(\delta) - \epsilon$. By definition this ensures that the message rate $\rho_m = \rho - \rho_r > 0$. Then set the query bound $b \leq K \cdot k_m/(\rho - \rho_r)$. Finally set the code length $n = k_m/\rho_m$ and the total message length $k = \rho n$.

Now we apply Theorem 7.3 to the parameters $\mathbb{F}, k, \rho, \rho_m, \rho_r, \epsilon_r, b$, to obtain a family of random encoding circuits with input randomness length

$$k_r = \frac{H_q(K) + \epsilon_r}{1 - H_q(\delta) - H_q(K) - \epsilon_r - \epsilon} k_m \ , \tag{2}$$

and output length,

$$n = \frac{1}{1 - H_q(\delta) - H_q(K) - \epsilon_r - \epsilon} k_m \ . \tag{3}$$

The failure to achieve relative distance at least δ occurs with probability at most $q^{1-\epsilon n}$.

The failure to achieve b-query uniformity occurs with probability at most $q^{-\epsilon_r n}$.

Obtaining Theorem 6. Suppose that $\beta \colon \mathbb{N} \to (0,1)$ is bounded away from 1, so that there exists some constant $K < 1$ with $\beta(k_m) \leq K$ for all k_m. This means that $H_q(\beta(k_m))$ is bounded away from 1. Thus, it is possible to choose ϵ_r and δ in $(0,1)$ satisfying Equation (1) for any value of k_m.

Consider Equation (2) and Equation (3). Since β is bounded away from 1, we see that $1 - H_q(\delta) - H_q(\beta) - \epsilon_r - \epsilon$ is bounded below by a constant, and so $k_r = (H_q(\beta) + \epsilon_r) \cdot O(k_m)$ and $n = O(k_m)$.

Continuing this reasoning implies that the randomness used to generate the code and the code length are both linear in k_m.

8 From tensor queries to point queries with zero knowledge

Definition 8.1. *The indexed relation R_{cons} is the set of tuples*

$$(\mathbb{i}, \mathbb{x}, \mathbb{w}) = \left(\perp, (\mathbb{F}, \mathcal{C}, \ell, \mathsf{q}, t, \{q^{(s)}\}_s, \{v_s\}_s), c\right)$$

such that $c = \mathrm{Enc}_{\mathcal{C}^{\otimes t}}(f) \in \mathbb{F}^{\ell \cdot n^t}$ for some $f \in \mathbb{F}^{\ell \cdot k^t}$, for each $s \in [\mathsf{q}]$, $q^{(s)} = (q_0^{(s)}, \ldots, q_t^{(s)}) \in \mathbb{F}^\ell \times \left(\mathbb{F}^k\right)^t$, and for all $s \in [\mathsf{q}]$, $\langle \otimes_i q_i^{(s)}, f \rangle = v_s$.

Lemma 8.2. *There exists an explicit polynomial-time transformation T that satisfies the following. The inputs to the transformation are as follows:*

- *A randomised t-linear tensor code $\mathcal{C}^{\otimes t}$ over $\mathbb{F}$ with encoding function $\tilde{\mathrm{Enc}}_{\mathcal{C}^{\otimes t}}$, which has rate $\rho_m = \frac{k_m^t}{n^t}$, randomness rate $\rho_r = \frac{k^t - k_m^t}{n^t}$, relative distance $\delta = \frac{d^t}{n^t}$, and encoding arithmetic complexity $\theta(k_m) \cdot k_m^t$.*
- *A holographic interactive oracle proof $\mathsf{IOP} = (\mathbf{I}, \mathbf{P}, \mathbf{V})$ with queries in $\mathcal{Q}_{\mathrm{tensor}}(\mathbb{F}, k_m, t)$ for an indexed relation R with: soundness error ϵ; round complexity rc; proof length $\mathsf{l} = \mathsf{li} + \mathsf{lp} = \ell k_m^t$; query complexity q; arithmetic complexity ti for the indexer; arithmetic complexity tp for the prover; and arithmetic complexity tv for the verifier; semi-honest-verifier zero-knowledge.*
- *An interactive oracle proof of proximity $\mathsf{IOPP} = (\mathbf{P}', \mathbf{V}')$ with queries in $\mathcal{Q}_{\mathrm{point}}$ for the indexed relation $R_{\mathsf{cons}}(\mathbb{F}, \mathcal{C}, \ell, \mathsf{q}, t)$ with soundness error ϵ'; round complexity rc'; proof length l'; input query complexity $\mathsf{q}'_{\mathbb{x}}$, the number of verifier queries to the input oracle; proof query complexity q'_π, the number of verifier queries to the IOPP proof oracles; arithmetic complexity tp' for the prover; arithmetic complexity tv' for the verifier.*

The output of the transformation $(\hat{\mathbf{I}}, \hat{\mathbf{P}}, \hat{\mathbf{V}}) := T(\mathcal{C}, \mathsf{IOP}, \mathsf{IOPP})$ is an interactive oracle proof with queries in $\mathcal{Q}_{\mathrm{point}}$ for the indexed relation R with the following parameters:

- *soundness error* $\max(\epsilon, (1 - 1/|\mathbb{F}|)\epsilon'(1/4) + 1/|\mathbb{F}|)$ *;*
- *round complexity* $\mathsf{rc} + \mathsf{rc}' + 1$*;*
- *proof length* $O\left(\frac{n^t}{k_m^t} \cdot \mathsf{l}\right) + \mathsf{l}'$*;*
- *query complexity* $2\mathsf{q}'_{\mathbb{x}} + \mathsf{q}'_\pi$*, where* $2\mathsf{q}'_{\mathbb{x}}$ *queries are to the* IOP *proof oracles, and* q'_π *queries are to the* IOPP *proof oracles;*
- *indexer time* $\mathsf{ti} + \mathsf{ti}'$*;*
- *prover time* $\mathsf{tp} + \mathsf{tp}'$*; and*
- *verifier time* $\mathsf{tv} + \mathsf{tv}'$*.*

Moreover,

1. *if the interleaved code $\left(\mathcal{C}^{\otimes t}\right)^\ell$ is $\mathbf{Q}$-query zero knowledge and the honest IOPP verifier $\mathbf{V}'$'s witness queries are compatible with $\mathbf{Q}$, then $(\hat{\mathbf{I}}, \hat{\mathbf{P}}, \hat{\mathbf{V}})$ is zero knowledge against semi-honest verifiers;*
2. *if $\left(\mathcal{C}^{\otimes t}\right)^\ell$ is b-query zero knowledge, $(\hat{\mathbf{I}}, \hat{\mathbf{P}}, \hat{\mathbf{V}})$ is zero knowledge against all verifiers making at most b queries to the IOP oracles.*

Finally, if inputs are public-coin protocols then the output of the transformation is also a public-coin protocol.

Remark 8.3. The round complexity is $\mathsf{rc} + \mathsf{rc}' + 1$ in the case that the verifier $\mathbf{V}$ makes non-adaptive queries, because then $\hat{\mathbf{V}}$ can handle all of the queries in a single round. In the case where the q queries are made as adaptively as possible, the round complexity would be $\mathsf{rc} + \mathsf{rc}' + \mathsf{q}$. If $\mathbf{V}$ is public-coin, then $\hat{\mathbf{P}}$ already knows all of the queries and the round complexity would be $\mathsf{rc} + \mathsf{rc}'$.

Remark 8.4. The stated time for the (honest) prover $\hat{\mathbf{P}}$ assumes that $\hat{\mathbf{P}}$ has access to the output of the indexer $\mathbf{I}$, since $\hat{\mathbf{P}}$ must compute and send the answers to tensor queries involving the output of $\mathbf{I}$.

8.1 Construction

Though Definition 3.5 specifies that tensor queries are of the form $q(\mathbb{x}, \Pi_0, \ldots, \Pi_i) = \langle q_0 \otimes q_1 \otimes \cdots \otimes q_t, \Pi_j \rangle$, our transformation below handles tensor queries of the form $\langle q_0 \otimes q_1 \otimes \cdots \otimes q_t, \mathrm{stack}(\Pi_0, \ldots, \Pi_{\mathsf{rc}}) \rangle$, which are more general.

Construction 8.5. We describe the construction of $(\hat{\mathbf{I}}, \hat{\mathbf{P}}, \hat{\mathbf{V}})$. The new indexer $\hat{\mathbf{I}}$, given an index $\mathbb{i}$, runs $\mathbf{I}$ on $\mathbb{i}$ to produce $\Pi_0 \in \mathbb{F}^{\ell_0 \cdot k_m^t}$, pads Π_0 with zeroes for encoding randomness to get $\tilde{\Pi}_0 \in \mathbb{F}^{\ell_0 \cdot k^t}$, computes $\hat{\Pi}_0 := \tilde{\mathrm{Enc}}_{\mathcal{C}^{\otimes t}}(\tilde{\Pi}_0) \in \mathbb{F}^{\ell_0 \cdot n^t}$, and outputs $\hat{\Pi}_0$. The new prover $\hat{\mathbf{P}}$ receives as input the index $\mathbb{i}$, an instance $\mathbb{x}$, and a witness $\mathbb{w}$, while the new verifier $\hat{\mathbf{V}}$ receives as oracle the encoded proof $\hat{\Pi}_0$ and as input the instance $\mathbb{x}$. The new prover $\hat{\mathbf{P}}$ and the new verifier $\hat{\mathbf{V}}$ interact in two phases, a simulation phase and a consistency phase. We describe each in turn.

(1) Simulation phase. For each $i \in [\mathsf{rc}]$, $\hat{\mathbf{P}}$ and $\hat{\mathbf{V}}$ simulate the i-th round of the interaction between $\mathbf{P}(\mathbb{i}, \mathbb{x}, \mathbb{w})$ and $\mathbf{V}^{\mathbf{I}(\mathbb{i})}(\mathbb{x})$, as well as any tensor queries by $\mathbf{V}$ to the received proof strings.

1. *Prover messages.* For any proof message $\Pi_i \in \mathbb{F}^{\ell_i \cdot k_m^t}$ that it receives from $\mathbf{P}$, $\hat{\mathbf{P}}$ pads Π_i with uniformly-sampled encoding randomness to get $\tilde{\Pi}_i \in \mathbb{F}^{\ell_i \cdot k^t}$, computes a new proof message $\hat{\Pi}_i := \tilde{\mathrm{Enc}}_{\mathcal{C}^{\otimes t}}(\tilde{\Pi}_i) \in \mathbb{F}^{\ell_i \cdot n^t}$, and sends $\hat{\Pi}_i$ to $\hat{\mathbf{V}}$. Also, $\hat{\mathbf{P}}$ forwards any non-oracle messages from $\mathbf{P}$ to $\mathbf{V}$ via $\hat{\mathbf{V}}$.

2. *Verifier messages.* $\hat{\mathbf{V}}$ receives challenge message ρ_i from $\mathbf{V}$ and forwards it to $\hat{\mathbf{P}}$, who forwards it to $\mathbf{P}$.

3. *Tensor queries.* If $\hat{\mathbf{V}}$ receives any tensor query (or queries) $q \in \mathcal{Q}_{\mathrm{tensor}}(\mathbb{F}, k_m, t)$ on $(\Pi_0, \Pi_1, \ldots, \Pi_i)$ from $\mathbf{V}$, it sends the query q to $\hat{\mathbf{P}}$, who checks whether the set of tensor queries $\hat{\mathbf{V}}$ sent so far could be made by the verifier $\mathbf{V}$. If not, $\hat{\mathbf{P}}$ terminates the interaction. If so, it responds by computing $q(\mathbb{x}, \Pi_0, \Pi_1, \ldots, \Pi_i)$ and sending it to $\hat{\mathbf{V}}$ as a non-oracle message. Then $\hat{\mathbf{V}}$ forwards this (alleged) query answer to $\mathbf{V}$.

This completes the simulation of the tensor IOP. If at this point the tensor IOP verifier $\mathbf{V}$ rejects, then the new verifier $\hat{\mathbf{V}}$ rejects too. (There is no need to check if the IOP prover $\hat{\mathbf{P}}$ answered tensor queries honestly.)

(2) Consistency phase. In this phase the IOP verifier $\hat{\mathbf{V}}$ checks that the IOP prover $\hat{\mathbf{P}}$ honestly answered the tensor queries of the tensor IOP verifier $\mathbf{V}$ in the simulation phase. Suppose that the tensor queries of $\mathbf{V}$ are $\{q^{(s)} = (q_0^{(s)}, \ldots, q_t^{(s)})\}_{s \in [\mathsf{q}]}$, and the prover $\hat{\mathbf{P}}$ has sent alleged answers $\{v_s\}_{s \in [\mathsf{q}]}$ to them; note that these are known to both parties. The IOP prover $\hat{\mathbf{P}}$ and IOP verifier $\hat{\mathbf{V}}$ interact as follows.

1. *Masking.* The prover $\hat{\mathbf{P}}$ samples uniformly random mask $\Xi \in \mathbb{F}^{\ell \cdot k_m^t}$, pads Ξ with uniformly-sampled encoding randomness to get $\tilde{\Xi} \in \mathbb{F}^{\ell \cdot k^t}$, computes an encoding $\hat{\Xi} \leftarrow \tilde{\mathrm{Enc}}_{\mathcal{C}^{\otimes t}}(\tilde{\Xi}) \in \mathbb{F}^{\ell \cdot n^t}$, and sends $\hat{\Xi}$ to $\hat{\mathbf{V}}$. The prover computes $w_s := q^{(s)}(\mathbb{x}, \Xi)$ for each $s \in [\mathsf{q}]$, and sends $(w_s)_{s \in [\mathsf{q}]} \in \mathbb{F}^{\mathsf{q}}$ to $\hat{\mathbf{V}}$ as a non-oracle message. The verifier $\hat{\mathbf{V}}$ responds with a challenge α sampled uniformly from $\mathbb{F}$.

   Define the functions

$$\tilde{\Pi} := \mathrm{Stack}(\tilde{\Pi}_0, \ldots, \tilde{\Pi}_{\mathsf{rc}}) \in \mathbb{F}^{\ell \cdot k^t} \quad ,$$

$$\hat{\Pi} := \mathrm{Stack}(\hat{\Pi}_0, \ldots, \hat{\Pi}_{\mathsf{rc}}) \in \mathbb{F}^{\ell \cdot n^t} \ ,$$
$$\tilde{f} := \alpha \cdot \tilde{\Pi} + \tilde{\Xi} \in \mathbb{F}^{\ell \cdot k^t} \ , \text{and}$$
$$c := \alpha \cdot \hat{\Pi} + \hat{\Xi} \in \mathbb{F}^{\ell \cdot n^t} \ .$$

Note that $c = \mathrm{Enc}_{\mathcal{C}^{\otimes t}}(\tilde{f})$. Moreover, $\hat{\mathbf{P}}$ already knows the value of $\tilde{f}$ and of c at every point, and $\hat{\mathbf{V}}$ can compute the value of any point in c by querying the corresponding points in $\hat{\Pi}$ and $\hat{\Xi}$.

2. *Consistency test.* Let $\{q^{(s)}\}_s$ be the set of tensor queries. Let $\{q_*^{(s)}\}_s$ be the set of queries obtained by taking each query $q^{(s)} \in \mathbb{F}^\ell \times (\mathbb{F}^{k_m})^t$ and appending zeroes to the end of each component to obtain an element of $\mathbb{F}^\ell \times (\mathbb{F}^k)^t$. Note that $\langle \otimes q_*^{(s)}, \tilde{f} \rangle = \langle \otimes q^{(s)}, f \rangle$ for each $s \in [\mathsf{q}]$.

   The prover $\hat{\mathbf{P}}$ and verifier $\hat{\mathbf{V}}$ engage in the IOP of proximity protocol $\mathsf{IOPP} = (\mathbf{P}', \mathbf{V}')$ for R_{cons} with index $\mathbb{i} = \bot$, instance $\mathbb{x} = (\mathbb{F}, \mathcal{C}, \ell, \mathsf{q}, t, \{q_*^{(s)}\}_s, \{\alpha \cdot v_s + w_s\}_s)$, and witness $\mathbb{w} = c$ to ensure that $c \in (\mathcal{C}^{\otimes t})^\ell$ (or at least close) and that for all $s \in [\mathsf{q}]$, $\langle \otimes q_*^{(s)}, \tilde{f} \rangle = \alpha \cdot v_s + w_s$. If $\mathbf{V}'$ rejects in this sub-protocol, then $\hat{\mathbf{V}}$ rejects.

8.2 Proof of Lemma 8.2

Proof. The round complexity, proof length, query complexity, indexer time, prover time, and verifer time follow directly from the construction. The completeness follows from the completeness of the original tensor IOP protocol $(\mathbf{I}, \mathbf{P}, \mathbf{V})$ and the completeness of the IOPP protocol $(\mathbf{P}', \mathbf{V}')$.

Soundness. Suppose $(\mathbb{i}, \mathbb{x}) \notin L(R)$, fix a malicious prover $\tilde{\mathbf{P}}$. Let $\hat{\Pi}$ be the oracle messages sent to $\hat{\mathbf{V}}$ during the simulation phase and $\hat{\Xi}$ be the mask sent at the start of the consistency phase. At least one of the following cases would hold:

- $\hat{\Pi}$ or $\hat{\Xi}$ is $1/2$-far from $(\mathcal{C}^{\otimes t})^\ell$ (here we consider blockwise distance). Let $\hat{\Pi}_\bot$ and $\hat{\Xi}_\bot$ denote the components of the oracles that are orthogonal to $(\mathcal{C}^{\otimes t})^\ell$. Let $T_e := \{j \in [n^t] \mid \exists i \in [\ell] \text{ s.t. } \hat{\Pi}_\bot[i,j] \neq 0 \vee \hat{\Xi}_\bot[i,j] \neq 0\}$. Define a collection of disjoint sets of coordinates $\{S_\beta\}_{\beta \in \mathbb{F}}$ where $S_\beta := \{j \in T_e \mid \forall i \in [\ell k],\ \beta \hat{\Pi}_\bot[i,j] + \hat{\Xi}_\bot[i,j] = 0\}$. Note that $\sum_{\beta \in \mathbb{F}} |S_\beta| \leq |T_e|$ and $|T_e| > n^t/2$. Then

$$\begin{aligned}
\Pr_{\alpha \sim \mathbb{F}}[\alpha \hat{\Pi} + \hat{\Xi} \text{ is } 1/4\text{-close to } (\mathcal{C}^{\otimes t})^\ell] &= \Pr_{\alpha \sim \mathbb{F}}[\alpha \hat{\Pi}_\bot + \hat{\Xi}_\bot \text{ is } 1/4\text{-close to } 0^{\ell \cdot n^t}] \\
&= \Pr_{\alpha \sim \mathbb{F}}[\delta_{|S_\alpha| \geq |T_e| - n^t/4}] \qquad (\delta_E \text{ is the indicator variable of event } E) \\
&\leq \frac{1}{|\mathbb{F}|} \cdot \frac{|T_e|}{|T_e| - n^t/4} \\
&< \frac{2}{|\mathbb{F}|} \ .
\end{aligned}$$

  Since the probability can only be multiples of $1/|\mathbb{F}|$, we obtain an upper bound of $1/|\mathbb{F}|$. In this case $\hat{\mathbf{V}}$ accepts with probability at most $1/|\mathbb{F}| + (1 - 1/|\mathbb{F}|)\epsilon'(1/4)$.

- $\hat{\Pi}$ and $\hat{\Xi}$ are $1/2$-close to $(\mathcal{C}^{\otimes t})^\ell$. In this case $\hat{\mathbf{V}}$ accepts with probability at most ϵ by the soundness of the original IOP protocol.

Zero knowledge. By construction, if the honest IOPP verifier $\mathbf{V}'$'s witness queries are compatible with some query checker $\mathbf{Q}$, then the resulting honest IOP verifier $\hat{\mathbf{V}}$'s queries to $\hat{\Pi}$ and $\hat{\Xi}$ are also compatible

with $\mathbf{Q}$. Thus, to show the two zero knowledge properties of the compiler, it suffices to prove that when $\left(\mathcal{C}^{\otimes t}\right)^{\ell}$ is $\mathbf{Q}$-query zero knowledge, the new protocol is zero knowledge for every verifier whose queries to $\hat{\Pi}$ and $\hat{\Xi}$ are compatible with $\mathbf{Q}$.

We define a simulator $\hat{\mathbf{S}}$ that when given access to such a verifier $\tilde{\mathbf{V}}$, the original IOP simulator $\mathbf{S}$, 2 identical simulators $\mathcal{S}_1$ and $\mathcal{S}_2$ for the code $\left(\mathcal{C}^{\otimes t}\right)^{\ell}$, and input $(\mathtt{i}, \mathtt{x})$ simulates the view of $\tilde{\mathbf{V}}$ in the real protocol.

1. Start simulating $\tilde{\mathbf{V}}$.
2. Answer any tensor query $q^{(s)} \in \mathcal{Q}_{\text{tensor}}(\mathbb{F}, k, t)$ on Π by first checking whether the set of tensor queries $\tilde{\mathbf{V}}$ sent so far could be made by the honest IOP verifier $\mathbf{V}$. If not, return "stop". If so, send $q^{(s)}$ to $\mathbf{S}$ and reply with $\mathbf{S}$'s answer v_s.
3. Forward any point query to $\hat{\Pi}$ to $\mathcal{S}_1$ and answer with the simulator's output. Denote the set of queries by $Q_{\hat{\Pi},0}$.
4. For each tensor query $q^{(s)} \in \mathcal{Q}_{\text{tensor}}(\mathbb{F}, k, t)$ to the encoded mask, sample a uniformly random element in $\mathbb{F}$ as the answer w_s and send it to $\tilde{\mathbf{V}}$. In the consistency phase, forward any query to $\hat{\Xi}$ to $\mathcal{S}_2$ and answer with the simulator's output. Let $Q_{\hat{\Xi}}$ be $\tilde{\mathbf{V}}$'s point queries to $\hat{\Xi}$ before the simulation of the next step. Let $A_{\hat{\Xi}}$ be $\hat{\mathbf{S}}$'s answers to these queries.
5. Receive the challenge α from $\tilde{\mathbf{V}}$.
6. For any $q \in Q_{\hat{\Xi}} \setminus Q_{\hat{\Pi},0}$, sample $\hat{\Pi}(q)$ uniformly at random in $\mathbb{F}$. Sample $X \in \mathbb{F}^{\ell \cdot k^t}$ uniformly at random conditioned on that

$$\forall q^{(s)} \in \mathcal{Q}_{\text{tensor}}(\mathbb{F}, k, t), \quad q^{(s)}(\mathtt{x}, X) = \alpha v_s + w_s, \tag{4}$$

$$\forall (j_0, \ldots, j_t) \in Q_{\hat{\Xi}}, \quad \hat{X}(j_0, \ldots, j_t) = \alpha \hat{\Pi}(j_0, \ldots, j_t) + A_{\hat{\Xi}}(j_0, \ldots, j_t) \ . \tag{5}$$

   Let $\tilde{f} := X$. In the rest of the consistency phase, run the IOPP prover $\mathbf{P}'$ on instance $\mathtt{x}_{\mathsf{IOPP}} = (\mathbb{F}, \mathcal{C}, \ell, \mathsf{q}, t, \{q_*^{(s)}\}_s, \{q_*^{(s)}(\mathtt{x}, X)\}_s)$ using witness $\hat{X}$ to generate messages and oracles for the IOPP protocol.
7. Answer any query q to $\hat{\Xi}$ (resp. $\hat{\Pi}$) by first checking if q has been made to $\hat{\Pi}$ (resp. $\hat{\Xi}$). If not, forward it to $\mathcal{S}_2$ (resp. $\mathcal{S}_1$) and answer with the simulator's output. If so, answer with $\hat{X}(q) - \alpha\hat{\Pi}(q)$ (resp. $\alpha^{-1}(\hat{X}(q) - \hat{\Xi}(q))$).

Note that $\hat{\mathbf{S}}$ runs in polynomial time. In particular, in Item 2 checking whether the set of tensor queries is in the support of the honest verifier's query sets requires solving a system of linear equations of constant dimensions. Additionally since an honest verifier makes only constant number of tensor queries, this step only takes constant time. In Item 6, we conditionally sample X. Let R denote the randomness used in encoding X. Equation (4) imposes linear constraint on X, while Equation (5) imposes linear constraint on X and R. These constraints can be efficiently generated from the queries $q^{(s)}$ and the generator matrix of the code

$\mathcal{C}$. Since $\mathcal{Q}_{\text{tensor}}(\mathbb{F}, k, t)$'s size is bounded by a constant and $Q_{\hat{\Xi}}$'s size is bounded by $|\mathcal{S}|$, the set of linear equations on X and R can be solved in $\text{poly}(|\mathcal{S}|, \ell k^t, \log|\mathbb{F}|)$ time. Other steps of $\hat{\mathbf{S}}$ are clearly efficient.

To see that $\hat{\mathbf{S}}$ simulates the view of $\tilde{\mathbf{V}}$ perfectly, we consider a hybrid experiment in which the "hybrid prover" reads all of the witness z (like the honest prover in the real world) but can modify messages after they are sent (like the simulator in the ideal world).

1. Start simulating $\tilde{\mathbf{V}}$.
2. Answer any tensor query $q^{(s)}$ to $\hat{\Pi}$ by first checking whether the set of tensor queries $\hat{\mathbf{V}}$ sent so far could be made by the honest IOP verifier $\mathbf{V}$. If not, return "stop". If so, send $q^{(s)}$ to $\mathbf{S}$ and reply with $\mathbf{S}$'s answer v_s.
3. Forward any point query to $\hat{\Pi}$ to $\mathcal{S}_1$ and answer with the simulator's output. Denote the set of queries by $Q_{\hat{\Pi},0}$ and the answers by $A_{\hat{\Pi},0}$.
4. Construct a uniformly random $\Pi \in \mathbb{F}^{\ell \cdot k_m^t}$ such that for each $s \in [\mathsf{q}]$, the answer to the tensor query $q^{(s)}$ on Π is v_s. Find the encoding of Π such that for all $q \in Q_{\hat{\Pi},0}$, $\hat{\Pi}(q) = A_{\hat{\Pi},0}(q)$.
5. Sample a random mask $\Xi \in \mathbb{F}^{\ell \cdot k_m^t}$, pad it with randomness to get $\tilde{\Xi} \in \mathbb{F}^{\ell \cdot k^t}$, and then encode the padded mask to obtain $\hat{\Xi} \in \mathbb{F}^{\ell \cdot n^t}$. Compute $w_s = q^{(s)}(\mathbb{x}, \Xi)$ and send the answers to $\tilde{\mathbf{V}}$.
6. Answer any query to $\hat{\Pi}$ by $\hat{\Pi}$, and any query to $\hat{\Xi}$ by $\hat{\Xi}$. Let $Q_{\hat{\Xi}}$ be $\tilde{\mathbf{V}}$'s queries to $\hat{\Xi}$ before the simulation of the next step. Let $A_{\hat{\Xi}}$ be $\hat{\mathbf{S}}$'s answers to these queries.
7. Receive the challenge α from $\tilde{\mathbf{V}}$.
8. Query $\hat{\Pi}$ at all locations in $Q_{\hat{\Xi}}$. Sample $X \in \mathbb{F}^{\ell \cdot k^t}$ uniformly at random conditioned on that

$$\begin{aligned} \forall q^{(s)} \in \mathcal{Q}_{\text{tensor}}(\mathbb{F}, k, t) \quad & q_*^{(s)}(\mathbb{x}, X) = \alpha v_s + w_s, \\ \forall (j_0, \ldots, j_t) \in Q_{\hat{\Xi}}, \quad & \hat{X}(j_0, \ldots, j_t) = \alpha \hat{\Pi}(j_0, \ldots, j_t) + A_{\hat{\Xi}}(j_0, \ldots, j_t) \ . \end{aligned}$$

9. Replace $\hat{\Xi}$ with $\hat{X} - \alpha\hat{\Pi}$.
10. Simulate the rest of the interaction with the oracles $\hat{\Xi}$, $\hat{\Pi}$, and $\hat{X}$.

By the zero-knowledge property of the original IOP protocol, in this experiment $(\{q^{(s)}, v_s\}_s, \Pi)$ are identically distributed as in the real protocol. Also since $\tilde{\mathbf{V}}$'s queries to $\hat{\Pi}$ and $\hat{\Xi}$ are compatible with $\mathbf{Q}$ and the interleaved code $\left(\mathcal{C}^{\otimes t}\right)^{\ell}$ is $\mathbf{Q}$-query zero knowledge, the distribution $(\{q_*^{(s)}, v_s\}_s, (Q_{\hat{\Pi},0}, A_{\hat{\Pi},0}))$ is identical to the distribution produced by the prover. So the joint distribution of $(\{q_*^{(s)}, v_s\}_s, (Q_{\hat{\Pi},0}, A_{\hat{\Pi},0}), \hat{\Pi})$ is also identical to that in the real protocol. Moreover all queries to $\hat{\Xi}$ after Item 9 have the correct distribution as in the actual protocol. Finally, the view of $\tilde{\mathbf{V}}$ in the above experiment follows the same distribution as the output of the simulator $\hat{\mathbf{S}}$. Therefore $(\hat{\mathbf{I}}, \hat{\mathbf{P}}, \hat{\mathbf{V}})$ is zero-knowledge for any $\tilde{\mathbf{V}}$whose queries to $\hat{\Pi}$ and $\hat{\Xi}$ are compatible with $\mathbf{Q}$.

□

9 Main theorem

Below we state our main theorem, Theorem 9.1, which is an IOP with point queries for rank-1 constraint satisfiability, and the formal analogue of Theorem 2. Theorem 9.1 differs from Theorem 2 in that we expose several parameters of interest, described below. See Remark 9.2 and Remark 9.3 for an explanation of how these parameters lead to Theorem 2.

In the following theorem, we denote by M the number of non-zero entries in the $N \times N$ matrices A, B and C in an instance of R_{R1CS}. Write $k_{\mathrm{SP}}(N)/|\mathbb{F}|$ for the soundness error of the twisted scalar-product protocol, $c_{\mathrm{SP}}(N)$ for its communication complexity and $r_{\mathrm{SP}}(N)$ for its round complexity. Write $k_{\mathrm{LC}}(N, M)/|\mathbb{F}|$, $c_{\mathrm{LC}}(N, M)$, and $r_{\mathrm{LC}}(N, M)$ for the corresponding parameters of the lincheck protocol.

Theorem 9.1 (main)**.** *Let $\mathbb{F}$ be a finite field. Let $t \in \mathbb{N}$ be a constant. Let $\mathcal{C} \colon \mathbb{F}^{N^{1/t}} \to \mathbb{F}^{O(N^{1/t})}$ be a code with constant rate and distance, which is linear-time encodable, with description size $|\mathcal{C}|$. There is a public-coin holographic IOP (with point queries) for R_{R1CS} instances with size $M = O(N)$ and fields $\mathbb{F}$ with $|\mathbb{F}| = \Omega(N)$ that has the following efficiency:*

- *answer alphabet $\mathbb{F}$;*
- *soundness error is $O((M + k_{\mathrm{SP}}(N) + k_{\mathrm{LC}}(N, M))/|\mathbb{F}|)$;*
- *round complexity is $O(r_{\mathrm{SP}}(N) + r_{\mathrm{LC}}(N, M))$;*
- *proof length is $O(M + N)$;*
- *query complexity is $O(c_{\mathrm{SP}}(N) + c_{\mathrm{LC}}(N, M))$;*
- *the indexer uses $O(M)$ field operations*
- *the prover uses $O(M + N) + \mathsf{poly}(|\mathcal{C}|, N^{1/t})$ field operations;*
- *the verifier uses $O(\ell_{\mathrm{in}} \cdot k^{t_{\mathrm{in}}}) + \mathsf{poly}\,(|\mathcal{C}|, \log N)$ field operations.*

Moreover, the protocol is semi-honest-verifier zero knowledge. Let $\beta \colon \mathbb{N} \to (0, 1)$ be a function bounded away from 1. If the code $\mathcal{C}$ is also $\beta N^{1/t}$-query zero knowledge, the protocol is zero knowledge for all verifiers making at most $\beta N^{1/t}$ queries.

Remark 9.2. If we just want a semi-honest-verifier zero-knowledge protocol, we can use explicit codes $\mathcal{C}$. In this case $|\mathcal{C}| = O(1)$. Thus the second component of verifier time is $\mathsf{poly}\,(|\mathcal{C}|, \log N) \in \mathsf{polylog}(M)$. So the total verifier time is polylogarithmic in M. We can pick t that exceeds the polynomial degree in the expression for prover time, so that the total prover time is linear in M.

If we want the stronger notion of bounded-query zero-knowledge, we can use the zero-knowledge codes given by Theorem 7.3. In this case $|\mathcal{C}| = O(N^{1/t})$. Then we can pick t that exceeds the polynomial degrees in the expressions for prover time and verifier time, so that $\mathsf{poly}(|\mathcal{C}|, N^{1/t})$ and $\mathsf{poly}\,(|\mathcal{C}|, \log N)$ are both in $o(M)$. So the prover time is linear in M and the verifier time is sublinear in M.

Remark 9.3. When $M = O(N)$, the best known efficiency parameters for the scalar-product and lincheck protocols satisfy $c_{\mathrm{SP}}(N), r_{\mathrm{SP}}(N), k_{\mathrm{SP}} = O(\log N)$, $c_{\mathrm{LC}}(N, M), r_{\mathrm{LC}}(N, M) = O(\log M)$, $k_{\mathrm{LC}}(N, M) = O(M)$. These arise from Theorem 4.8, and Lemma 4.11 instantiated with the bias-biased generator of Definition 4.6.

We obtain this IOP for R_{R1CS} in three steps. First, we construct a robust IOPP for the relation R_{cons}. Then we use proof composition to improve the round complexity and query complexity of the IOPP. Finally, we apply the compiler in Lemma 8.2 to this efficient IOPP for R_{cons} and to the tensor-query IOP for R_{R1CS} in Theorem 4.13 to obtain the protocol in the theorem statement. In the rest of the section, we explain each of the steps.

9.1 Step 1: robustification

We start with the following IOPP for the relation R_{cons}.

Definition 9.4. *The indexed relation R'_{cons} is the set of tuples*

$$(\mathbb{i}, \mathbb{x}, \mathbb{w}) = \Big(\bot, (\mathbb{F}, \mathcal{C}, \ell, \mathsf{q}, t, \{q^{(s)}\}_s, \{v_s\}_s), c\Big)$$

such that $c = \mathrm{Enc}_{\mathcal{C}^{\otimes(t-1)}}(f) \in \mathbb{F}^{\ell k \cdot n^{t-1}}$ for some $f \in \mathbb{F}^{\ell \cdot k^t}$, for each $s \in [\mathsf{q}]$, $q^{(s)} = (q_0^{(s)}, \ldots, q_t^{(s)}) \in \mathbb{F}^\ell \times \left(\mathbb{F}^k\right)^t$, and for all $s \in [\mathsf{q}]$, $\langle q^{(s)}, f\rangle = v_s$.

Lemma 9.5. *Let $b \in \Theta(k)$. Let $\mathcal{C} \colon \mathbb{F}^k \to \mathbb{F}^n$ be a linear code with constant rate $\rho = k/n$, constant relative distance $\delta = d/n$, linear encoding arithmetic complexity $\theta(k) \cdot k$, and description size $|\mathcal{C}|$. There exists an non-adaptive interactive oracle proof of proximity $\mathsf{IOPP} = (\mathbf{P}, (\mathbf{V}_{\mathrm{q}}, \mathbf{V}_{\mathrm{d}}))$ with point queries for the relation $R'_{\mathsf{cons}}(\mathbb{F}, \mathcal{C}, \ell, \mathsf{q}, t, \{q^{(s)}\}_s, \{v_s\}_s)$ with the following parameters:*

- *answer alphabet $\mathbb{F}^{\ell k}$ for the witness, and $\mathbb{F}^k$ for the proof;*
- *soundness error $\epsilon(\Delta) = \ell k n^{t-1}/|\mathbb{F}| + 1 - \min\{\delta^{t-1}/4, \Delta/2^{t-1}\}$;*
- *round complexity $O(t)$;*
- *proof length $O(\mathsf{q} \cdot n^{t-2})$ symbols in $\mathbb{F}^k$;*
- *query complexity 1 to the witness and $O(t \cdot \mathsf{q})$ to the proof;*
- *prover time $O(|\mathcal{C}| + \mathsf{q} \cdot t\ell k^t)$;*
- *verifier time $O(|\mathcal{C}| + \mathsf{q}k \cdot (\ell + t))$;*
- *verifier randomness complexity $O(t \cdot \mathsf{q})$ elements in $\mathbb{F}$;*
- *$\mathbf{V}_{\mathrm{d}}$'s decision state $\sigma = (\mathbb{x}, r)$ where r is the verifier randomness;*
- *$\mathbf{V}_{\mathrm{d}}$ time $O(|\mathcal{C}| + \mathsf{q}k \cdot (\ell + t))$.*

Moreover, the verifier's witness queries are compatible with a query checker $\mathbf{Q}$ which allows revealing at most 1 column (in $\mathbb{F}^{\ell k}$) from the interleaved codeword in $\left(\mathcal{C}^{\otimes(t-1)}\right)^{\ell k}$.

Proof sketch. This IOPP is obtained by modifying the 'consistency phase' of [BCG20, Construction 9.4] which uses a proximity test [BCG20, Lemma 10.1] as a subroutine.

To obtain an IOPP for R_{cons}, we run the consistency phase of [BCG20, Construction 9.4], but parse the message $f_0^{(0)}$ as a t-dimensional array of size $\ell k \cdot k^{t-1}$ rather than a $(t+1)$-dimensional array of size $\ell \cdot k^t$. The first components $q_0^{(s)} \in \mathbb{F}^\ell$ and $q_1^{(s)} \in \mathbb{F}^k$ of each tensor query are then grouped together as a single component $q_0^{(s)} \otimes q_1^{(s)} \in \mathbb{F}^{\ell k}$, and the first 'folding' step applies $q_0^{(s)} \otimes q_1^{(s)}$ of $f_0^{(0)}$.

We also derandomise the proximity test [BCG20, Lemma 10.1] to reduce the verifier randomness complexity from $O((\ell + t) \cdot k \cdot \mathsf{q})$ field elements to $O(t \cdot \mathsf{q})$ field elements. Although this change increases the soundness error, it will help us to improve the final IOP verifier's running time after applying proof composition.

The proximity test checks whether, for a set of messages $\{\tilde{c}_0, \ldots, \tilde{c}_t\}$, we have that $\tilde{c}_i$ is close to an interleaved $\mathcal{C}^{\otimes(t-i)}$-codeword. This is achieved by recursively 'folding' the higher dimensional messages to reduce their dimension, by taking a random linear combination of their coefficients at each stage. To perform the random folding, the verifier samples random vectors $v_1, \ldots, v_t$ each in $\mathbb{F}^n$ and $\mathbb{F}^{\ell n}$. Executing the protocol with the array sizes described above would require the verifier to sample uniformly random vectors $v_1 \in \mathbb{F}^{\ell k}$ and $v_2, \ldots, v_t \in \mathbb{F}^k$. However, in this case, the verifier's randomness complexity would be $O(k)$, which prevents polylogarithmic verification after proof composition. However, we can replace each

v_i with a random vector of the form $(\alpha_i^1, \alpha_i^2, \alpha_i^3, \ldots, \alpha_i^{|v_i|})$ for a uniformly random $\alpha_i \leftarrow \mathbb{F}$, and replace [BCG20, Lemma 10.10] with [BSCIKS20, Theorem 1.2] in the soundness proof of the proximity test. This modification increases the soundness error, but we can tolerate the increase for sufficiently large fields. □

Remark 9.6. We also note that the analysis in [BSCIKS20] could potentially be adapted to show that if random Reed-Muller vectors are used to fold the messages, the first term in the soundess error could be improved from $\ell k n^{t-1}/|\mathbb{F}|$ to $\mathsf{polylog}(k,n)/|\mathbb{F}|$, with verifier randomness complexity changing to $O(t \cdot \mathsf{q} \log \ell k)$.

In the construction above, we use a b-query zero-knowledge code $\mathcal{C}$ with for $b \leq k$.

Next, we reduce the verifier query complexity through proof composition. We first robustify the current IOPP for R'_{cons} to obtain a robust IOPP for R_{cons}.

Corollary 9.7. *Let $\mathcal{C} \colon \mathbb{F}^k \to \mathbb{F}^{O(k)}$ be a randomised linear code with constant rate and relative distance, linear-time encoding, membership-deciding, and decoding time, and description size $|\mathcal{C}|$. The relation R_{cons} has a non-adaptive point-query IOPP $(\mathbf{P}_{\mathrm{r}}, (\mathbf{V}_{\mathrm{q_r}}, \mathbf{V}_{\mathrm{d_r}}))$ with:*

- *answer alphabet $\mathbb{F}$;*
- *round complexity $O(t)$;*
- *proof length $O(\mathsf{q} \cdot \ell k^t)$;*
- *query complexity $O(\mathsf{q}k \cdot (\ell + t))$;*
- *soundness error $O(k^t/|\mathbb{F}|) + O(1)$ with robustness $\Theta(1/(\ell + t))$;*
- *prover time $O(|\mathcal{C}| + \mathsf{q} \cdot t\ell k^t)$;*
- *verifier time $O(|\mathcal{C}| + \mathsf{q}k \cdot (\ell + t))$;*
- *verifier randomness complexity $O(t\mathsf{q})$;*
- *$\mathbf{V}_{\mathrm{d_r}}$'s decision state $\sigma = (\mathbb{x}, r)$ where r is the verifier randomness;*
- *$\mathbf{V}_{\mathrm{d_r}}$ time $O(|\mathcal{C}| + \mathsf{q}k \cdot (\ell + t))$.*

Moreover, the verifier's witness queries are compatible with a query checker $\mathbf{Q}_{\mathrm{r}}$ which allows revealing at most 1 column (in $\mathbb{F}^{O(\ell k)}$) from the interleaved codeword in $\left(\mathcal{C}^{\otimes(t-1)}\right)^{O(\ell k)}$.

Proof. Apply Lemma A.1 to the point-query IOPP for R'_{cons} in Lemma 9.5. In Lemma 9.5, the witness has alphabet $\mathbb{F}^{\ell k}$, and will be encoded in ℓ parts by applying $\mathcal{C}$ to blocks of length k. This converts a witness for the relation R'_{cons} into a witness for the relation R_{cons}. Thus the new witness can be viewed as ℓ codewords in $\mathcal{C}^{\otimes t}$, or equivalently $O(\ell k)$ codewords in $\mathcal{C}^{\otimes(t-1)}$. Proof elements have alphabet $\mathbb{F}^k$ and will be encoded using $\mathcal{C}$. □

9.2 Step 2: composition

Lemma 9.8 ([Mie09])**.** *Fix any $T(n) = \Omega(n)$, $\delta = \delta(n) > 0$, and $\epsilon = \epsilon(n) > 0$. Every relation R in $\mathsf{NTIME}(T)$ has a PCPP for R with (1) answer alphabet $\{0,1\}$; (2) proof length $T(|\mathbb{x}|) \cdot \mathsf{polylog}(T(|\mathbb{x}|))$; (3) query complexity $O(\log(1/\epsilon) \cdot \log(1/\delta)/\delta)$; (4) soundness error ϵ with proximity parameter δ; (5) prover time $\mathsf{poly}(T(|\mathbb{x}|))$; (6) verifier time $\mathsf{poly}(|\mathbb{x}|, \log T(|\mathbb{x}|))$.*

Corollary 9.9. *Let $\epsilon \in (0,1)$ be a constant. For instances of R_{cons} in which each query $q^{(s)}$ is described using a short seed of length $O(t \log k)$, there is a point-query IOPP $(\mathbf{P}, \mathbf{V})$ with:*

- *answer alphabet $\mathbb{F}$;*

- *round complexity* $O(t)$*;*
- *proof length* $O(\mathsf{q} \cdot \ell k^t)$*;*
- *query complexity* $O_\epsilon(t \log t)$*;*
- *soundness error* $O(k^t/|\mathbb{F}|) + O(1) + \epsilon$ *with proximity parameter* $\Theta(1/t)$*;*
- *prover time* $\mathsf{poly}(|\mathcal{C}| + \mathsf{q}k \cdot (\ell + t))$*;*
- *verifier time* $\mathsf{poly}(|\mathcal{C}| + O(\mathsf{q} \cdot t \log k), \log(|\mathcal{C}| + \mathsf{q}k \cdot (\ell + t)))$.

Moreover, the verifier's witness queries are still compatible with a query checker $\mathbf{Q}_{\mathrm{r}}$ *which allows revealing at most* 1 *column (in* $\mathbb{F}^{O(\ell k)}$*) from the interleaved codeword in* $\left(\mathcal{C}^{\otimes(t-1)}\right)^{O(\ell k)}$.

Proof. Apply Lemma B.1 to the robust outer IOP provided by Corollary 9.7 and the inner IOP of proximity provided by Lemma 9.8.

The inner relation $R(\mathbf{V}_{\mathrm{dr}})$ has instance $(\mathbb{x}, r) := (\mathbb{F}, \mathcal{C}, \ell, \mathsf{q}, t, \{q^{(s)}\}_s, \{v_s\}_s, r)$, where $\mathcal{C}$ has description size $|\mathcal{C}|$, r has size $O(\mathsf{q} \cdot t)$, and by assumption, the query set has a description of size $O(\mathsf{q} \cdot t \log k)$. This means that the instance size is $|\mathcal{C}| + O(\mathsf{q} \cdot t \log k)$. The witness size of $R(\mathbf{V}_{\mathrm{dr}})$ is equal to the query complexity of the robust outer IOP. Thus it has size $O(\mathsf{q}k \cdot (\ell + t))$. The verifier's decision time is $T(|\mathbb{x}|) = O(|\mathcal{C}| + \mathsf{q}k \cdot (\ell + t))$. Substituting these values into Lemma 9.8, the final IOPP's proof length, prover time and verifier time are as written in the statement.

The round complexity $O(t)$ is inherited from the robust outer IOPP, while the query complexity $O_\epsilon(t \log t)$ is obtained from the applying Lemma 9.8 with the proximity parameter $\delta = \Theta(1/t)$. □

9.3 Step 3: tensor queries to point queries

Finally, we apply the transformation given in Lemma 8.2 to the tensor-query R_{R1CS} IOP of Theorem 4.13, the R_{cons} IOPP in Corollary 9.9, and two different zero knowledge codes to produce two point-query IOPs for R_{R1CS} and complete the proof of Theorem 9.1. Set $N = k_m^t$ and $M = \ell \cdot k_m^t$ for some $k_m, t \in \mathbb{N}$. Theorem 4.13 gives a $(\mathbb{F}, k_m, t)$-tensor IOP for R_{R1CS}, which has query complexity $\mathsf{q} = O(1)$, and for which each query can be described by a short seed of size $O(t \log k_m)$.

By Definition 3.22, there exists a randomized linear code $\mathcal{C} : \mathbb{F}^{k_m} \times \mathbb{F}^{O(k_m)} \to \mathbb{F}^{O(k_m)}$ which has constant message rate, randomness rate, and relative distance, and is linear-time encodable and 1-query zero-knowledge. Therefore the tensor code $\mathcal{C}^{\otimes(t-1)}$ is also 1-query zero-knowledge. If we encode the interleaved code $\left(\mathcal{C}^{\otimes(t-1)}\right)^{\ell k}$ as in the proof of Corollary 9.7, the resulting interleaved code is $\mathbf{Q}_{\mathrm{r}}$-query zero knowledge. Moreover, by Corollary 9.9, the R_{cons} IOPP verifier $\mathbf{V}'$ for code $\mathcal{C}$ is compatible with $\mathbf{Q}_{\mathrm{r}}$. As a result, applying Lemma 8.2 to the tensor-query R_{R1CS} IOP, the R_{cons} IOPP, and $\mathcal{C}$ gives a point-query R_{R1CS} IOP with the stated parameters. Moreover, the IOP is zero knowledge against semi-honest verifier zero-knowledge.

By Theorem 7.3, there also exists a randomized linear code $\mathcal{C}' : \mathbb{F}^{k_m} \times \mathbb{F}^{O(k_m)} \to \mathbb{F}^{O(k_m)}$ which has constant message rate, randomness rate, and relative distance, is linear-time encodable, and is βk_m-query zero-knowledge. Thus the interleaved code $\left(\mathcal{C}'^{\otimes(t-1)}\right)^{O(\ell k_m)}$ is βk_m-query zero knowledge. Apply Lemma 8.2 to the tensor-query R_{R1CS} IOP, the R_{cons} IOPP, and the code $\mathcal{C}'$. The resulting point-query IOP has the stated parameters, and it is zero knowledge against all verifiers making at most βk_m queries to the IOP oracles.

A Robustification

We state a generic lemma about robustification for IOPs (Lemma A.1); it extends corresponding generic robustifications for PCPs and IPs. We include the construction for convenience (Construction A.3) and sketch its proof. We also prove properties about how zero knowledge is affected by proof composition (Lemma A.4).

Lemma A.1. *Suppose that we are given the following two ingredients.*

- ***IOP.*** *A holographic IOP* $(\mathbf{I}, \mathbf{P}, \mathbf{V})$ *for* R *with queries in* $\mathcal{Q}_{\text{point}}$ *with the following parameters: round complexity* rc*; answer alphabet* Σ*; oracle length* $\mathsf{l} = \mathsf{li} + \mathsf{lp}$*; query complexity* q*; soundness error* ϵ*; indexer time* ti*; prover time* tp*; verifier time* tv*; verifier randomness* r*.*
- ***Code.*** *A randomized code* $\mathcal{C} \colon \Sigma \to \Lambda^n$ *with public parameters* $(\mathrm{Enc}, G, H, G^+)$*, relative distance* δ *(over alphabet* Λ*), encoding time* θ*, membership-deciding time* ψ*, and error-free decoding time* η*. (In particular, the rate of* $\mathcal{C}$ *is* $\rho := \frac{\log|\Sigma|}{n \cdot \log|\Lambda|}$*.)*

Then the protocol in Construction A.3 (see below) is a holographic IOP $(\mathbf{I}_{\mathrm{r}}, \mathbf{P}_{\mathrm{r}}, \mathbf{V}_{\mathrm{r}})$ *with queries in* $\mathcal{Q}_{\text{point}}$ *for the indexed relation* R *with the following parameters:*

- *round complexity* rc*;*
- *answer alphabet* Λ*;*
- *index oracle length* $n \cdot \mathsf{li}$ *and proof oracle length* $n \cdot \mathsf{lp}$*;*
- *query complexity* $n \cdot \mathsf{q}$*;*
- *soundness error* ϵ *with robustness* $\frac{\delta}{2\mathsf{q}}$*;*
- *indexer time* $\mathsf{ti} + \mathsf{li} \cdot \theta$*;*
- *prover time* $\mathsf{tp} + \mathsf{lp} \cdot \theta$*;*
- *verifier time* $\mathsf{tv} + \mathsf{q} \cdot (\psi + \eta)$*;*
- *verifier randomness* r*.*

If $(\mathbf{I}, \mathbf{P}, \mathbf{V})$ *is public coin (resp., has non-adaptive queries) then* $(\mathbf{I}_{\mathrm{r}}, \mathbf{P}_{\mathrm{r}}, \mathbf{V}_{\mathrm{r}})$ *is public coin (resp., has non-adaptive queries).*

Remark A.2 (multiple alphabets)**.** Lemma A.1 and Lemma A.4 extend to holographic IOPs with multiple alphabets $\Lambda_1, \ldots, \Lambda_s$ provided that $|\Lambda_i| = \Theta(|\Lambda_1)$ for each $i \in [2, \ldots, s]$, and error-correcting codes $\mathcal{C}_1, \ldots, \mathcal{C}_s$ with respect to these alphabets. The parameters of $(\mathbf{I}_{\mathrm{r}}, \mathbf{P}_{\mathrm{r}}, \mathbf{V}_{\mathrm{r}})$ are then modified to use the minimum of the relative distances and the maximum of the block lengths, relative distances, encoding, checking, and error-free decoding times.

Construction A.3. The new indexer $\mathbf{I}_{\mathrm{r}}$ runs the old indexer $\mathbf{I}$ on the given index to obtain an index oracle $\Pi_0 = (a_{0,1}, \ldots, a_{0,\mathsf{li}}) \in \Sigma^{\mathsf{li}}$, encodes each symbol $a_{0,j} \in \Sigma$ using the given code to obtain a string of symbols $\mathcal{C}(a_{0,j}) \in \Lambda^n$, and then outputs the new index oracle $\mathcal{C}(\Pi_0) := (\mathcal{C}(a_{0,1}), \ldots, \mathcal{C}(a_{0,\mathsf{li}})) \in \Lambda^{n \cdot \mathsf{li}}$.

The new prover $\mathbf{P}_{\mathrm{r}}$ and new verifier $\mathbf{V}_{\mathrm{r}}$ respectively simulate the old prover $\mathbf{P}$ and old verifier $\mathbf{V}$, with modifications depending on the code $\mathcal{C}$ as described below.

- *The new prover encodes each symbol in each proof.* For each round $i \in [\mathsf{rc}]$, when the old prover $\mathbf{P}$ outputs a proof oracle $\Pi_i = (a_{i,1}, \ldots, a_{i,\mathsf{lp}_i}) \in \Sigma^{\mathsf{lp}_i}$, $\mathbf{P}_{\mathrm{r}}$ encodes each symbol $a_{i,j} \in \Sigma$ using the given code to obtain a string of symbols $\mathcal{C}(a_{i,j}) \in \Lambda^n$ and sends the encoded proof $\mathcal{C}(\Pi_i) := (\mathcal{C}(a_{i,1}), \ldots, \mathcal{C}(a_{i,\mathsf{lp}_i})) \in \Lambda^{n \cdot \mathsf{lp}_i}$.
- *The new verifier checks membership in the code.* Whenever the old verifier $\mathbf{V}$ wishes to read the j-th symbol of the i-th oracle Π_i, the new verifier $\mathbf{V}_{\mathrm{r}}$ does the following:

– Use n queries queries to read $\tilde{c}_{i,j} \in \Lambda^n$, the j-th string in the i-th oracle.
– If $i = 0$ ($\mathbf{V}$'s query was to the index oracle Π_0) then decode $\tilde{c}_{i,j}$ to the symbol $a_{i,j} \in \Sigma$. (Note that we already know that $\tilde{c}_{i,j}$ is in the code $\mathcal{C}$ because it is an output of the new indexer $\mathbf{I}_{\mathrm{r}}$.)
– If $i \in [\mathsf{rc}]$ ($\mathbf{V}$'s query was to one of the proof oracles $\Pi_1, \ldots, \Pi_{\mathsf{rc}}$) check that $\tilde{c}_{i,j}$ is in the code $\mathcal{C}$ (e.g. using the parity-check matrix H of $\mathcal{C}$ or a more efficient algorithm if known) and, if so, decode $\tilde{c}_{i,j}$ to the symbol $a_{i,j} \in \Sigma$ (e.g. using the matrix G^+ or otherwise).
– Give $a_{i,j}$ as the query answer to the old verifier $\mathbf{V}$.

In sum, the new verifier $\mathbf{V}_{\mathrm{r}}$ accepts if and only if all read strings are in the code $\mathcal{C}$ and the old verifier $\mathbf{V}$ accepts the corresponding decodings.

Proof of Lemma A.1. Completeness and the efficiency parameters of $(\mathbf{I}_{\mathrm{r}}, \mathbf{P}_{\mathrm{r}}, \mathbf{V}_{\mathrm{r}})$ claimed in the lemma (round complexity, answer alphabet, proof length, query complexity, prover time, and verifier time) directly follow from the construction. Below we prove robustness and preservation of zero knowledge.

We show that $(\mathbf{I}_{\mathrm{r}}, \mathbf{P}_{\mathrm{r}}, \mathbf{V}_{\mathrm{r}})$ is $(\frac{\delta}{2\mathsf{q}}, \epsilon)$-robust. Namely, we show that, for any cheating prover $\tilde{\mathbf{P}}_{\mathrm{r}}$, with probability at most ϵ, the view of $\mathbf{V}_{\mathrm{r}}$ is $\frac{\delta}{2\mathsf{q}}$-close to some accepting view.

Fix an arbitrary malicious prover $\tilde{\mathbf{P}}_{\mathrm{r}}$ and, for each round $i \in [\mathsf{rc}]$, denote by $\tilde{c}_{i,j} \in \Lambda^n$ the j-th string in the i-th proof string sent by $\tilde{\mathbf{P}}_{\mathrm{r}}$; note that each $\tilde{c}_{i,j}$ is a random variable that depends on the transcript so far. Denote by $\mathbf{v}_\rho$ the view of the new verifier $\mathbf{V}_{\mathrm{r}}$, when using randomness ρ, and interacting with the malicious prover $\tilde{\mathbf{P}}_{\mathrm{r}}$. Note that $\mathbf{v}_\rho$ consists of q strings in Λ^n. Let $\overline{c}_{i,j}$ be any codeword in $\mathcal{C}$ that is closest to $\tilde{c}_{i,j}$ (which need not be in $\mathcal{C}$), and let $\overline{\mathbf{v}}_\rho$ be the view of $\mathbf{V}_{\mathrm{r}}$ when each $\tilde{c}_{i,j}$ is replaced with its "correction" $\overline{c}_{i,j}$.

Let E be the event that the "corrected" view $\overline{\mathbf{v}}_\rho$ makes the new verifier $\mathbf{V}_{\mathrm{r}}$ accept. The event E occurs with probability at most ϵ because, if not, then the malicious prover $\tilde{\mathbf{P}}$ whose proofs are decodings of proofs output by $\tilde{\mathbf{P}}_{\mathrm{r}}$ would make the old verifier $\mathbf{V}$ accept with probability greater than ϵ, a contradiction to the soundness of $\mathbf{V}$. We now argue that if E does *not* occur ($\overline{\mathbf{v}}_\rho$ is rejecting) then it holds that the view of $\mathbf{V}_{\mathrm{r}}$ is $\frac{\delta}{2\mathsf{q}}$-far from accepting, which will conclude the proof.

We distinguish between two cases.

- *There exists a string $\tilde{c}_{i,j}$ in the view $\mathbf{v}_\rho$ that is $\delta/2$-far from $\mathcal{C}$.* Then we directly conclude that $\mathbf{v}_\rho$ is $\frac{\delta}{2\mathsf{q}}$-far from being accepted (regardless of E in fact), because all accepting views consist of strings in the code. Note that the loss in distance in the denominator is because the string $\tilde{c}_{i,j}$ contains n symbols out of the $n \cdot \mathsf{q}$ symbols in the view $\mathbf{v}_\rho$, and in the worst case all other strings are in the code and contribute no distance.

- *Every string $\tilde{c}_{i,j}$ in the view $\mathbf{v}_\rho$ is $\delta/2$-close to $\mathcal{C}$.* Fix any view $\mathbf{v}'$ that is $\frac{\delta}{2\mathsf{q}}$-close to $\mathbf{v}_\rho$. We need to argue that $\mathbf{v}'$ is a rejecting view. If $\mathbf{v}'$ contains any string that is not in $\mathcal{C}$ then, by construction of the new verifier $\mathbf{V}_{\mathrm{r}}$, $\mathbf{v}'$ is rejecting. So suppose that every string in $\mathbf{v}'$ is in $\mathcal{C}$. In this case we argue that $\mathbf{v}'$ equals the corrected view $\overline{\mathbf{v}}_\rho$, which is rejecting because we assumed that E does not hold.

  Consider any string c' in $\mathbf{v}'$. Let c and $\overline{c}$ be the corresponding strings in $\mathbf{v}_\rho$ and $\overline{\mathbf{v}}_\rho$ respectively. Via the triangle inequality:

$$0 \leq \Delta_\Lambda(c', \overline{c}) \leq \Delta_\Lambda(c', c) + \Delta_\Lambda(c, \overline{c}) < \mathsf{q} \cdot \frac{\delta}{2\mathsf{q}} + \frac{\delta}{2} = \delta \ .$$

  Since $\mathcal{C}$ has relative distance δ, we conclude that $c' = \overline{c}$. And since the chosen string was arbitrary, we deduce that, as claimed, $\mathbf{v}'$ equals the corrected view $\overline{\mathbf{v}}_\rho$.

□

Lemma A.4. *The implications below hold for the robustification in Lemma A.1.*

1. **Semi-honest-verifier zero knowledge.** *If* $(\mathbf{I}, \mathbf{P}, \mathbf{V})$ *is zero knowledge again semi-honest verifiers, then so is* $(\mathbf{I}_{\mathrm{r}}, \mathbf{P}_{\mathrm{r}}, \mathbf{V}_{\mathrm{r}})$.

2. **Bounded-query zero knowledge.** *If* $\mathcal{C}$ *is* b'*-query zero knowledge and* $(\mathbf{I}, \mathbf{P}, \mathbf{V})$ *is* b*-query zero knowledge, then* $(\mathbf{I}_{\mathrm{r}}, \mathbf{P}_{\mathrm{r}}, \mathbf{V}_{\mathrm{r}})$ *is* $(b'b + b' + b)$*-query zero knowledge.*

Proof of Item 2 in Lemma A.4. Let $\mathbf{S}$ be the simulator for the old IOP $(\mathbf{I}, \mathbf{P}, \mathbf{V})$ and let $\mathcal{S}$ be the simulator for the code $\mathcal{C}$. Below we construct the new simulator $\mathbf{S}_{\mathrm{r}}$ for the new IOP $(\mathbf{I}_{\mathrm{r}}, \mathbf{P}_{\mathrm{r}}, \mathbf{V}_{\mathrm{r}})$.

Fix an index $\mathbb{i}$, instance $\mathbb{x}$, and witness $\mathbb{w}$ such that $(\mathbb{i}, \mathbb{x}, \mathbb{w}) \in R$ and fix a $(b'b + b' + b)$-query malicious verifier $\tilde{\mathbf{V}}_{\mathrm{r}}$. We need the random variables $\mathbf{S}_{\mathrm{r}}^{\tilde{\mathbf{V}}_{\mathrm{r}}}(\mathbb{i}, \mathbb{x})$ and $\mathrm{View}(\mathbf{P}_{\mathrm{r}}(\mathbb{i}, \mathbb{x}, \mathbb{w}), \tilde{\mathbf{V}}_{\mathrm{r}})$ to be identically distributed.

The new simulator $\mathbf{S}_{\mathrm{r}}$ receives as input an index $\mathbb{i}$ and instance $\mathbb{x}$, and computes the encoded index $\mathbf{I}_{\mathrm{r}}(\mathbb{i})$ for later use. Then $\mathbf{S}$ starts running $\tilde{\mathbf{V}}_{\mathrm{r}}$ and simulates its view by taking different actions depending on $\tilde{\mathbf{V}}_{\mathrm{r}}$. Whenever $\tilde{\mathbf{V}}_{\mathrm{r}}$ outputs a message for the new prover $\mathbf{P}_{\mathrm{r}}$, the new simulator $\mathbf{S}_{\mathrm{r}}$ forwards this message to the old simulator $\mathbf{S}$ (as if it were a message for the old prover $\mathbf{P}$). Whenever $\tilde{\mathbf{V}}_{\mathrm{r}}$ makes a query to the index oracle $\mathbf{I}_{\mathrm{r}}(\mathbb{i})$, the new simulator $\mathbf{S}_{\mathrm{r}}$ directly answers the query (as it knows the index and has computed the encoded index so that it knows the answer to all such queries). Whenever $\tilde{\mathbf{V}}_{\mathrm{r}}$ makes a query to a proof oracle, $\mathbf{S}_{\mathrm{r}}$ proceeds as follows. Suppose that $\tilde{\mathbf{V}}_{\mathrm{r}}$'s query is to the j-th string in the i-th proof oracle (for $i \in [\mathsf{rc}]$).

- $\mathbf{S}_{\mathrm{r}}$ answers the first b' distinct queries to this string by using an (independent) execution of the code simulator $\mathcal{S}$ for this string. Any subsequent queries (beyond the first b') exceed the ZK bound of the code $\mathcal{C}$, which means that we cannot rely on $\mathcal{S}$ to simulate answers to them. This brings us to the next option.
- $\mathbf{S}_{\mathrm{r}}$ answers subsequent queries by relying on the old simulator $\mathbf{S}$. Specifically, $\mathbf{S}_{\mathrm{r}}$ asks $\mathbf{S}$ to simulate the j-th symbol in the i-th proof oracle, which leads to an answer $a_{i,j} \in \Sigma$; then $\mathbf{S}_{\mathrm{r}}$ samples a codeword $\mathcal{C}(a_{i,j}) \in \Lambda^n$ that agrees with the previous b' answers, and answers all future queries to this string via this sampled codeword.

Observe that $\mathbf{S}_{\mathrm{r}}$ asks $\mathbf{S}$ to simulate at most $\lfloor \frac{b'b+b'+b}{b'+1} \rfloor = b$ queries, which $\mathbf{S}$ can do by assumption.

Also observe that the codeword in the second step can be found efficiently. Let R denote the randomness used in encoding $a_{i,j}$. The answers to the previous b' queries each imposes a linear constraint on R. These constraints can be efficiently generated from the previous queries and the generator matrix of the code $\mathcal{C}$. The set of linear equations on R can be solved in $\mathrm{poly}(b', \log|\Sigma|, \log|\Lambda|)$ time. □

Proof of Item 1 in Lemma A.4. Let $\mathbf{S}$ be the simulator for the old IOP $(\mathbf{I}, \mathbf{P}, \mathbf{V})$. Below we construct the new simulator $\mathbf{S}_{\mathrm{r}}$ for the new IOP $(\mathbf{I}_{\mathrm{r}}, \mathbf{P}_{\mathrm{r}}, \mathbf{V}_{\mathrm{r}})$.

Fix an index $\mathbb{i}$, instance $\mathbb{x}$, and witness $\mathbb{w}$ such that $(\mathbb{i}, \mathbb{x}, \mathbb{w}) \in R$ and fix the randomness ρ for the old verifier $\mathbf{V}$. Note that $\mathbf{V}_{\mathrm{r}}$ does not use any additional randomness beyond the randomness used by $\mathbf{V}$. We need the random variables $\mathbf{S}_{\mathrm{r}}^{\mathbf{V}_{\mathrm{r}}(\mathbb{i},\mathbb{x};\rho)}(\mathbb{i}, \mathbb{x})$ and $\mathrm{View}(\mathbf{P}_{\mathrm{r}}(\mathbb{i}, \mathbb{x}, \mathbb{w}), \mathbf{V}_{\mathrm{r}}(\mathbb{i}, \mathbb{x}; \rho))$ to be identically distributed.

The new simulator $\mathbf{S}_{\mathrm{r}}$ receives as input an index $\mathbb{i}$ and instance $\mathbb{x}$, and computes the encoded index $\mathbf{I}_{\mathrm{r}}(\mathbb{i})$ for later use. Then $\mathbf{S}_{\mathrm{r}}$ starts running $\mathbf{V}_{\mathrm{r}}$ and simulates its view by taking different actions depending on $\mathbf{V}_{\mathrm{r}}$. Whenever $\mathbf{V}_{\mathrm{r}}$ outputs a message for the new prover $\mathbf{P}_{\mathrm{r}}$, the new simulator $\mathbf{S}_{\mathrm{r}}$ forwards this message to the old simulator $\mathbf{S}$ (as if it were a message for the old prover $\mathbf{P}$). Whenever $\mathbf{V}_{\mathrm{r}}$ makes a query to the index oracle $\mathbf{I}_{\mathrm{r}}(\mathbb{i})$, the new simulator $\mathbf{S}_{\mathrm{r}}$ directly answers the query (as it knows the index and has computed the encoded index so that it knows the answer to all such queries). Whenever $\mathbf{V}_{\mathrm{r}}$ makes a query to a proof oracle, $\mathbf{S}_{\mathrm{r}}$ proceeds as follows. If $\mathbf{V}_{\mathrm{r}}$'s query is to the j-th string in the i-th proof oracle (for $i \in [\mathsf{rc}]$), then since $\mathbf{V}_{\mathrm{r}}$ is semi-honest, it queries the entire j-th string. To simulate the j-th string, $\mathbf{S}_{\mathrm{r}}$ asks $\mathbf{S}$ to simulate the j-th symbol in the i-th proof oracle, which leads to an answer $a_{i,j} \in \Sigma$; then $\mathbf{S}_{\mathrm{r}}$ samples a codeword

$\mathcal{C}(a_{i,j}) \in \Lambda^n$ that agrees with the previous b' answers, and answers all future queries to this string via this sampled codeword. This procedure is efficient because the encoding algorithm for the code $\mathcal{C}$ is efficient.

Since, by assumption, $\mathbf{S}$ simulates the view of $\mathbf{V}$ perfectly, the encoded view is a perfect simulation of the view of $\mathbf{V}_{\mathrm{r}}$. □

B Proof composition

We state a generic lemma about proof composition for IOPs (Lemma B.1). We include the construction for convenience (Construction B.5) but omit the proof; see [BCGRS17] for more details on interactive proof composition for IOPs (which extends the classical notion of non-interactive proof composition for PCPs [AS98]). We also prove properties about how zero knowledge is affected by proof composition (Lemma B.6).

Lemma B.1. *Suppose that the conditions below hold with parameters such that* $\delta_{\mathsf{in}}(\sigma_{\mathsf{out}}(n)) \leq \alpha_{\mathsf{out}}(n)$.

- **Outer IOP with robustness.** *There is a holographic IOP* $(\mathbf{I}_{\mathsf{out}}, \mathbf{P}_{\mathsf{out}}, \mathbf{V}_{\mathsf{out}})$ *with* **non-adaptive** *queries in* $\mathcal{Q}_{\mathrm{point}}$ *for an indexed relation* R *with the following parameters: round complexity* $\mathsf{rc}_{\mathsf{out}}$*; answer alphabet* Σ*; oracle length* $\mathsf{l}_{\mathsf{out}} = \mathsf{li}_{\mathsf{out}} + \mathsf{lp}_{\mathsf{out}}$*; randomness* $\mathsf{r}_{\mathsf{out}}$*; query complexity* $\mathsf{q}_{\mathsf{out}}$*; decision state size* σ_{out}*; soundness error* ϵ_{out}*; robustness* α_{out}*; indexer time* $\mathsf{ti}_{\mathsf{out}}$*; prover time* $\mathsf{tp}_{\mathsf{out}}$*; verifier time* $\mathsf{tv}_{\mathsf{out}} = \mathsf{tq}_{\mathsf{out}} + \mathsf{td}_{\mathsf{out}}$.
- **Inner IOP with proximity.** *There is an IOPP* $(\mathbf{P}_{\mathsf{in}}, \mathbf{V}_{\mathsf{in}})$ *with queries in* $\mathcal{Q}_{\mathrm{point}}$ *for the relation* $R(\mathbf{V}_{\mathsf{out}})$ *with round complexity* $\mathsf{rc}_{\mathsf{in}}$*; answer alphabet* Σ*; proof length* l_{in}*; randomness* r_{in}*; query complexity* q_{in}*; soundness error* ϵ_{in}*; proximity* δ_{in}*; prover time* $\mathsf{tp}_{\mathsf{in}}$*; verifier time* $\mathsf{tv}_{\mathsf{in}}$.

Then the protocol in Construction B.5 (see below) is a holographic IOP $(\mathbf{I}, \mathbf{P}, \mathbf{V})$ *with queries in* $\mathcal{Q}_{\mathrm{point}}$ *for the indexed relation* R *with the following parameters:*

- *round complexity* $\mathsf{rc}(n) = \mathsf{rc}_{\mathsf{out}}(n) + \mathsf{rc}_{\mathsf{in}}(\sigma_{\mathsf{out}}(n))$*;*
- *answer alphabet* Σ*;*
- *index oracle length* $\mathsf{li}(n) = \mathsf{li}_{\mathsf{out}}(n)$*;*
- *proof oracle length* $\mathsf{lp}(n) = \mathsf{lp}_{\mathsf{out}}(n) + \mathsf{l}_{\mathsf{in}}(\sigma_{\mathsf{out}}(n))$*;*
- *randomness* $\mathsf{r}(n) = \mathsf{r}_{\mathsf{out}}(n) + \mathsf{r}_{\mathsf{in}}(\sigma_{\mathsf{out}}(n))$*;*
- *query complexity* $\mathsf{q}(n) = \mathsf{q}_{\mathsf{in}}(\sigma_{\mathsf{out}}(n))$*;*
- *soundness error* $\epsilon(n) = \epsilon_{\mathsf{out}}(n) + \epsilon_{\mathsf{in}}(\sigma_{\mathsf{out}}(n))$*;*
- *indexer time* $\mathsf{ti}(n) = \mathsf{ti}_{\mathsf{out}}(n)$*;*
- *prover time* $\mathsf{tp}(n) = \mathsf{tp}_{\mathsf{out}}(n) + \mathsf{tq}_{\mathsf{out}}(n) + \mathsf{tp}_{\mathsf{in}}(\sigma_{\mathsf{out}}(n))$*;*
- *verifier time* $\mathsf{tv}(n) = \mathsf{tq}_{\mathsf{out}}(n) + \mathsf{tv}_{\mathsf{in}}(\sigma_{\mathsf{out}}(n))$.

If the two proof systems are public coin then so is $(\mathbf{I}, \mathbf{P}, \mathbf{V})$.

We provide some remarks, then state the construction for proof composition (see Construction B.5), and finally prove how zero-knowledge properties are preserved in the construction.

Remark B.2 (outer proximity or inner robustness)**.** If the outer IOP has proximity parameter δ_{out}, then the composed proof system is an IOP with proximity parameter $\delta(n) = \delta_{\mathsf{out}}(n)$. If the inner IOP has robustness parameter α_{in} then the composed proof system is an IOP with robustness parameter $\alpha(n) = \alpha_{\mathsf{in}}(\sigma_{\mathsf{out}}(n))$.

Remark B.3 (different alphabets)**.** Lemma B.1 asserts that the outer and inner proof systems use the same alphabet for the prover messages. This is not essential: if the two alphabets differ, the same argument as above goes through, and the composed proof system will have some messages over one alphabet and other messages over the other alphabet. In particular, in Section 9 the outer proof system will be over a given field $\mathbb{F}$ and the inner proof system will be over the boolean field $\mathbb{F}_2$.

Remark B.4 (on the ZK query bound)**.** The expression for b in Equation (6) may look odd, but it can be intuitively explained. First, if the outer proof system does not satisfy any zero knowledge property ($b_{\mathsf{out}} = 0$)

then the composed proof system also will not ($b = 0$), regardless of the inner proof system. This makes sense since the verifier in the composed proof system does have access to the proof transcript of the outer proof system. So suppose that the outer proof system has a non-trivial zero knowledge property ($b_{\mathsf{out}} > 0$); this property may degrade a little or not at all in the composed proof system, depending on the inner proof system, because it is used to argue that a local view is satisfied. For illustrative purposes, let us consider two cases:

- Suppose that the inner proximity proof satisfies no zero knowledge property, which means that the condition holds with the constant function $f_{\mathsf{in}}(\cdot) := \mathsf{q}_{\mathsf{out}}$, because a witness for the relation $R(\mathbf{V}_{\mathsf{out}})$ consists of $\mathsf{q}_{\mathsf{out}}$ symbols. (All proximity proofs satisfy the "trivial" zero knowledge property where the simulator queries all locations of the witness, which means that f returns the size of the witness.) In this case, it could be that the inner proof system reveals all q_{in} locations of the local view and in addition the verifier can make b queries to the outer proof transcript. We need that $b + \mathsf{q}_{\mathsf{out}} \leq b_{\mathsf{out}}$ which gives $b \leq b_{\mathsf{out}} - \mathsf{q}_{\mathsf{out}}$, i.e., we "lost" $\mathsf{q}_{\mathsf{out}}$ queries.

- Suppose that the inner proximity proof satisfies zero knowledge with $f_{\mathsf{in}}(i) := i$, i.e., the simulator only has to query the witness at i locations whenever a malicious verifier makes i queries across the witness *and proof transcript*. In this case the expression $\max_{j=0,1,\ldots,i}\{j + f_{\mathsf{in}}(i-j)\}$ equals i, which means that we get the better bound $b \leq b_{\mathsf{out}}$, which means no degradation in the query bound.

Construction B.5. The new indexer $\mathbf{I}$ takes as input an index $\mathbb{i}$ and outputs the encoded index $\Pi_0 := \mathbf{I}_{\mathsf{out}}(\mathbb{i})$. The new prover $\mathbf{P}$ receives as input the index $\mathbb{i}$, an instance $\mathbb{x}$, and witness $\mathbb{w}$. The new verifier $\mathbf{V}$ receives as oracle the encoded index Π_0 and as input the instance $\mathbb{x}$. The protocol between $\mathbf{P}$ and $\mathbf{V}$ proceeds as follows.

- *Outer.* The new prover $\mathbf{P}$ and new verifier $\mathbf{V}$ engage in the interactive phase of $(\mathbf{P}_{\mathsf{out}}(\mathbb{i}, \mathbb{x}, \mathbb{w}), \mathbf{V}_{\mathsf{out}}^{\mathbf{I}_{\mathsf{out}}(\mathbb{i})}(\mathbb{x}))$ for the indexed relation R, leading to the outer proof transcript Π_{out} (which is in addition to the encoded index $\mathbf{I}_{\mathsf{out}}(\mathbb{i})$). Note that the interactive phase does *not* include the (non-adaptive) query phase of $\mathbf{V}_{\mathsf{out}}$.

- *Sample local view.* The new verifier $\mathbf{V}$ sends the (whole) randomness string $\mathsf{r}_{\mathsf{out}}$ of the outer verifier $\mathbf{V}_{\mathsf{out}}$ to the new prover $\mathbf{P}$. Both $\mathbf{P}$ and $\mathbf{V}$ compute $(\sigma, I) := \mathbf{V}_{\mathsf{q}_{\mathsf{out}}}(\mathbb{x}, \mathsf{r}_{\mathsf{out}})$. These define an inner instance $\mathbb{x}_{\mathsf{in}} := \sigma$ and inner witness $\mathbb{w}_{\mathsf{in}} := (\mathbf{I}_{\mathsf{out}}(\mathbb{i}), \Pi_{\mathsf{out}})|_I$ for the relation $R(\mathbf{V}_{\mathsf{out}})$ of accepting local views for the outer verifier $\mathbf{V}_{\mathsf{out}}$. Note that the new verifier $\mathbf{V}$ has oracle access to the encoded index $\mathbf{I}_{\mathsf{out}}(\mathbb{i})$ and outer proof transcript Π_{out}, and in particular also to the inner witness $\mathbb{w}_{\mathsf{in}}$.

- *Inner.* The new prover $\mathbf{P}$ and the new verifier $\mathbf{V}$ engage in an execution of $(\mathbf{P}_{\mathsf{in}}(\mathbb{x}_{\mathsf{in}}, \mathbb{w}_{\mathsf{in}}), \mathbf{V}_{\mathsf{in}}^{\mathbb{w}_{\mathsf{in}}}(\mathbb{x}_{\mathsf{in}}))$ for the relation $R(\mathbf{V}_{\mathsf{out}})$. The new verifier $\mathbf{V}$ accepts if and only if the inner verifier $\mathbf{V}_{\mathsf{in}}$ does.

Lemma B.6. *The implications below hold for the proof composition in Lemma B.1.*

1. **Semi-honest-verifier zero knowledge.** *If* $(\mathbf{I}_{\mathsf{out}}, \mathbf{P}_{\mathsf{out}}, \mathbf{V}_{\mathsf{out}})$ *is zero knowledge against semi-honest verifiers, then so is* $(\mathbf{I}, \mathbf{P}, \mathbf{V})$ *(regardless of any zero-knowledge properties of* $(\mathbf{P}_{\mathsf{in}}, \mathbf{V}_{\mathsf{in}})$*).*

2. **Bounded-query zero knowledge.** *If* $(\mathbf{I}_{\mathsf{out}}, \mathbf{P}_{\mathsf{out}}, \mathbf{V}_{\mathsf{out}})$ *is zero knowledge with query bound* b_{out} *and* $(\mathbf{P}_{\mathsf{in}}, \mathbf{V}_{\mathsf{in}})$ *is zero knowledge with query function* f_{in} *then* $(\mathbf{I}, \mathbf{P}, \mathbf{V})$ *is zero knowledge with query bound*

$$b := \max\left\{ i \in \mathbb{N} \;\middle|\; \max_{j=0,1,\ldots,i}\{j + f_{\mathsf{in}}(i-j)\} \leq b_{\mathsf{out}} \right\} \quad . \tag{6}$$

Proof of Item 2 in Lemma B.6. Recall the zero knowledge properties of the outer and inner proof systems:

- By the zero-knowledge property of $(\mathbf{P}_{\mathsf{out}}, \mathbf{V}_{\mathsf{out}})$, there is an efficient simulator $\mathbf{S}_{\mathsf{out}}$ such that, for every index-instance-witness triple $(\mathbb{i}, \mathbb{x}, \mathbb{w}) \in R$ and b_{out}-query malicious verifier $\tilde{\mathbf{V}}_{\mathsf{out}}$, the random variables $\mathbf{S}_{\mathsf{out}}^{\tilde{\mathbf{V}}_{\mathsf{out}}}(\mathbb{i}, \mathbb{x})$ and $\mathrm{View}(\mathbf{P}_{\mathsf{out}}(\mathbb{i}, \mathbb{x}, \mathbb{w}), \tilde{\mathbf{V}}_{\mathsf{out}})$ are identically distributed.
- By the zero-knowledge property of $(\mathbf{P}_{\mathsf{in}}, \mathbf{V}_{\mathsf{in}})$, there is an efficient simulator $\mathbf{S}_{\mathsf{in}}$ such that, for every instance-witness pair $(\mathbb{x}_{\mathsf{in}}, \mathbb{w}_{\mathsf{in}}) \in R(\mathbf{V}_{\mathsf{out}})$ (the relation of accepting local views for $\mathbf{V}_{\mathsf{out}}$), query bound $b \in \mathbb{N}$, and b-query malicious verifier $\tilde{\mathbf{V}}_{\mathsf{in}}$, the random variables $\mathbf{S}_{\mathsf{in}}^{\tilde{\mathbf{V}}_{\mathsf{in}}, \mathbb{w}_{\mathsf{in}}}(\mathbb{x}_{\mathsf{in}})$ and $\mathrm{View}(\mathbf{P}_{\mathsf{in}}(\mathbb{x}_{\mathsf{in}}, \mathbb{w}_{\mathsf{in}}), \tilde{\mathbf{V}}_{\mathsf{in}}^{\mathbb{w}_{\mathsf{in}}})$ are identically distributed and $\mathbf{S}_{\mathsf{in}}$ makes at most $f_{\mathsf{in}}(b)$ queries to $\mathbb{w}_{\mathsf{in}}$.

We use the above simulators to construct an efficient simulator $\mathbf{S}$ for the composed proof system $(\mathbf{I}, \mathbf{P}, \mathbf{V})$.

Let $\tilde{\mathbf{V}}$ be a b-query malicious verifier for $(\mathbf{I}, \mathbf{P}, \mathbf{V})$ whose view $\mathbf{S}$ must simulate. Unlike the honest verifier, $\tilde{\mathbf{V}}$ may arbitrarily query any of the proof strings sent by the new prover $\mathbf{P}$. Also, $\tilde{\mathbf{V}}$ does not make queries to the encoded index $\mathbf{I}(\mathbb{i})$, as $\tilde{\mathbf{V}}$ can itself apply the (deterministic) indexer $\mathbf{I}$ to the index $\mathbb{i}$.

The new simulator $\mathbf{S}$ receives as input an index $\mathbb{i}$ and instance $\mathbb{x}$, and computes the encoded index $\mathbf{I}_{\mathsf{out}}(\mathbb{i})$ for later use. Then $\mathbf{S}$ starts running $\tilde{\mathbf{V}}$ and simulates its view by relying on executions of the outer simulator $\mathbf{S}_{\mathsf{out}}^{\tilde{\mathbf{V}}_{\mathsf{out}}}(\mathbb{i}, \mathbb{x})$ and the inner simulator $\mathbf{S}_{\mathsf{in}}^{\tilde{\mathbf{V}}_{\mathsf{in}}, \mathbb{w}_{\mathsf{in}}}(\mathbb{x}_{\mathsf{in}})$ as described below. Note that the inner instance $\mathbb{x}_{\mathsf{in}}$ is explicitly defined, while the outer verifier $\tilde{\mathbf{V}}_{\mathsf{out}}$, inner verifier $\tilde{\mathbf{V}}_{\mathsf{in}}$, and inner witness $\mathbb{w}_{\mathsf{in}}$ are implicitly defined.

- *Simulate outer proof system.* Whenever $\tilde{\mathbf{V}}$ outputs a message for $\mathbf{P}_{\mathsf{out}}$, $\mathbf{S}$ forwards this message to $\mathbf{S}_{\mathsf{out}}$. Whenever $\tilde{\mathbf{V}}$ makes a query to a proof string of the outer proof system (i.e., to a location in Π_{out}), $\mathbf{S}$ forwards this query to $\mathbf{S}_{\mathsf{out}}$ and replies with the simulated answer provided by $\mathbf{S}_{\mathsf{out}}$. (Note that $\tilde{\mathbf{V}}$ may make queries during this phase, even though the honest verifier $\mathbf{V}$ does not.)
- *Determine malicious local view.* After the interactive phase of the outer proof system, $\tilde{\mathbf{V}}$ outputs a malicious choice of randomness string $\tilde{\mathsf{r}}_{\mathsf{out}}$. The simulator $\mathbf{S}$ runs the outer query sampler $\mathbf{V}_{\mathsf{q}_{\mathsf{out}}}(\mathbb{x}, \tilde{\mathsf{r}}_{\mathsf{out}})$ to obtain the inner instance $\mathbb{x}_{\mathsf{in}} := \tilde{\sigma}$ and the positions of the local view $\tilde{I}$ (which are positions in $\mathbf{I}_{\mathsf{out}}(\mathbb{i})$ and Π_{out}).
- *Simulate inner proof system.* Whenever $\tilde{\mathbf{V}}$ outputs a message for $\mathbf{P}_{\mathsf{in}}$, $\mathbf{S}$ forwards this message to $\mathbf{S}_{\mathsf{in}}$. Whenever $\tilde{\mathbf{V}}$ makes a query to a proof string of the inner proof system, $\mathbf{S}$ forwards this query to $\mathbf{S}_{\mathsf{in}}$ and replies with the simulated answer provided by $\mathbf{S}_{\mathsf{in}}$. Note that this may in turn cause $\mathbf{S}_{\mathsf{in}}$ to query some locations in $\mathbb{w}_{\mathsf{in}}$ (which is the encoded index $\mathbf{I}_{\mathsf{out}}(\mathbb{i})$ and outer proof transcript Π_{out} restricted to positions in $\tilde{I}$). For all such queries, if the query is to $\mathbf{I}_{\mathsf{out}}(\mathbb{i})$ then $\mathbf{S}$ answers it directly (as it has the index and already has computed the encoded index); if instead the query is to Π_{out} then $\mathbf{S}$ forwards the query to $\mathbf{S}_{\mathsf{out}}$ and replies with the resulting simulated answer. Moreover, in this phase $\tilde{\mathbf{V}}$ may still choose to directly query a location of the outer proof transcript Π_{out}, and again $\mathbf{S}$ relies on $\mathbf{S}_{\mathsf{out}}$ to simulate the answer to such queries.

Suppose that the b-query verifier $\tilde{\mathbf{V}}$ makes j queries to the outer proof transcript Π_{out} and $b - j$ queries to the inner proof transcript Π_{in}. The inner simulator $\mathbf{S}_{\mathsf{in}}$ will make $f_{\mathsf{in}}(b - j)$ queries to the inner witness $\mathbb{w}_{\mathsf{in}}$, and $\mathbf{S}$ will forward up to $f_{\mathsf{in}}(b - j)$ queries to the outer simulator $\mathbf{S}_{\mathsf{out}}$. (Some queries to $\mathbb{w}_{\mathsf{in}}$ are to the encoded $\mathbf{I}_{\mathsf{out}}(\mathbb{i})$, and those are not forwarded to $\mathbf{S}_{\mathsf{out}}$.) This means that in total $\mathbf{S}_{\mathsf{out}}$ will have to simulate up to $j + f_{\mathsf{in}}(b - j)$ queries. Since $\tilde{\mathbf{V}}$ gets to choose the value of j, $\mathbf{S}_{\mathsf{out}}$ will have to simulate, in the worst case, $\max_{j=0,1,\ldots,b}\{j + f_{\mathsf{in}}(b - j)\}$ queries. As long as the outer query bound b_{out} is at least this worst-case bound, the outer simulator $\mathbf{S}_{\mathsf{out}}$ will succeed in the simulation, and in turn so will the new simulator $\mathbf{S}$. □

Proof of Item 1 of Lemma B.6. The semi-honest-verifier zero-knowledge property of $(\mathbf{P}_{\mathsf{out}}, \mathbf{V}_{\mathsf{out}})$ states that there is an efficient simulator $\mathbf{S}_{\mathsf{out}}$ such that, for every index-instance-witness triple $(\mathbb{i}, \mathbb{x}, \mathbb{w}) \in R$ and every choice of randomness ρ_{out} for $\mathbf{V}_{\mathsf{out}}$, the following random variables are identically distributed:

$$\mathbf{S}_{\mathsf{out}}^{\mathbf{V}_{\mathsf{out}}(\mathbb{i}, \mathbb{x}; \rho_{\mathsf{out}})}(\mathbb{i}, \mathbb{x}) \quad \text{and} \quad \mathrm{View}(\mathbf{P}_{\mathsf{out}}(\mathbb{i}, \mathbb{x}, \mathbb{w}), \mathbf{V}_{\mathsf{out}}(\mathbb{i}, \mathbb{x}; \rho_{\mathsf{out}})) \ .$$

We use the simulator $\mathbf{S}_{\mathsf{out}}$ to construct an efficient simulator $\mathbf{S}$ for the composed proof system $(\mathbf{I}, \mathbf{P}, \mathbf{V})$.

The new simulator $\mathbf{S}$ receives as input an index $\mathbb{i}$ and instance $\mathbb{x}$, and computes the encoded index $\mathbf{I}_{\mathsf{out}}(\mathbb{i})$ for later use. Then $\mathbf{S}$ starts running $\mathbf{V}$ and simulates its view by relying on executions of the outer simulator $\mathbf{S}_{\mathsf{out}}^{\mathbf{V}_{\mathsf{out}}(\mathbb{i},\mathbb{x};\rho_{\mathsf{out}})}(\mathbb{i}, \mathbb{x})$. Note that the inner instance $\mathbb{x}_{\mathsf{in}}$ is explicitly defined, while the outer verifier $\mathbf{V}_{\mathsf{out}}$, inner verifier $\mathbf{V}_{\mathsf{in}}$, and inner witness $\mathbb{w}_{\mathsf{in}}$ are implicitly defined.

- *Simulate outer proof system.* Whenever $\mathbf{V}$ outputs a message for $\mathbf{P}_{\mathsf{out}}$, $\mathbf{S}$ forwards this message to $\mathbf{S}_{\mathsf{out}}$. Since $\mathbf{V}$ is semi-honest, $\mathbf{V}$ does not make any queries during this phase.

- *Determine semi-honest local view.* After the interactive phase of the outer proof system, $\mathbf{V}$ outputs a choice of randomness string $\mathsf{r}_{\mathsf{out}}$ (which can be distributed arbitrarily). The simulator $\mathbf{S}$ runs the outer query sampler $\mathbf{V}_{\mathsf{q}_{\mathsf{out}}}(\mathbb{x}, \mathsf{r}_{\mathsf{out}})$ to obtain the inner instance $\mathbb{x}_{\mathsf{in}} := \sigma$, the positions of the local view I (which are positions in $\mathbf{I}_{\mathsf{out}}(\mathbb{i})$ and Π_{out}), and the view of $\mathbf{V}_{\mathsf{out}}$, which forms the witness $\mathbb{w}_{\mathsf{in}}$ for the inner proof system.

- *Simulate inner proof system.* The simulator $\mathbf{S}$ runs the inner prover $\mathbf{P}_{\mathsf{in}}$, and mediates an interaction between $\mathbf{V}$ and $\mathbf{P}_{\mathsf{in}}$. Whenever $\mathbf{V}$ outputs a message for $\mathbf{P}_{\mathsf{in}}$, $\mathbf{S}$ forwards this message to $\mathbf{P}_{\mathsf{in}}$. Whenever $\mathbf{V}$ makes a query to a proof string of the inner proof system, $\mathbf{S}$ answers this query using oracles output by $\mathbf{P}_{\mathsf{in}}$. The verifier $\mathbf{V}$ may also query some locations in the encoded index $\mathbf{I}_{\mathsf{out}}(\mathbb{i})$ and outer proof transcript Π_{out} restricted to positions in I (which forms $\mathbb{w}_{\mathsf{in}}$). For all such queries, if the query is to $\mathbf{I}_{\mathsf{out}}(\mathbb{i})$ then $\mathbf{S}$ answers it directly (as it has the index and has already computed the encoded index); if instead the query is to Π_{out} then $\mathbf{S}$ forwards the query to $\mathbf{S}_{\mathsf{out}}$ and replies with the resulting simulated answer. Note that since $\mathbf{V}$ is semi-honest, no other locations of the outer proof transcript Π_{out} are queried.

By assumption, $\mathbf{S}_{\mathsf{out}}$ simulates the view of $\mathbf{V}_{\mathsf{out}}$ perfectly. In either a simulated proof, or a real proof, the view of $\mathbf{V}$ is obtained by running $\mathbf{P}_{\mathsf{in}}$ on the view of $\mathbf{V}_{\mathsf{out}}$. Thus, $\mathbf{S}$ provides a perfect simulation. Since $\mathbf{S}_{\mathsf{out}}$ and $\mathbf{P}_{\mathsf{in}}$ are efficient, so is $\mathbf{S}$. □

C Equivalence of zero-knowledge code definitions

The following is a reformulation of [Wei16, Definition 2.3.2].

Definition C.1. *Let $\mathcal{C}$ be a randomized linear code with randomized encoding function* Enc*. Let $b \in \mathbb{N}$ be a query bound and $\epsilon > 0$ a statistical distance parameter. We say that $\mathcal{C}$ is (b, ϵ)-ZK if for every set $J \subseteq [n]$ of size at most b, and every message pair $m, m' \in \mathbb{F}^{k_m}$, the statistical distance between $\mathrm{Enc}(m)|_J$ and $\mathrm{Enc}(m')|_J$ is at most ϵ.*

Lemma C.2 and Lemma C.3 show that Definition C.1 and Definition 3.21 are equivalent when $\mathcal{C}$ has a polynomial-time encoding function. However, Definition 3.21 may be stronger, for example, for codes with an exponential-size block length for which a polynomial number of entries can be efficiently simulated.

Lemma C.2. *If a randomized linear code $\mathcal{C}$ is b-query zero-knowledge then it is $(b, 0)$-ZK.*

Proof. The distribution of $\mathcal{S}^A$ equals $\{\mathrm{View}(\mathrm{Enc}(m; r), A)\}_{r \leftarrow \mathbb{F}^{k_r}}$ for any message $m \in \mathbb{F}^{k_m}$ and any b-query verifier A, so specializing to A which queries a set J demonstrates that $\mathrm{Enc}(m)|_J$ is identically distributed for any query set. □

Lemma C.3. *If a randomized linear code $\mathcal{C}$ is $(b, 0)$-ZK, then it is b-query zero-knowledge. Moreover, the running time of the simulator $\mathcal{S}$ is the time taken to compute* $\mathrm{Enc}(0)$.

Proof. The simulator $\mathcal{S}$ simply computes the codeword $\mathrm{Enc}(0)$ and answers the verifier's queries with the entries of $\mathrm{Enc}(0)$. Since the distribution of $\mathrm{Enc}(0)$ is identical to that of $\mathrm{Enc}(m)$ for any message m, the simulation is perfect. □

Acknowledgments

This research was supported in part by: the Berkeley Haas Blockchain Initiative and a donation from the Ethereum Foundation. Part of the work was conducted while the first author was employed by UC Berkeley.

References

[AHIKV17] Benny Applebaum, Naama Haramaty, Yuval Ishai, Eyal Kushilevitz, and Vinod Vaikuntanathan. "Low-Complexity Cryptographic Hash Functions". In: *Proceedings of the 8th Innovations in Theoretical Computer Science Conference*. ITCS '17. 2017, 7:1–7:31.

[AHIV17] Scott Ames, Carmit Hazay, Yuval Ishai, and Muthuramakrishnan Venkitasubramaniam. "Ligero: Lightweight Sublinear Arguments Without a Trusted Setup". In: *Proceedings of the 24th ACM Conference on Computer and Communications Security*. CCS '17. 2017, pp. 2087–2104.

[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.

[BBBPWM18] Benedikt Bünz, Jonathan Bootle, Dan Boneh, Andrew Poelstra, Pieter Wuille, and Greg Maxwell. "Bulletproofs: Short Proofs for Confidential Transactions and More". In: *Proceedings of the 39th IEEE Symposium on Security and Privacy*. S&P '18. 2018, pp. 315–334.

[BBHR19] Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, and Michael Riabzev. "Scalable Zero Knowledge with No Trusted Setup". In: *Proceedings of the 39th Annual International Cryptology Conference*. CRYPTO '19. 2019, pp. 733–764.

[BCC88] Gilles Brassard, David Chaum, and Claude Crépeau. "Minimum disclosure proofs of knowledge". In: *Journal of Computer and System Sciences* 37.2 (1988), pp. 156–189.

[BCCGP16] Jonathan Bootle, Andrea Cerulli, Pyrros Chaidos, Jens Groth, and Christophe Petit. "Efficient Zero-Knowledge Arguments for Arithmetic Circuits in the Discrete Log Setting". In: *Proceedings of the 35th Annual International Conference on Theory and Application of Cryptographic Techniques*. EUROCRYPT '16. 2016, pp. 327–357.

[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.

[BCG20] Jonathan Bootle, Alessandro Chiesa, and Jens Groth. "Linear-Time Arguments with Sublinear Verification from Tensor Codes". In: *Proceedings of the 18th Theory of Cryptography Conference*. TCC '20. 2020, ??–??

[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: *Proceedings of the 23rd International Conference on the Theory and Applications of Cryptology and Information Security*. ASIACRYPT '17. 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.

[BCGT13] 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.

[BCIOP13] Nir Bitansky, Alessandro Chiesa, Yuval Ishai, Rafail Ostrovsky, and Omer Paneth. "Succinct Non-Interactive Arguments via Linear Interactive Proofs". In: *Proceedings of the 10th Theory of Cryptography Conference*. TCC '13. 2013, pp. 315–333.

[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.

[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.

[BSCIKS20] Eli Ben-Sasson, Dan Carmon, Yuval Ishai, Swastik Kopparty, and Shubhangi Saraf. *Proximity Gaps for Reed-Solomon Codes*. Cryptology ePrint Archive, Report 2020/654. `https://eprint.iacr.org/2020/654`. 2020.

[CCGHV07] Hao Chen, Ronald Cramer, Shafi Goldwasser, Robbert de Haan, and Vinod Vaikuntanathan. "Secure Computation from Random Error Correcting Codes". In: *Proceedings of the 26th Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Eurocrypt' 07. 2007, pp. 291–310.

[CHMMVW20] Alessandro Chiesa, Yuncong Hu, Mary Maller, Pratyush Mishra, Noah Vesely, and Nicholas Ward. "Marlin: Preprocessing zkSNARKs with Universal and Updatable SRS". In: *Proceedings of the 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques*. EUROCRYPT '20. 2020, pp. 738–768.

[COS20] Alessandro Chiesa, Dev Ojha, and Nicholas Spooner. "Fractal: Post-Quantum and Transparent Recursive Proofs from Holography". In: *Proceedings of the 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques*. EUROCRYPT '20. 2020, pp. 769–793.

[Cer19] Andrea Cerulli. "Efficient zero-knowledge proofs and their applications". 2019.

[Cha+17] Melissa Chase, David Derler, Steven Goldfeder, Claudio Orlandi, Sebastian Ramacher, Christian Rechberger, Daniel Slamanig, and Greg Zaverucha. "Post-Quantum Zero-Knowledge and Signatures from Symmetric-Key Primitives". In: *Proceedings of the 24th ACM Conference on Computer and Communications Security*. CCS '17. 2017, pp. 1825–1842.

[DI14] Erez Druk and Yuval Ishai. "Linear-time encodable codes meeting the Gilbert–Varshamov bound and their cryptographic applications". In: *Proceedings of the 5th Innovations in Theoretical Computer Science Conference*. ITCS '14. 2014, pp. 169–182.

[GGPR13] Rosario Gennaro, Craig Gentry, Bryan Parno, and Mariana Raykova. "Quadratic Span Programs and Succinct NIZKs without PCPs". In: *Proceedings of the 32nd Annual International Conference on Theory and Application of Cryptographic Techniques*. EUROCRYPT '13. 2013, pp. 626–645.

[GH98] Oded Goldreich and Johan Håstad. "On the complexity of interactive proofs with bounded communication". In: *Information Processing Letters* 67.4 (1998), pp. 205–214.

[GIMS10] Vipul Goyal, Yuval Ishai, Mohammad Mahmoody, and Amit Sahai. "Interactive locking, zero-knowledge PCPs, and unconditional cryptography". In: *Proceedings of the 30th Annual Conference on Advances in Cryptology*. CRYPTO'10. 2010, pp. 173–190.

[GMO16] Irene Giacomelli, Jesper Madsen, and Claudio Orlandi. "ZKBoo: Faster Zero-Knowledge for Boolean Circuits". In: *Proceedings of the 25th USENIX Security Symposium*. Security '16. 2016, pp. 1069–1083.

[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.

[GVW02] Oded Goldreich, Salil Vadhan, and Avi Wigderson. "On interactive proofs with a laconic prover". In: *Computational Complexity* 11.1/2 (2002), pp. 1–53.

[HK20] David Heath and Vladimir Kolesnikov. "Stacked Garbling for Disjunctive Zero-Knowledge Proofs". In: *Proceedings of the 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques*. EUROCRYPT '20. 2020, pp. 569–598.

[IKOS07] Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. "Zero-knowledge from secure multiparty computation". In: STOC'07 (2007), pp. 21–30.

[IMS12] Yuval Ishai, Mohammad Mahmoody, and Amit Sahai. "On Efficient Zero-Knowledge PCPs". In: *Proceedings of the 9th Theory of Cryptography Conference on Theory of Cryptography*. TCC '12. 2012, pp. 151–168.

[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.

[ISVW13] Yuval Ishai, Amit Sahai, Michael Viderman, and Mor Weiss. "Zero Knowledge LTCs and Their Applications". In: *Proceedings of the 16th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems, and of the 17th International Workshop on Randomization and Computation*. APPROX-RANDOM '13. 2013, pp. 607–622.

[IW14] Yuval Ishai and Mor Weiss. "Probabilistically Checkable Proofs of Proximity with Zero-Knowledge". In: *Proceedings of the 11th Theory of Cryptography Conference*. TCC '14. 2014, pp. 121–145.

[KKB88] Michael Kaminski, David Kirkpatrick, and Nader Bshouty. "Addition Requirements for Matrix and Transposed Matrix Products". In: *Journal of Algorithms* 9.3 (1988), pp. 354–364.

[KKW18] Jonathan Katz, Vladimir Kolesnikov, and Xiao Wang. "Improved Non-Interactive Zero Knowledge with Applications to Post-Quantum Signatures". In: *Proceedings of the 25th ACM Conference on Computer and Communications Security*. CCS '18. 2018, pp. 525–537.

[KMP20] Abhiram Kothapalli, Elisaweta Masserova, and Bryan Parno. *A Direct Construction for Asymptotically Optimal zkSNARKs*. Cryptology ePrint Archive, Report 2020/1318. 2020.

[KPT97] Joe Kilian, Erez Petrank, and Gábor Tardos. "Probabilistically checkable proofs with zero knowledge". In: *Proceedings of the 29th Annual ACM Symposium on Theory of Computing*. STOC '97. 1997, pp. 496–505.

[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.

[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.

[RR20] Noga Ron-Zewi and Ron Rothblum. "Local Proofs Approaching the Witness Length". In: *Proceedings of the 61st Annual IEEE Symposium on Foundations of Computer Science*. FOCS '20. 2020.

[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.

[SL20] Srinath Setty and Jonathan Lee. *Quarks: Quadruple-efficient transparent zkSNARKs*. Cryptology ePrint Archive, Report 2020/1275. 2020.

[Set20] Srinath Setty. "Spartan: Efficient and general-purpose zkSNARKs without trusted setup". In: *Proceedings of the 40th Annual International Cryptology Conference*. CRYPTO '20. Referencing Cryptology ePrint Archive, Report 2019/550, revision from 2020.02.28. 2020.

[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.

[Tha13] Justin Thaler. "Time-Optimal Interactive Proofs for Circuit Evaluation". In: *Proceedings of the 33rd Annual International Cryptology Conference*. CRYPTO '13. 2013, pp. 71–89.

[WTSTW18] Riad S. Wahby, Ioanna Tzialla, Abhi Shelat, Justin Thaler, and Michael Walfish. "Doubly-efficient zkSNARKs without trusted setup". In: *Proceedings of the 39th IEEE Symposium on Security and Privacy*. S&P '18. 2018, pp. 926–943.

[WYKW20] Chenkai Weng, Kang Yang, Jonathan Katz, and Xiao Wang. "Fast, Scalable, and Communication-Efficient Zero-Knowledge Proofs for Boolean and Arithmetic Circuits". In: (2020).

[Wei16] Mor Weiss. "Secure Computation and Probabilistic Checking". 2016.

[XZZPS19] Tiancheng Xie, Jiaheng Zhang, Yupeng Zhang, Charalampos Papamanthou, and Dawn Song. "Libra: Succinct Zero-Knowledge Proofs with Optimal Prover Computation". In: *Proceedings of the 39th Annual International Cryptology Conference*. CRYPTO '19. 2019, pp. 733–764.

[ZWZZ20] Jiaheng Zhang, Weijie Wang, Yinuo Zhang, and Yupeng Zhang. *Doubly Efficient Interactive Proofs for General Arithmetic Circuits with Linear Prover Time*. Cryptology ePrint Archive, Report 2020/1247. 2020.

[ZXZS20] Jiaheng Zhang, Tiancheng Xie, Yupeng Zhang, and Dawn Song. "Transparent Polynomial Delegation and Its Applications to Zero Knowledge Proof". In: *Proceedings of the 41st IEEE Symposium on Security and Privacy*. S&P '20. 2020, pp. 859–876.