Zero-Knowledge IOPs with Linear-Time Prover and Polylogarithmic-Time Verifier *

Jonathan Bootle[1][0000−0003−3582−3368], Alessandro Chiesa[2,3] and Siqi Liu[3]

[1] IBM Research, Zurich, Switzerland
jbt@zurich.ibm.com
[2] École Polytechnique Fédérale de Lausanne, Lausanne, Switzerland
alessandro.chiesa@epfl.ch
[3] University of California, Berkeley, Berkeley, USA
sliu18@berkeley.edu

Abstract. Interactive oracle proofs (IOPs) are a multi-round generalization of probabilistically checkable proofs that play a fundamental role in the construction of efficient cryptographic proofs.
We present an IOP that simultaneously achieves the properties of zero knowledge, linear-time proving, and polylogarithmic-time verification. We construct a zero-knowledge 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).
Our result implies progress on a basic goal in the area of efficient zero knowledge. Via a known transformation, we obtain a zero knowledge argument system where the prover runs in linear time and the verifier runs in polylogarithmic time; the construction is plausibly post-quantum and only makes a black-box use of lightweight cryptography (collision-resistant hash functions).

Keywords: interactive oracle proofs; zero knowledge; succinct arguments

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.

* The full version of this paper is available at https://eprint.iacr.org/2020/1527.

Several works (e.g., [IKOS07; GMO16; CDG+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.[4] 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 one 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.[5]

Succinctness. In a seminal work, Kilian [Kil92] constructed zero knowledge arguments that are *succinct*: communication complexity and verifier time are *polylogarithmic* in computation size. While these are essentially optimal, the prover in Kilian's construction is a polynomial-time algorithm that fails to achieve the asymptotically-optimal linear time achieved via the aforementioned (non-succinct) zero knowledge proofs. Improving the prover time in succinct arguments has been a major goal in a subsequent line of work.

Essentially all approaches for constructing succinct arguments follow the same high-level template: first construct a probabilistic proof in some proof model, and then make a black-box use of cryptography to compile the probabilistic proof into an argument system.

A notable exception are zero-knowledge arguments with a linear-time prover and a polylogarithmic-time verifier. This goal is presently achieved as a consequence of the zero-knowledge argument in [BCG+17b] (see Section 1.3), but only via a non-black-box use of cryptography. This is unfortunate, as black-box results are a cryptographic "gold standard" that typically reflect a a deeper understanding, and over time lead to more efficient solutions (once each black-box is suitably optimized), when compared to non-black-box results.

Interactive oracle proofs. The above status quo is due to inefficiencies in probabilistic proofs. Prior results on zero-knowledge argument systems with a linear-time prover and sublinear-time verifier rely on compiling interactive oracle proofs (IOPs) [BCS16; RRR16] into corresponding succinct arguments via a black-box use of suitable collision-resistant hash functions. The verifier time was sublinear rather than polylogarithmic due to the underlying IOP constructions. In particular, the following basic question has remained open:

Do there exist zero-knowledge IOPs with a linear-time prover and a polylogarithmic-time verifier?

In this paper we give a positive answer to this question for arithmetic computations over a large field, and obtain a corresponding black-box result about zero-knowledge

[4] 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.

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

succinct arguments. The question of whether an analogous result can be proved for boolean computations remains an exciting open problem.

IOP	encode circuit cost	prover cost	verifier cost	query complexity	zero-knowledge
[BCG+17b]	$O(n)$ $\mathbb{F}$-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

Fig. 1. Comparison of known IOPs with a linear-time prover, for soundness error $1/2$. 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. Sublinear verification is achieved in the holographic setting (the verifier has oracle access to an encoding of the circuit).

1.1 Our results

Our main result is an interactive oracle proof (IOP) [BCS16; RRR16] that simultaneously achieves zero knowledge, linear-time proving, and polylogarithmic-time verification (so also linear proof length and polylogarithmic query complexity). This implies the first zero-knowledge argument system with linear-time proving and polylogarithmic-time verification (and thus polylogarithmic communication complexity) that makes a black-box use of cryptography. Jumping ahead, our solution uses a lightweight cryptographic primitive (linear-time collision-resistant hash functions) for which there are plausibly post-quantum candidates.

IOP for R1CS. Our IOP 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. This NP-complete problem is widely used in the probabilistic proof literature (and beyond) because it efficiently expresses arithmetic circuits[6] and is convenient to use when designing a succinct argument.

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,[7] 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.)*

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.

[6] 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.)

[7] 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.

We construct an IOP for the R1CS problem with the parameters below. Our result significantly improves over prior linear-time IOPs, as summarized in Figure 1 and further discussed in Section 1.2.

Theorem 1 (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 is $O(1)$.*

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

Succinct argument for R1CS. The above theorem directly implies a zero-knowledge succinct argument with a linear-time prover and polylogarithmic-time verifier, obtained in a black-box way under standard cryptographic assumptions. The implication involves combining IOPs and linear-time collision resistant hashing [BCG+17b], as reviewed in Section 2.7.

In more detail, the result below relies on any linear-time collision-resistant hash function. Such hash functions are known to exist, e.g., under certain assumptions about finding short codewords in linear codes [AHI+17]; 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 2 (informal)**.** *Using any linear-time collision-resistant hash function with security parameter λ as a black box, one can obtain an 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 with private coins.[8]

On zero knowledge. The notion of *semi-honest-verifier* zero-knowledge in Theorem 1 means that the IOP prover leaks no information to an honest IOP verifier for any choice of verifier randomness. This suffices for malicious-verifier zero-knowledge in Theorem 2, as explained in Section 2.7. We also present results (see Section 2.6) that allow us to prove a variant of Theorem 1 where the IOP satisfies the stronger property of

[8] The private coins come from using the Goldreich–Kahan technique [GK96]. Achieving public coins is also possible via different relaxations: (i) we could rely on a reference string (which enables the zero knowledge simulator to access a trapdoor); or (ii) we could relax the goal to honest-verifier zero-knowledge while remaining in the plain model. See [IMSX15] for more on these considerations.

bounded-query zero-knowledge, but at the cost of a sublinear verifier time rather than polylogarithmic. Bounded-query zero-knowledge is the hiding notion typically studied for PCPs [KPT97], and often enables reductions in communication complexity when compiling the IOP into a succinct argument. The aforementioned loss in verifier time only comes from the fact that known constructions of "zero knowledge codes" with a linear-time encoder are probabilistic, and the loss could be avoided by derandomizing such families — overcoming this barrier remains an exciting open problem in coding theory.

On sublinear verification. The polylogarithmic verifier time in Theorem 1 is achieved in the holographic setting, which means that the verifier is given query access to a linear-length encoding of the coefficient matrices that is computable in linear time. Similarly, polylogarithmic verifier time in Theorem 2 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). This follows a general paradigm wherein holographic proofs lead to preprocessing arguments [CHM+20; COS20]. Holography/preprocessing is necessary for sublinear verification in the general case because just reading the R1CS instance takes linear time.[9]

Open questions. Our IOP works for satisfiability problems over fields of at least linear size, as is the case for all known linear-time IOPs (see Section 1.2); obtaining analogous results for all fields, or just the boolean field, is open. Moreover, our IOP achieves constant soundness error, and the question of additionally achieving a sub-constant soundness error (ideally, negligible in a security parameter) is open. Finally, while our focus is asymptotic efficiency, we are optimistic that the ideas in this paper will facilitate further research that may additionally achieve good concrete efficiency. (We point to specific ideas for this in Section 2.) Initial progress in this direction has been made in subsequent work discussed in Section 1.3.

1.2 Related work on probabilistic proofs

As our main result concerns IOPs, we summarize prior works on probabilistic proofs that study related questions. Further connections to prior work are given in Section 2 where we overview our techniques.

First we discuss a line of work on probabilistic proofs with linear proof length, a necessary condition for a linear-time prover (our goal). The first result was [BKK+13], which provides a PCP for boolean circuit satisfiability with linear proof length and sublinear query complexity; this is the only known result for PCPs, and constructing PCPs with linear proof length and polylogarithmic query complexity remains a major open problem. Subsequently, [BCG+17a] obtained a 3-round IOP for boolean circuit satisfiability with linear proof length and constant query complexity; and [RR20] showed how to reduce the multiplicative constant in the proof length to arbitrarily close to 1 at the cost of a slightly larger constant round complexity. None of these works study linear-time proving or sublinear-time verification. Here we omit a discussion of numerous works

[9] Holography/preprocessing may be avoidable by focusing on R1CS instances with a short description [BCG+19] or, more generally, uniform models of computation. Achieving results analogous to ours in such a setting remains an open problem.

that achieve IOPs with linear size, but not linear prover time, for many other models of computation.

Next, [BCG+17b] obtained a zero-knowledge IOP for arithmetic circuit satisfiability with linear-time prover and square-root-time verifier. Then [BCG20] improved the verifier time to any sublinear polynomial, but without zero knowledge. *We improve on this by simultaneously achieving the properties of zero knowledge and polylogarithmic-time verifier.* All of these results require working over a finite field of linear size, and analogous results for boolean circuits are not known. See Figure 1 for a table comparing these latter works.

Recurring tools across many of these works, as well as this paper, include: the sumcheck protocol for tensor codes [Mei13], proof composition (for PCPs [AS98] and for IOPs [BCG+17a]), the linear-time sumcheck [Tha13], and the use of codes without the multiplication property. (The property states that coordinate-wise multiplication of codewords yields codewords in a code whose relative distance is still good.)

The main challenge in designing IOPs with linear-time provers is that one cannot use "useful" codes like the Reed–Solomon code since the encoding time is quasilinear. Instead, prior works resorted 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. (See [Mei12; Mei13] for more on why the multiplication property is useful in constructing probabilistic proofs.)

Our zero-knowledge IOPs with linear-time prover and polylogarithmic-time verifier achieve a central goal in the area of probabilistic proofs, and to construct them we contribute several novel pieces all towards zero knowledge: (i) constructions of linear-time-encodable codes that satisfy a zero-knowledge property; (ii) structural results on the tensor products of codes that satisfy the zero-knowledge property; (iii) a tensor-query zero-knowledge holographic IOP for R1CS with low randomness complexity; (iv) results on zero knowledge preservation under proof composition.

1.3 Related work on succinct arguments

Our main result implies a result on succinct arguments, and below we summarize prior works relevant to that.

A non-black-box construction. A relaxation of Theorem 2 that makes a non-black-box use of cryptography is a straightforward implication of [BCG+17b]. In more detail, [BCG+17b] obtained a zero-knowledge argument system for arithmetic circuit satisfiability over linear-size fields where the prover runs in linear time and the verifier runs in square-root time. The verifier time can be reduced to polylogarithmic, while preserving zero knowledge and a linear-time prover, by using any zero-knowledge succinct argument with subquadratic prover time to prove that the "outer" verifier would have accepted. A similar implication, however from the non-zero-knowledge succinct argument in [BCG20], is described in subsequent work [LSTW21; GLS+21], and thus we refer the reader to that work for more details on these non-black-box approaches. (We remark that [LSTW21; GLS+21] additionally contribute ideas and implementations to improve the concrete efficiency of argument systems with a linear-time prover and sublinear-time verifier.)

Black-box constructions from probabilistic proofs. Essentially all approaches for constructing succinct arguments follow this high-level template: first construct a probabilistic proof in some proof model, and then make a black-box use of 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 [BCC+16; BBB+18; WTS+18; XZZ+19; 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,[10] 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);[11] we refer the reader to follow up work [LSTW21; GLS+21] for a detailed discussion of these quasilinear costs in terms of computation size and the security parameter. In sum, the above line of works has contributed among the best asymptotic prover times for succinct arguments (as well as excellent concrete efficiency), but the cryptography has precluded linear-time provers.

Bootle et al. [BCG+17b] 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 2.7.) Prior work leveraged this observation to construct argument systems with linear-time prover and sublinear-time verifier, given a collision-resistant hash function as a black box.

- [BCG+17b] 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.

There are linear-time candidates for the hash function [AHI+17], leading to a linear-time prover.

In both cases the technical core is the construction of IOPs with a linear-time prover, but, as discussed in Section 1.2, these prior works only achieved sublinear query complexity thereby, after compilation, falling short of the goal of polylogarithmic communication complexity. No prior work thus achieves Theorem 2.

Our main result (Theorem 1) offers improved IOP constructions, and we are then able to improve the state of the art of succinct arguments that make a black-box use of cryptography (Theorem 2).

[10] The quasilinear costs in some works (due to cryptography [XZZ+19; 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.

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

2 Techniques

We overview our approach towards Theorem 1 in Section 2.1 and the construction in Section 2.2. We provide additional details behind different aspects of the construction in Sections 2.3 to 2.6. Finally, in Section 2.7 we explain how our result about zero-knowledge succinct arguments (Theorem 2) is a direct implication of our result about zero-knowledge IOPs (Theorem 1).

Throughout, 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 time and verifier time.

2.1 Approach overview

We provide an overview of our approach to Theorem 1.

Review: proof composition. Many constructions of PCPs rely on proof composition [AS98] to achieve the desired goal by combining an "outer" PCP and an "inner" PCP with suitable properties. The composed PCP (roughly) has the prover complexity of the outer PCP, and the verifier complexity of the inner PCP. Informally, the new PCP string consists of the outer PCP string and also, for every choice of randomness of the outer PCP verifier, an inner PCP string attesting that the outer PCP verifier would have accepted the local view of the outer PCP string induced by that choice of randomness. Soundness of the composed PCP requires the outer PCP to be *robust*[12] and the inner PCP to be a *proximity proof*.[13]

Proof composition extends to the IOP model [BCG+17a]: the outer and inner proof systems can be IOPs instead of PCPs, and must satisfy corresponding notions of robustness and proximity; moreover, composition is more efficient because the inner IOP has only to be invoked once rather than for every choice of randomness of the outer IOP verifier (this is because, after running the outer IOP, the verifier can simply send the chosen randomness to the prover and then run the inner IOP on that randomness). Proof composition of IOPs also plays a central role in constructions of IOPs, and we also use it in our construction, as described next.

Our setting. Using proof composition in our setting involves several considerations.

– *Zero knowledge.* We want the composed IOP to be semi-honest-verifier zero-knowledge, and for this, one can prove that it suffices for the outer IOP to be semi-honest-verifier

[12] A proof system is robust if the local view of the verifier is far (e.g. in Hamming distance) from an accepting view with high probability (over the verifier's randomness) whenever the instance is not in the language.

[13] A proximity proof shows that a given input is close to some input in the language.

zero-knowledge, regardless of any zero knowledge properties of the inner IOP. We prove this and other properties about zero knowledge within proof composition in the full version.

- *Prover time.* We want the prover of the composed IOP to run in linear time. The composed prover time is the sum of the outer IOP prover time and the inner IOP prover time. This means that the outer IOP prover must run in time that is linear, e.g., in the R1CS instance. The requirement on the inner IOP prover is less straightforward: the inner IOP prover attests to a computation related to the outer IOP verifier. For example, if the outer IOP verifier runs in cube-root time (relative to the R1CS instance) then we can afford an inner IOP prover that runs in up to cubic time (as the cubic blow up applied to a cube-root time gives linear time overall). In other words, we require the polynomial blow up of the inner IOP prover time to be made up by the savings offered by the outer IOP verifier time.
- *Verifier time.* We want the verifier of the composed IOP to run in polylogarithmic time. The composed verifier time equals the time of the inner IOP verifier when used to test that the outer IOP verifier would have accepted. At minimum, the inner IOP verifier needs to read the description of the outer IOP verifier computation, which consists of its input instance (e.g., the R1CS public input) and its randomness. This implies that the outer IOP verifier can have at most polylogarithmic randomness complexity, and also implies that the compound savings in running time of the outer IOP verifier and inner IOP verifier must lead to a polylogarithmic running time.

The above considerations suggest that one approach that suffices is the following: (i) an inner IOP of proximity for general computations with polylogarithmic verifier time; and (ii) an outer IOP for R1CS that is semi-honest-verifier zero-knowledge, is robust, has a linear prover time, has polylogarithmic randomness complexity, and has a verifier time that is sufficiently small so that we can afford the blowup incurred by the inner IOP prover time. For the inner IOP of proximity we choose the state-of-the-art PCP of proximity for $\mathsf{NTIME}(T)$ due to Mie [Mie09] (discussed later). Our technical contribution is constructing a suitable outer IOP. As the blowup incurred by the inner PCP prover time will be polynomial, we need the outer IOP verifier to run in time that is sufficiently sublinear. We now outline the challenges that arise given prior work.

Challenges. There are two natural paths to explore in order to construct the outer IOP.

1. One path would be to somehow construct the desired outer IOP by starting from the semi-honest-verifier zero-knowledge IOP for arithmetic circuit satisfiability in [BCG+17b], which works over any field of linear size and has linear prover time and square-root verifier time. This would require addressing some challenges. First, we would need to robustify the IOP, but robustification techniques typically work for verifiers with constant query complexity (possibly over a large alphabet), and so one would have to adapt [BCG+17b] for this setting. Second, the IOP verifier in [BCG+17b] would have to be derandomized to achieve polylogarithmic randomness complexity. Third, we cannot afford more than a quadratic blow up in the inner IOP prover time because the verifier in [BCG+17b] runs in square-root time.
2. An alternative path would be to somehow construct the desired outer IOP by starting from the IOP for R1CS in [BCG20], which over any field of size $O(m)$ has prover

time $O(m)$ and verifier time $O(m^\epsilon)$ for any a-priori fixed constant $\epsilon > 0$. This would require somehow additionally achieving zero knowledge (not a goal in [BCG20]), and moreover would still require addressing the robustification and derandomization challenges mentioned above. On the other hand, because we can choose ϵ to be small enough, we can afford an inner proximity proof whose prover runs in any fixed polynomial time (in particular, the PCP of proximity in [Mie09] would suffice).

This paper. We believe that both paths are plausible. In this paper we use an approach that (roughly) follows the second path, because we can use an off-the-shelf inner proximity proof and we can focus our attention solely on constructing an appropriate outer IOP. Moreover, we believe that building on [BCG20] will contribute new understanding of zero knowledge techniques that are likely to be useful elsewhere, and will lead to a simpler exposition due to the modular nature of that construction.

2.2 Construction overview

We outline the steps in the construction of an IOP that satisfies Theorem 1. We elaborate on each of these steps in subsequent subsections.

Review: the tensor-to-point approach. The IOP for R1CS in [BCG20] is obtained in two steps: first construct a *tensor IOP* for R1CS with linear prover time and constant query complexity; then 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 $q = (q_1, \ldots, q_t)$ and receiving the corresponding answer $v := \langle \otimes_i q_i, \Pi \rangle$. This differs from a standard IOP, where the verifier makes *point queries*, that is, it queries single locations of proof messages. As mentioned in Section 2.1, the resulting (point-query) IOP in [BCG20] has prover time $O(m)$ and verifier time $O(m^\epsilon)$ for any a-priori fixed constant $\epsilon > 0$. (Here m is the maximum number of non-zero entries in an R1CS coefficient matrix.)

Steps in our proof. We take an analogous two-step approach as in [BCG20], except that we additionally achieve semi-honest-verifier zero knowledge, while still achieving a prover time of $O(m)$ and reducing the verifier time from $O(m^\epsilon)$ to $\mathsf{poly}(|x|, \log m)$. (Here x is the instance vector of the R1CS instance.)

- *Step 1: tensor IOP for R1CS with zero knowledge.* Given any finite field $\mathbb{F}$, we construct a tensor IOP for R1CS over $\mathbb{F}$ that is semi-honest-verifier zero-knowledge, has soundness error $O(\frac{m}{|\mathbb{F}|})$, has prover time $O(m)$, and has verifier time $O(|x| + \log m)$; moreover, the verifier makes $O(1)$ tensor queries (and also interacts with the prover in a $O(\log m)$-round interactive proof). In Section 2.4 we outline the main ideas that we use to additionally achieve zero knowledge compared to the tensor IOP for R1CS in [BCG20].
- *Step 2: from tensor IOPs to standard IOPs while preserving zero knowledge.* Given any finite field $\mathbb{F}$, we construct a compiler that maps a tensor IOP over the field $\mathbb{F}$ into a standard IOP *while preserving the zero knowledge property*; moreover, efficiency measures are preserved up to overheads in the dimension of the tensor and the query complexity of the input tensor IOP. In Section 2.3 we outline the main ideas that we

use compared to the tensor-query to point-query compiler in [BCG20] (which does not preserve zero knowledge and leads to a large verifier time).

Theorem 1 follows by applying the compiler in the second step to the tensor IOP for R1CS in the first step, as shown diagrammatically in Figure 2. Below we highlight two aspects of our construction of the compiler.

(a) Proof composition. Differing from the approach overview in Section 2.1, the proof composition step actually happens within the tensor-query to point-query compiler rather than as a final step. This choice leads to a compiler that preserves efficiency measures of the tensor IOP up to constants (of independent interest), and moreover invokes the inner proximity proof on a linear computation rather than an arbitrary computation.

(b) Linear codes that are linear-time encodable and zero knowledge. A key ingredient in the construction of our compiler is *tensor codes that simultaneously are linear-time encodable and satisfy a zero-knowledge property* (informally, codewords do not reveal any information about the underlying message when queried in a restricted way). For this, we establish structural properties of zero-knowledge codes and prove that they are preserved under tensor products, which reduces the problem to constructing a linear-time encodable zero-knowledge code to act as the base of the tensor product code. We obtain a suitable base code via an explicit (deterministic) 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 1, which suffices for our main theorem. We also give a probabilistic construction of zero-knowledge codes based on [DI14] codes which 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 1 with the stronger property of *bounded-query zero-knowledge*. We review notions of zero knowledge for linear codes in Section 2.5, and then describe our results about zero-knowledge codes in Section 2.6.

Concrete efficiency. We do not make any claims regarding the concrete efficiency of our construction. That said, we are optimistic that the ideas introduced in this work can lead to improved constructions with the same asymptotic efficiency but better concrete efficiency. In particular, we believe that further research into zero-knowledge linear-time-encodable codes and further research in specializing the proof composition step to the specific outer statement (a certain linear computation) may significantly improve efficiency. Subsequent work has made progress in this direction [GLS+21].

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*. 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 only required to satisfy a weaker notion of zero knowledge, called *semi-honest*-verifier zero knowledge, that we describe further below. Here, "$(\mathbb{F}, k, t)$-tensor IOP" means that each tensor-IOP query $q = (q_1, \ldots, q_t)$ lies in $(\mathbb{F}^k)^t$.

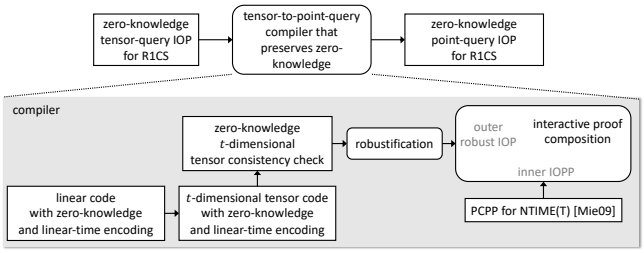

Fig. 2. Diagram of our construction of the IOP for Theorem 1.

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* $\Psi(k) \cdot k$*, and description size* $|\mathcal{C}|$*. (The description of a linear code consists of a specification of the circuit used to compute the encoding function, including any random coins used to generate the circuit.)*
- 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* $O_t(\mathsf{q})$*, prover arithmetic complexity* $\mathsf{tp} + O_{\rho,t}(\mathsf{q} \cdot \mathsf{l}) \cdot \Psi(k) + \mathsf{poly}(|\mathcal{C}|, t, \mathsf{q}, k)$*, 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 the following also holds:

- *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 Ψ *of* $\mathcal{C}$ *and terms that depend on* ρ *and* t*: if the indexer for the input tensor-query IOP runs in time* ti *and produces an index of length* li*, then the indexer for the output point-query IOP runs in time* $\mathsf{ti} + O_{\rho,t}(\mathsf{q} \cdot \mathsf{li}) \cdot \Psi(k)$*.*

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 $\Pi \in \mathbb{F}^{k^t}$ is replaced by its encoding $\hat{\Pi} \in \mathbb{F}^{n^t}$ using the

tensor product code $\mathcal{C}^{\otimes t}$; (ii) instead of making tensor queries to the proof oracles directly, the new verifier $\hat{\mathbf{V}}$ sends tensor queries $q^{(s)}$ to the prover, 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 of length k sampled by the verifier instead of structured tensor queries. In both cases, the verifier checks linear relations between successive encodings $c_j^{(s)}$ and $c_{j+1}^{(s)}$ by making $O(k)$ point queries.

This compiler preserves prover time up to the encoding overhead $\Psi(k)$ as in our Theorem 3, but has two 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 $\Omega(k)$ and verifier complexity $\Omega(k)$, which does not suffice for Theorem 3 (we can at most afford a polylogarithmic dependence in k). Below we elaborate on how we overcome these shortcomings for zero-knowledge (Section 2.3.1) and for efficiency (Section 2.3.2).

2.3.1 Preserving zero-knowledge

We explain semi-honest verifier zero knowledge (the property of the tensor IOP used to achieve zero knowledge for the output IOP) and then how we preserve zero knowledge in the compiler.

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.[14] 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 or against verifiers making a bounded number of point queries (depending on the zero-knowledge property of the code).

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 encoded proof message $\hat{\Pi}$ or folded proof message $c_j^{(s)}$ leaks information about Π. We address the information leaked by $\hat{\Pi}$ and $c_j^{(s)}$ in two ways.

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

We ensure that the folded proof messages $c_j^{(s)}$ do not leak any information by leveraging the fact that the consistency test 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* $c = \gamma\hat{\Pi} + \hat{\Xi}$ [BCF+17]. (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 test 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 test.

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-verifier zero knowledge for the output IOP, it is important that the tensor product code $\mathcal{C}^{\otimes t}$ 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.[15] This leads to the problem of finding a zero-knowledge code which is encodable in linear time, which we discuss in Section 2.6.3, and showing that the zero-knowledge property of codes is preserved under tensor products, which we discuss in Section 2.6.2.

2.3.2 Improving 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 the compiler preserves linear arithmetic complexity of the prover. However, when applied to our $(\mathbb{F}, k, t)$-tensor IOP with $n = \Theta(k^t)$, the improved consistency test has query complexity $O(k)$, prover arithmetic complexity $O(k^t)$, and verifier arithmetic complexity $O(k)$. Though the query complexity and verifier complexity can be improved by increasing t, they remain sublinear in n, which does not suffice for Theorem 3. To prove Theorem 3, we must reduce the query complexity from $\Omega(k)$ to $O(1)$ and the verifier complexity from $\Omega(k)$ to $\mathsf{poly}(\log k)$.

We achieve these goals using interactive proof composition and derandomization techniques. First, we strengthen the improved consistency test through robustification, and then use interactive proof composition for IOPs [BCG+17a]. This reduces the query complexity so that it is independent of k, and makes the verifier complexity depend only on the randomness complexity of the consistency test verifier. Next, we show that linear prover complexity is preserved. Finally, we explain how to derandomize the consistency test to obtain the desired verifier complexity. (It remains an interesting question whether

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

one can also achieve proof length that approaches witness length, the efficiency goal studied in [RR20] via related techniques.)

Interactive proof composition. Interactive proof composition involves an "outer" IOP that is robust and is for the desired relation, and an "inner" IOP of proximity that is for a relation about the outer IOP's verifier. At a high level, we wish to apply this with the zero-knowledge consistency test from Section 2.3.1 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 is not robust, and also because our target parameters do not leave much wiggle room. Below, we elaborate on how we robustify the outer protocol, and how we perform proof composition.

- *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).[16] 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).
  Superficially, this robustification seems insufficient to prove Theorem 3 because zero-knowledge consistency test from Section 2.3.1 has sublinear query complexity $\mathsf{q} = O(k) = O(n^{1/t})$, which would lead to a robustness parameter that is sub-constant. However, fortunately, the queries 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 linear-time encoding such as [Spi96] (here zero-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. We are now ready for the next step, proof composition.
- *Composition.* Interactive proof composition [BCG+17a] 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.
  We apply proof composition as follows: (i) the outer proof system is the robust zero-knowledge IOPP for tensor queries obtained above; (ii) the inner proof system is the PCP of proximity for $\mathsf{NTIME}(T)$ due to Mie [Mie09] (which achieves any constant soundness error and constant proximity 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

[16] An IOP is said to have robustness parameter α if the local view of the verifier is α-close (in relative Hamming distance) to an accepting view with probability bounded by the IOP's soundness error

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.

Preserving prover complexity. We discuss prover complexity for the composed proof system. The cost of the prover in the composed IOP 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 prover time in the PCPP of [Mie09], which is polynomial in $O(n^{1/t})$, is dominated by $O(n)$ field operations.

Reducing verifier complexity. We discuss verifier complexity for the composed proof system. The cost of the verifier in the composed IOP is dominated by $\mathsf{poly}(|\mathbb{x}|, \log T(|\mathbb{x}|))$ bit operations, the time to run the PCPP verifier in [Mie09]. From our discussion of prover complexity for the composed proof system, we know that $T = O(n^{1/t})$ and so $\log T(|\mathbb{x}|) = O(\log n)$. 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. The description consists of: (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 *verifier randomness* for the robust IOPP.

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. As for the last term, the randomness complexity of the robust consistency check is $O(\mathsf{q} \cdot k \cdot t)$, due to the random linear combinations used in the proximity test of [BCG20], whose randomness complexity is unchanged by robustification. In sum, the cost of the verifier in the composed system is $\mathsf{poly}(|\mathcal{C}|, \log n, \mathsf{q} \cdot k \cdot t)$ bit operations. This leads to a sublinear verifier complexity and does not suffice for Theorem 3.

Fortunately, these linear combinations can be derandomized so to reduce their description size to $O(t)$ (a low-order term), as we now explain. The linear combinations are used in the soundness analysis of the [BCG20] proximity test 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. Ben-Sasson et al. [BKS18] prove distortion statements for linear combinations of the form $\zeta = (\alpha^1, \alpha^2, \alpha^3, \ldots, \alpha^k)$ for a uniformly random $\alpha \in \mathbb{F}$, at the cost of a tolerable increase 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 $O(\mathsf{q} \cdot t)$. After some work, the result is a verifier complexity of $\mathsf{poly}(|\mathcal{C}|, \log n)$ bit operations in the composed system which suffices for Theorem 3.

Remark 1. We use the freedom to choose a large enough t in our robust zero-knowledge IOPP based on [BCG20] to obtain query complexity (and verifier time) that is $O(n^{1/t})$. It is plausible that [BCG+17b] similarly implies a robust zero-knowledge IOPP with

query complexity (and verifier time) $O(n^{1/2})$. We do not know how to leverage such a result because that would require an inner IOPP with subquadratic prover time and constant query complexity, and we do not know of such a result. While it is plausible that the prover of [Mie09] prover runs in subquadratic time, proving this seems an arduous task.

Remark 2 (zero knowledge). We do not require the the IOPP for tensor queries used in Section 2.3.1 to be zero knowledge because we can invoke it on a random instance as explained in Section 2.3.1. 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 the full version. We view these results as an independent contribution to foundational transformations of IOPs.

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.*

Our starting point is the holographic tensor IOP for R1CS in [BCG20], which achieves the same parameters as in the above theorem[17] except that it is not zero knowledge. We use re-randomization techniques to additionally 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 our ideas for how to additionally achieve zero knowledge.

[17] Note that as described in [BCG20], the tensor IOP of [BCG20] achieves verifier complexity $O(|x| + k)$ because some of the verifier's tensor queries are generated from seeds of length $O(k)$. We reduce the verifier complexity by generating the verifier's tensor queries using short seeds.

The holographic tensor IOP of BCG. The holographic tensor IOP for R1CS in [BCG20] follows a standard blueprint for constructing protocols for R1CS [BCR+19], adapted to the case of tensor queries. The 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). The verifier makes 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 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.[18] 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 [BCI+13]), they do not seem to be useful for our setting (tensor IOPs with linear-time proving). So our approach here will be to 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 the padded vectors would not satisfy the R1CS condition.

[18] 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.

This naive strategy, however, can be fixed as follows. 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.[19] 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.[20] In the rest of the protocol, the random solution acts as padding as described above, while preserving completeness. The choice of how much to pad depends on how many independent queries each oracle receives.

Though conceptually simple, this approach requires careful design and analysis. Intuitively, this is because the solutions to the R1CS gadget, which act as random padding, satisfy some non-linear relations, and therefore cannot consist entirely of uniformly random field elements. For example, since $z_C = z_A \circ z_B$, if the padding for z_A and z_B was uniformly random, then the padding for z_C would be a sum of products of uniformly random field elements, which would *not* lead to uniformly random answers to queries on z_C. However, introducing uniformly random padding into z_C requires setting some elements of z_A (or z_B) to fixed non-zero elements, which cannot then be used to make queries on z_A uniformly random. In sum, the solutions to the R1CS gadgets must hide not only the results to queries on the vectors z_A, z_B, z_C, but also the dependencies in the solutions themselves.

(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 $e = \gamma z + y$, where y 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.) This enables the simulator to randomly sample e and honestly run the lincheck protocol, which reveals no information, since the honest verifier only queries $e = \gamma z + y$, and never z and y separately. As 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 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 [BCF+17]. The simulator can randomly sample u and honestly run the sumcheck protocol, while simulating answers to q by querying p instead.

[19] This is distinct from how zero knowledge is achieved for prior IOPs for R1CS based on the Reed–Solomon code [BCR+19]. 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; BCI+13].

[20] 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.

We cannot apply this idea in our setting of linear-time provers without change. 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 $(\gamma f + r) \cdot (\gamma g + s)$. (If $p = f \cdot g$ sums to α then $(\gamma f + r) \cdot (\gamma g + s)$ 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 $\widetilde{\mathrm{Enc}} \colon \mathbb{F}^{k_m} \times \mathbb{F}^{k_r} \to \mathbb{F}^n$ defined as $\widetilde{\mathrm{Enc}}(m; r) := \mathrm{Enc}(m \| r)$. In turn, this function naturally induces a probabilistic encoding: we define $\widetilde{\mathrm{Enc}}(m)$ to be the random variable $\{\widetilde{\mathrm{Enc}}(m; r)\}_{r \leftarrow \mathbb{F}^{k_r}}$. 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 $\widetilde{\mathrm{Enc}}$ 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 $\mathrm{View}(\widetilde{\mathrm{Enc}}(m; r), A)$ the view of an oracle algorithm A that is given query access to the codeword $\widetilde{\mathrm{Enc}}(m; r)$. 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:

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

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; BCF+17]). 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).

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 $\{\widetilde{\mathrm{Enc}}(m;r)\}_{r\leftarrow\mathbb{F}^{k_r}}$ 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 the full version 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. The Reed–Solomon code also happens to be a versatile tool for constructing efficient IOPs, and indeed many IOPs rely on the Reed–Solomon code to additionally achieve bounded-query zero knowledge [BCGV16; BCF+17; BBHR19; AHIV17; BCR+19; 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 On bounded-query zero knowledge

Theorem 1 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 1 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 above 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 1 with *both* polylogarithmic verifier complexity and bounded-query zero-knowledge.

2.6.1 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 $\widetilde{\mathrm{Enc}}\colon \mathbb{F}^{k_m} \times \mathbb{F}^{k_r} \to \mathbb{F}^n$. We can split the generator matrix $G \in \mathbb{F}^{n \times k}$ associated with $\widetilde{\mathrm{Enc}}$ into two parts $G = \begin{bmatrix} G_m & G_r \end{bmatrix}$ so that $\widetilde{\mathrm{Enc}}(m; r) = G_m m + G_r r$ 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 1. *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 $\mathcal{D}(\mathcal{C})$.

The characterizations in Lemma 1 strengthen weaker statements in prior works such as [CDBN15; CCG+07; Wei16]. Below we provide intuition about the equivalences, first discussing bounded-query uniformity and then bounded-query zero knowledge. Technical details and a discussion of how the characterizations are related statements in prior works can be found in the full version.

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 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 $\mathcal{D}(\mathcal{C}) \setminus \mathcal{C}^\perp$.

2.6.2 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 2.6.3.)

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 the full version. 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 1, 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 1 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 1.

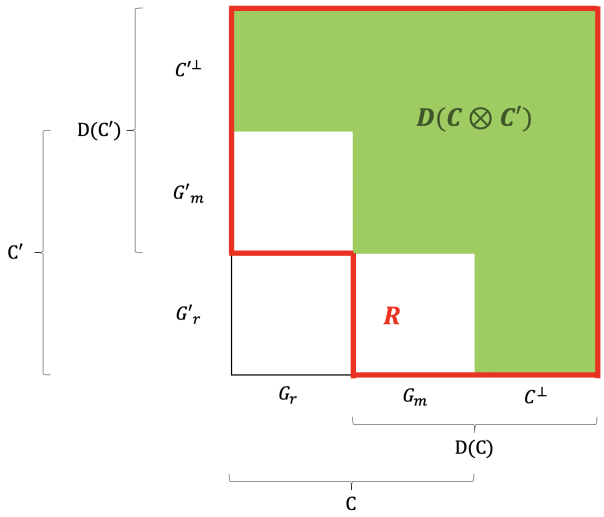

Fig. 3. The linear spaces $\mathcal{D}(\mathcal{C} \otimes \mathcal{C}')$ and R as subspaces of $\mathbb{F}^n \otimes \mathbb{F}^{n'}$.

2.6.3 Zero-knowledge codes with linear-time encoding

To prove our main theorem, Theorem 1, 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 [BCG+17b; 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 1, 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.

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.

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 $\{E_{k_m}\colon \mathbb{F}^{k_m} \times \mathbb{F}^{(H_q(\beta(k_m))+\epsilon)\cdot O(k_m)} \times \mathbb{F}^{O(k_m)} \to \mathbb{F}^{O(k_m)}\}_{k_m \in \mathbb{N}}$ 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 $\widetilde{\mathrm{Enc}}_{k_m}(m;r) := E_{k_m}(m,r,R)$ 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 the full version.

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 1 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, $G_R x$ 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 1, 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. [CCG+07] 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 [CCG+07, 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.

2.7 Linear-time succinct arguments from linear-time IOPs

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. [BCG+17b] observe that Kilian's approach [Kil92] is a notable exception. We review this below because Theorem 1 implies Theorem 2 via this approach; our technical contribution is Theorem 1.

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 [AHI+17]). 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.[21] While constructions of linear-time PCPs are not known (and seem far beyond current techniques), the foregoing implication equally holds for IOPs [BCS16; RRR16]. This route was used in [BCG+17b; BCG20] to obtain interactive arguments with linear-time prover and sublinear-time verifier from IOPs with linear-time prover and sublinear-time verifier.

Zero knowledge. 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.[22] Ishai et al. [IMSX15] then proved that if the underlying probabilistic proof satisfies a mild notion of zero knowledge then Kilian's approach can be significantly simplified to yield a zero-knowledge succinct argument where *the 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 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.

In sum. The interactive argument in Theorem 2 is constructed via the above approach from the IOP in Theorem 1 (with sublinear verification in the holographic setting

[21] 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.

[22] Modify the Merkle tree to be over hiding commitments to 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.

mapping to sublinear verification in the preprocessing setting). The semi-honest-verifier zero-knowledge property of the IOP implies the malicious-verifier zero-knowledge property of the interactive argument. The linear prover time and polylogarithmic verifier time of the IOP imply the corresponding running times of the interactive argument, as the given collision-resistant hash function runs in linear time.

References

[AHI+17] B. Applebaum, N. Haramaty, Y. Ishai, E. Kushilevitz, and V. Vaikuntanathan. "Low-Complexity Cryptographic Hash Functions". In: ITCS '17.

[AHIV17] S. Ames, C. Hazay, Y. Ishai, and M. Venkitasubramaniam. "Ligero: Lightweight Sublinear Arguments Without a Trusted Setup". In: CCS '17.

[AS98] S. Arora and S. Safra. "Probabilistic checking of proofs: a new characterization of NP". In: *Journal of the ACM* (1998). Preliminary version in FOCS '92.

[BBB+18] B. Bünz, J. Bootle, D. Boneh, A. Poelstra, P. Wuille, and G. Maxwell. "Bulletproofs: Short Proofs for Confidential Transactions and More". In: S&P '18.

[BBHR19] E. Ben-Sasson, I. Bentov, Y. Horesh, and M. Riabzev. "Scalable Zero Knowledge with No Trusted Setup". In: CRYPTO '19.

[BCC+16] J. Bootle, A. Cerulli, P. Chaidos, J. Groth, and C. Petit. "Efficient Zero-Knowledge Arguments for Arithmetic Circuits in the Discrete Log Setting". In: EUROCRYPT '16.

[BCC88] G. Brassard, D. Chaum, and C. Crépeau. "Minimum disclosure proofs of knowledge". In: *Journal of Computer and System Sciences* (1988).

[BCF+17] E. Ben-Sasson, A. Chiesa, M. A. Forbes, A. Gabizon, M. Riabzev, and N. Spooner. "Zero Knowledge Protocols from Succinct Constraint Detection". In: TCC '17.

[BCG+17a] E. Ben-Sasson, A. Chiesa, A. Gabizon, M. Riabzev, and N. Spooner. "Interactive Oracle Proofs with Constant Rate and Query Complexity". In: ICALP '17.

[BCG+17b] J. Bootle, A. Cerulli, E. Ghadafi, J. Groth, M. Hajiabadi, and S. K. Jakobsen. "Linear-Time Zero-Knowledge Proofs for Arithmetic Circuit Satisfiability". In: ASIACRYPT '17.

[BCG+19] E. Ben-Sasson, A. Chiesa, L. Goldberg, T. Gur, M. Riabzev, and N. Spooner. "Linear-Size Constant-Query IOPs for Delegating Computation". In: TCC '19.

[BCG20] J. Bootle, A. Chiesa, and J. Groth. "Linear-Time Arguments with Sublinear Verification from Tensor Codes". In: TCC '20.

[BCGV16] E. Ben-Sasson, A. Chiesa, A. Gabizon, and M. Virza. "Quasilinear-Size Zero Knowledge from Linear-Algebraic PCPs". In: TCC '16-A.

[BCI+13] N. Bitansky, A. Chiesa, Y. Ishai, R. Ostrovsky, and O. Paneth. "Succinct Non-Interactive Arguments via Linear Interactive Proofs". In: TCC '13.

[BCR+19] E. Ben-Sasson, A. Chiesa, M. Riabzev, N. Spooner, M. Virza, and N. P. Ward. "Aurora: Transparent Succinct Arguments for R1CS". In: EUROCRYPT '19.

[BCS16] E. Ben-Sasson, A. Chiesa, and N. Spooner. "Interactive Oracle Proofs". In: TCC '16-B.

[BKK+13] E. Ben-Sasson, Y. Kaplan, S. Kopparty, O. Meir, and H. Stichtenoth. "Constant Rate PCPs for Circuit-SAT with Sublinear Query Complexity". In: FOCS '13.

[BKS18] E. Ben-Sasson, S. Kopparty, and S. Saraf. "Worst-Case to Average Case Reductions for the Distance to a Code". In: CCS '18.

[CCG+07] H. Chen, R. Cramer, S. Goldwasser, R. de Haan, and V. Vaikuntanathan. "Secure Computation from Random Error Correcting Codes". In: EUROCRYPT' 07.

[CDBN15] R. Cramer, I. Damgård, and J. Buus Nielsen. *Secure Multiparty Computation and Secret Sharing*. Cambridge University Press, 2015.

[CDG+17] M. Chase, D. Derler, S. Goldfeder, C. Orlandi, S. Ramacher, C. Rechberger, D. Slamanig, and G. Zaverucha. "Post-Quantum Zero-Knowledge and Signatures from Symmetric-Key Primitives". In: CCS '17.

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

[CHM+20] A. Chiesa, Y. Hu, M. Maller, P. Mishra, N. Vesely, and N. Ward. "Marlin: Preprocessing zkSNARKs with Universal and Updatable SRS". In: EUROCRYPT '20.

[COS20] A. Chiesa, D. Ojha, and N. Spooner. "Fractal: Post-Quantum and Transparent Recursive Proofs from Holography". In: EUROCRYPT '20.

[DI14] E. Druk and Y. Ishai. "Linear-time encodable codes meeting the Gilbert–Varshamov bound and their cryptographic applications". In: ITCS '14.

[GGPR13] R. Gennaro, C. Gentry, B. Parno, and M. Raykova. "Quadratic Span Programs and Succinct NIZKs without PCPs". In: EUROCRYPT '13.

[GH98] O. Goldreich and J. Håstad. "On the complexity of interactive proofs with bounded communication". In: *Information Processing Letters* (1998).

[GIMS10] V. Goyal, Y. Ishai, M. Mahmoody, and A. Sahai. "Interactive locking, zero-knowledge PCPs, and unconditional cryptography". In: CRYPTO'10.

[GK96] O. Goldreich and A. Kahan. "How to Construct Constant-Round Zero-Knowledge Proof Systems for NP". In: *Journal of Cryptology* (1996).

[GLS+21] A. Golovnev, J. Lee, S. Setty, J. Thaler, and R. Wahby. "Brakedown: Linear-time and post-quantum SNARKs for R1CS". Cryptology ePrint Archive, Report 2021/1043.

[GMO16] I. Giacomelli, J. Madsen, and C. Orlandi. "ZKBoo: Faster Zero-Knowledge for Boolean Circuits". In: Security '16.

[GMR89] S. Goldwasser, S. Micali, and C. Rackoff. "The knowledge complexity of interactive proof systems". In: *SIAM Journal on Computing* (1989). Preliminary version appeared in STOC '85.

[GVW02] O. Goldreich, S. Vadhan, and A. Wigderson. "On interactive proofs with a laconic prover". In: *Computational Complexity* (2002).

[HK20] D. Heath and V. Kolesnikov. "Stacked Garbling for Disjunctive Zero-Knowledge Proofs". In: EUROCRYPT '20.

[IKOS07] Y. Ishai, E. Kushilevitz, R. Ostrovsky, and A. Sahai. "Zero-knowledge from secure multiparty computation". In: STOC'07.

[IMS12] Y. Ishai, M. Mahmoody, and A. Sahai. "On Efficient Zero-Knowledge PCPs". In: TCC '12.

[IMSX15] Y. Ishai, M. Mahmoody, A. Sahai, and D. Xiao. "On Zero-Knowledge PCPs: Limitations, Simplifications, and Applications". Available at `http://www.cs.virginia.edu/~mohammad/files/papers/ZKPCPs-Full.pdf`.

[ISVW13] Y. Ishai, A. Sahai, M. Viderman, and M. Weiss. "Zero Knowledge LTCs and Their Applications". In: APPROX-RANDOM '13.

[Kil92] J. Kilian. "A note on efficient zero-knowledge proofs and arguments". In: STOC '92.

[KKW18] J. Katz, V. Kolesnikov, and X. Wang. "Improved Non-Interactive Zero Knowledge with Applications to Post-Quantum Signatures". In: CCS '18.

[KMP20] A. Kothapalli, E. Masserova, and B. Parno. "A Direct Construction for Asymptotically Optimal zkSNARKs". Cryptology ePrint Archive, Report 2020/1318.

[KPT97] J. Kilian, E. Petrank, and G. Tardos. "Probabilistically checkable proofs with zero knowledge". In: STOC '97.

[LSTW21] J. Lee, S. Setty, J. Thaler, and R. Wahby. "Linear-time zero-knowledge SNARKs for R1CS". Cryptology ePrint Archive, Report 2021/030.

[Mei12] O. Meir. "Combinatorial PCPs with Short Proofs". In: CCC '12.

[Mei13] O. Meir. “IP = PSPACE Using Error-Correcting Codes”. In: *SIAM Journal on Computing* (2013).

[Mie09] T. Mie. “Short PCPPs verifiable in polylogarithmic time with O(1) queries”. In: *Annals of Mathematics and Artificial Intelligence* (3 2009).

[RR20] N. Ron-Zewi and R. Rothblum. “Local Proofs Approaching the Witness Length”. In: FOCS ’20.

[RRR16] O. Reingold, R. Rothblum, and G. Rothblum. “Constant-Round Interactive Proofs for Delegating Computation”. In: STOC ’16.

[Set20] S. Setty. “Spartan: Efficient and general-purpose zkSNARKs without trusted setup”. In: CRYPTO ’20.

[SL20] S. Setty and J. Lee. “Quarks: Quadruple-efficient transparent zkSNARKs”. Cryptology ePrint Archive, Report 2020/1275.

[Spi96] D. A. Spielman. “Linear-time encodable and decodable error-correcting codes”. In: *IEEE Transactions on Information Theory* (1996). Preliminary version appeared in STOC ’95.

[Tha13] J. Thaler. “Time-Optimal Interactive Proofs for Circuit Evaluation”. In: CRYPTO ’13.

[Wei16] M. Weiss. “Secure Computation and Probabilistic Checking”. 2016.

[WTS+18] R. S. Wahby, I. Tzialla, A. Shelat, J. Thaler, and M. Walfish. “Doubly-efficient zkSNARKs without trusted setup”. In: S&P ’18.

[WYKW20] C. Weng, K. Yang, J. Katz, and X. Wang. “Wolverine: Fast, Scalable, and Communication-Efficient Zero-Knowledge Proofs for Boolean and Arithmetic Circuits”. IACR Cryptology ePrint Archive, Report 2020/925.

[XZZ+19] T. Xie, J. Zhang, Y. Zhang, C. Papamanthou, and D. Song. “Libra: Succinct Zero-Knowledge Proofs with Optimal Prover Computation”. In: CRYPTO ’19.

[ZWZZ20] J. Zhang, W. Wang, Y. Zhang, and Y. Zhang. “Doubly Efficient Interactive Proofs for General Arithmetic Circuits with Linear Prover Time”. Cryptology ePrint Archive, Report 2020/1247.

[ZXZS20] J. Zhang, T. Xie, Y. Zhang, and D. Song. “Transparent Polynomial Delegation and Its Applications to Zero Knowledge Proof”. In: S&P ’20.