Aurora: Transparent Succinct Arguments for R1CS

Eli Ben-Sasson[1], Alessandro Chiesa[2], Michael Riabzev[1], Nicholas Spooner[2], Madars Virza[3] and Nicholas P. Ward[2]

[1] Technion/STARKWare eli@cs.technion.ac.il
mriabzev@cs.technion.ac.il
[2] UC Berkeley alexch@berkeley.edu nick.spooner@berkeley.edu
npward@berkeley.edu
[3] MIT Media Lab madars@mit.edu

Abstract. We design, implement, and evaluate a zero knowledge succinct non-interactive argument (SNARG) for Rank-1 Constraint Satisfaction (R1CS), a widely-deployed NP language undergoing standardization. Our SNARG has a transparent setup, is plausibly post-quantum secure, and uses lightweight cryptography. A proof attesting to the satisfiability of n constraints has size $O(\log^2 n)$; it can be produced with $O(n \log n)$ field operations and verified with $O(n)$. At 128 bits of security, proofs are less than $250\,\mathrm{kB}$ even for several million constraints, more than $10\times$ shorter than prior SNARGs with similar features.

A key ingredient of our construction is a new Interactive Oracle Proof (IOP) for solving a *univariate* analogue of the classical sumcheck problem [LFKN92], originally studied for *multivariate* polynomials. Our protocol verifies the sum of entries of a Reed–Solomon codeword over any subgroup of a field.

We also provide `libiop`, a library for writing IOP-based arguments, in which a toolchain of transformations enables programmers to write new arguments by writing simple IOP sub-components. We have used this library to specify our construction and prior ones, and plan to open-source it.

Keywords: zero knowledge; interactive oracle proofs; succinct arguments; sumcheck protocol

Table of Contents

1 Introduction

A zero knowledge proof is a protocol that enables one party (the *prover*) to convince another (the *verifier*) that a statement is true without revealing any information beyond the fact that the statement is true. Since their introduction [49], zero knowledge proofs have become fundamental tools not only in the theory of cryptography but also, more recently, in the design of real-world systems with strong privacy properties.

For example, zero knowledge proofs are the core technology in Zcash [18, 1], a popular cryptocurrency that preserves a user's payment privacy. While in Bitcoin [65] users broadcast their private payment details in the clear on the public blockchain (so other participants can check the validity of the payment), users in Zcash broadcast *encrypted* transaction details and *prove*, in zero knowledge, the validity of the payments without disclosing what the payments are.

Many applications, including the aforementioned, require that proofs are *succinct*, namely, that proofs scale *sublinearly* in the size of the witness for the statement, or perhaps even in the size of the computation performed to check the statement. This strong efficiency requirement cannot be achieved with statistical soundness (under standard complexity assumptions) [47], and thus one must consider proof systems that are merely computationally sound, known as *argument systems* [34]. Many applications further require that a proof consists of a single non-interactive message that can be verified by anyone; such proofs are cheap to communicate and can be stored for later use (e.g., on a public ledger). Constructions that satisfy these properties are known as (publicly verifiable) *succinct non-interactive arguments* (SNARGs) [46].

In this work we present Aurora, a zero knowledge SNARG for (an extension of) arithmetic circuit satisfiability whose argument size is polylogarithmic in the circuit size. Aurora also has attractive features: it uses a transparent setup, is plausibly post-quantum secure, and only makes black-box use of fast symmetric cryptography (any cryptographic hash function modeled as a random oracle).

Our work makes an exponential asymptotic improvement in argument size over Ligero [4], a recent zero knowledge SNARG with similar features but where proofs scale as the *square root* of the circuit size. For example, Aurora's proofs are $20\times$ smaller than Ligero's for circuits with a million gates (which already suffices for representative applications such as Zcash).

Our work also complements and improves on Stark [13], a recent zero knowledge SNARG that targets computations expressed as bounded halting problems on random access machines. While Stark was designed for a different computation model, we can still study its efficiency when applied to arithmetic circuits. In this case Aurora's prover is faster by a logarithmic factor (in the circuit size) and Aurora's proofs are concretely much shorter, e.g., $15\times$ smaller for circuits with a million gates.

The efficiency features of Aurora stem from a new Interactive Oracle Proof (IOP) that solves a *univariate* analogue of the celebrated sumcheck problem [61], in which query complexity is *logarithmic* in the degree of the polynomial being summed. This is an *exponential* improvement over the original multi-variate protocol, where communication complexity is (at least) *linear* in the degree of the polynomial. We believe this protocol and its analysis are of independent interest.

1.1 The need for a transparent setup

The first succinct argument is due to Kilian [57], who showed how to use collision-resistant hashing to compile any Probabilistically Checkable Proof (PCP) [9, 43, 6, 5] into a corresponding interactive argument. Micali then showed how a similar construction, in the random oracle model, yields succinct *non*-interactive arguments (SNARGs) [63]. Subsequent work [55] noted that if the underlying PCP is zero knowledge then so is the SNARG. Unfortunately, PCPs remain very expensive, and this approach has not led to SNARGs with good concrete efficiency.

In light of this, a different approach was initially used to achieve SNARG implementations with good concrete efficiency [67, 19]. This approach, pioneered in [50, 45, 60, 29], relied on combining certain linearly homomorphic encodings with lightweight information-theoretic tools known as linear PCPs [54, 29, 71]; this approach was refined and optimized in several works [23, 22, 40, 51, 30, 52]. These constructions underlie widely-used open-source libraries [70] and deployed systems [1], and their main feature is that proofs are very short (a few hundred bytes) and very cheap to verify (a few milliseconds).

Unfortunately, the foregoing approach suffers from a severe limitation, namely, the need for a central party to generate system parameters for the argument system. Essentially, this party must run a probabilistic algorithm, publish its output, and "forget" the secret randomness used to generate it. This party must be trustworthy because knowing these secrets allows forging proofs for false assertions. While this may sound like an inconvenience, it is a *colossal* challenge to real-world deployments. When using cryptographic proofs in distributed systems, relying on a central party negates the benefits of distributed trust and, even though it is invoked only once in a system's life, a party trusted by all users typically *does not exist*!

The responsibility for generating parameters can in principle be shared across multiple parties via techniques that leverage secure multi-party computation [20, 32, 33]. This was the approach taken for the launch of Zcash [2], but it also demonstrated how unwieldy such an approach is, involving a costly and logistically difficult real-world multi-party "ceremony". Successfully running such a multi-party protocol was a singular feat, and systems without such expensive setup are decidedly preferable.

Some setup is unavoidable because if SNARGs without *any* setup existed then so would sub-exponential algorithms for SAT [81]. Nevertheless, one could still aim for a "transparent setup", namely one that consists of *public randomness*, because in practice it is cheaper to realize. Recent efforts have thus focused on designing SNARGs with transparent setup (see discussion in Section 1.4).

1.2 Our goal

The goal of this paper is to obtain *transparent SNARGs* that satisfy the following desiderata.

– *Post-quantum security.* Practitioners, and even standards bodies [66], have a strong interest in cryptographic primitives that are plausibly secure against efficient quantum adversaries. This is motivated by the desire to ensure long-term security of deployed systems and protocols.

– *Concrete efficiency.* We seek argument systems that not only exhibit good asymptotics (in argument size and prover/verifier time) but also demonstrably offer good efficiency via a prototype.

The second bullet warrants additional context. Most argument systems support an NP-complete problem, so they are in principle equivalent under polynomial-time reductions. Yet, whether such protocols can be efficiently used in practice actually depends on: (a) the particular NP-complete problem "supported" by the protocol; (b) the concrete efficiency of the protocol relative to this problem. This creates a complex tradeoff.

Simple NP-complete problems, like boolean circuit satisfaction, facilitate simple argument systems; but reducing the statements we wish to prove to boolean circuits is often expensive. On the other hand, one can design argument systems for rich problems (e.g., an abstract computer) for which it is cheap to express the desired statements; but such argument systems may use expensive tools to support these rich problems.

Our goal is concretely-efficient argument systems for *rank-1 constraint satisfaction* (R1CS), which is the following natural NP-complete problem: given a vector $v \in \mathbb{F}^k$ and three matrices $A, B, C \in \mathbb{F}^{m \times n}$, can one augment v to $z \in \mathbb{F}^n$ such that $Az \circ Bz = Cz$? (We use "$\circ$" to denote the entry-wise product.)

We choose R1CS because it strikes an attractive balance: it generalizes circuits by allowing "native" field arithmetic and having no fan-in/fan-out restrictions, but it is simple enough that one can design efficient argument systems for it. Moreover, R1CS has demonstrated *strong empirical value*: it underlies real-world systems [1] and there are compilers that reduce program executions to it (see [80] and references therein). This has led to efforts to standardize R1CS formats across academia and industry [3].

1.3 Our contributions

In this work we study *Interactive Oracle Proofs* (IOPs) [21, 69], a notion of "multi-round PCPs" that has recently received much attention [17, 15, 12, 14, 13, 25]. These types of interactive proofs can be compiled into non-interactive arguments in the random oracle model [21], and in particular can be used to construct transparent SNARGs. Building on this approach, we present several contributions: (1) an IOP protocol for R1CS with attractive efficiency features; (2) the design, implementation, and evaluation of a transparent SNARG for R1CS, based on our IOP protocol; (3) a generic library for writing IOP-based non-interactive arguments. We now describe each contribution.

(1) IOP for R1CS. We construct a zero knowledge IOP protocol for rank-1 constraint satisfaction (R1CS) with *linear* proof length and *logarithmic* query complexity.

Given an R1CS instance $\mathcal{C} = (A, B, C)$ with $A, B, C \in \mathbb{F}^{m \times n}$, we denote by $N = \Omega(m + n)$ the total number of non-zero entries in the three matrices and by $|\mathcal{C}|$ the number of bits required to represent these; note that $|\mathcal{C}| = \Theta(N \log |\mathbb{F}|)$. One can view N as the number of "arithmetic gates" in the R1CS instance.

Theorem 1 (informal). *There is an $O(\log N)$-round IOP protocol for R1CS with proof length $O(N)$ over alphabet $\mathbb{F}$ and query complexity $O(\log N)$. The prover uses $O(N \log N)$ field operations, while the verifier uses $O(N)$ field operations. The IOP protocol is public coin and is a zero knowledge proof.*

The core of our result is a solution to a *univariate* analogue of the classical sumcheck problem [61]. Our protocol (including zero knowledge and soundness error reduction) is relatively simple: it is specified in a single page (see Fig. 12 in Section 6), given a low-degree test as a subroutine. The low degree test that we use is a recent highly-efficient IOP for testing proximity to the Reed–Solomon code [14].

(2) SNARG for R1CS. We design, implement, and evaluate **Aurora**, a zero knowledge SNARG of knowledge (zkSNARK) for R1CS with several notable features: (a) it only makes black-box use of fast symmetric cryptography (any cryptographic hash function modeled as a random oracle); (b) it has a transparent setup (users merely need to "agree" on which cryptographic hash function to use); (c) it is plausibly post-quantum secure (there are no known efficient quantum attacks against this construction). These features follow from the fact that Aurora is obtained by applying the transformation of [21] to our IOP for R1CS. This transformation preserves both zero knowledge and proof of knowledge of the underlying IOP.

In terms of asymptotics, given an R1CS instance $\mathcal{C}$ over $\mathbb{F}$ with N gates (and here taking for simplicity $\mathbb{F}$ to have size $2^{O(\lambda)}$ where λ is the security parameter), Aurora provides proofs of length $O_\lambda(\log^2 N)$; these can be produced in time $O_\lambda(N \log N)$ and checked in time $O_\lambda(N)$.

For example, setting our implementation to a security level of 128 bits over a 192-bit finite field, proofs range from $50\,\mathrm{kB}$ to $250\,\mathrm{kB}$ for instances of up to millions of gates; producing proofs takes on the order of several minutes and checking proofs on the order of several seconds. (See Section 4 for details.)

Overall, as indicated in Fig. 2, we achieve the smallest argument size among (plausibly) post-quantum non-interactive arguments for circuits, *by more than an order of magnitude*. Other approaches achieve smaller argument sizes by relying on (public-key) cryptography that is insecure against quantum adversaries.

(3) `libiop`: a library for non-interactive arguments. We provide `libiop`, a codebase that enables the design and implementation of non-interactive arguments based on IOPs. The codebase uses the C++ language and has three main components: (1) a library for writing IOP protocols; (2) a realization of [21]'s transformation, mapping any IOP written with our library to a corresponding non-interactive argument; (3) a portfolio of IOP protocols, including Ligero [4], Stark [13], and ours.

We plan to open-source `libiop` under a permissive software license for the community, so that others may benefit from its portfolio of IOP-based arguments, and may even write new IOPs tailored to new applications. We believe that our library will serve as a powerful tool in meeting the increasing demand by practitioners for transparent non-interactive arguments.

1.4 Prior implementations of transparent SNARGs

We summarize prior work that has designed *and implemented* transparent SNARGs; see Fig. 2.[4]

[4] We omit a discussion of prior works without implementations, or that study non-transparent SNARGs; we refer the reader to the survey of Walfish and Blumberg [80] for an overview of

Based on asymmetric cryptography. *Bulletproofs* [31, 35] proves the satisfaction of an N-gate arithmetic circuit via a recursive use of a low-communication protocol for inner products, achieving a proof with $O(\log N)$ group elements. *Hyrax* [79] proves the satisfaction of a *layered* arithmetic circuit of depth D and width W via proofs of $O(D \log W)$ group elements; the construction applies the Cramer–Damgård transformation [41] to doubly-efficient Interactive Proofs [48, 39]. Both approaches use Pedersen commitments, and so are *vulnerable to quantum attacks*. Also, in both approaches the verifier performs *many expensive cryptographic operations*: in the former, the verifier uses $O(N)$ group exponentiations; in the latter, the verifier's group exponentiations are linear in the circuit's witness size. (Hyrax allows fewer group exponentiations but with longer proofs; see [79].)

Based on symmetric cryptography. The "original" SNARG construction of Micali [63, 55] has advantages beyond transparency. First, it is unconditionally secure given a random oracle, which can be instantiated with extremely fast symmetric cryptography.[5] Second, it is plausibly post-quantum secure, in that there are no known efficient quantum attacks. But the construction relies on PCPs, which remain expensive.

IOPs are "multi-round PCPs" that can also be compiled into non-interactive arguments in the random oracle model [21]. This compilation retains the foregoing advantages (transparency, lightweight cryptography, and plausible post-quantum security) and, *in addition*, facilitates greater efficiency, as IOPs have superior efficiency compared to PCPs [17, 15, 12, 14, 13].

In this work we follow the above approach, by constructing a SNARG based on a new IOP protocol. Two recent works have also taken the same approach, but with different underlying IOP protocols, which have led to different features. We provide both of these works as part of our library (Section 5), and experimentally compare them with our protocol (Section 4). The discussion below is a qualitative comparison.

- **Ligero** [4] is a non-interactive argument that proves the satisfiability of an N-gate circuit via proofs of size $O(\sqrt{N})$ that can be verified in $O(N)$ cryptographic operations. As summarized in Fig. 1, the IOP underlying Ligero achieves the same oracle proof length, prover time, and verifier time as our IOP. However, we reduce query complexity from $O(\sqrt{N})$ to $O(\log N)$, which is an exponential improvement, at the expense of increasing round complexity from 2 to $O(\log N)$. The arguments that we obtain are still non-interactive, but our smaller query complexity translates into shorter proofs (see Fig. 2).
- **Stark** [13] is a non-interactive argument for bounded halting problems on a random access machine. Given a program P and a time bound T, it proves that P accepts within T steps on a certain abstract computer (when given suitable nondeterministic advice) via succinct proofs of size $\mathrm{polylog}(T)$. Moreover, verification is *also* succinct:

---

sublinear argument systems. We also note that recent work [11] has used lattice cryptography to achieve sublinear zero knowledge arguments that are plausibly post-quantum secure, which raises the exciting question of whether these recent protocols can lead to efficient implementations.

[5] Some cryptographic hash functions, such as BLAKE2, can process almost 1 gibibyte per second [8].

checking a proof takes time only $|P| + \text{polylog}(T)$, which is polynomial in the size of the statement and much better than "naive verification" which takes time $\Omega(|P| + T)$. The main difference between Stark and Aurora is the computational models that they support. While Stark supports *uniform* computations specified by a program and a time bound, Aurora supports *non-uniform* computations specified by an explicit circuit (or constraint system). Despite this difference, we can compare the cost of Stark and Aurora with respect to the explicit circuit model, since one can reduce a given N-gate circuit (or N-constraint system) to a corresponding bounded halting problem with $|P|, T = \Theta(N)$.

In this case, Stark's verification time is the same as Aurora's, $O(N)$; this is best possible because just *reading* an N-gate circuit takes time $\Omega(N)$. But Stark's prover is a logarithmic factor more expensive because it uses a switching network to verify a program's accesses to memory. Stark's prover uses an IOP with oracles of size $O(N \log N)$, leading to an arithmetic complexity of $O(N \log^2 N)$. (See Figs. 1 and 2.) Both Stark and Aurora have argument size $O(\log^2 N)$, but additional costs in Stark (e.g., due to switching networks) result in Stark proofs being *one order of magnitude larger* than Aurora proofs. That said, we view Stark and Aurora as complementing each other: Stark offers savings in verification time for succinctly represented programs, while Aurora offers savings in argument size for explicitly represented circuits.

2 Techniques

Our main technical contribution is a linear-length logarithmic-query IOP for R1CS (Theorem 1), which we use to design, implement, and evaluate a transparent SNARG for R1CS. Below we summarize the main ideas behind our protocol, and postpone to Sections 4 and 5 discussions of our system. In Section 2.1, we describe our approach to obtain the IOP for R1CS; this approach leads us to solve the univariate sumcheck problem, as discussed in Section 2.2; finally, in Section 2.3, we explain how we achieve zero knowledge. In Section 2.4 we conclude with a wider perspective on the techniques used in this paper.

2.1 Our interactive oracle proof for R1CS

The R1CS relation consists of instance-witness pairs $((A, B, C, v), w)$, where A, B, C are matrices and v, w are vectors over a finite field $\mathbb{F}$, such that $(Az) \circ (Bz) = Cz$ for $z := (1, v, w)$ and "$\circ$" denotes the entry-wise product.[6] For example, R1CS captures arithmetic circuit satisfaction: A, B, C represent the circuit's gates, v the circuit's public input, and w the circuit's private input and wire values.[7]

[6] Throughout, we assume that $\mathbb{F}$ is "friendly" to FFT algorithms, i.e., $\mathbb{F}$ is a binary field or its multiplicative group is smooth.

[7] The reader may be familiar with a standard arithmetization of circuit satisfaction (used, e.g., in the inner PCP of [5]). Given an arithmetic circuit with m gates and n wires, each addition gate $x_i \leftarrow x_j + x_k$ is mapped to the linear constraint $x_i = x_j + x_k$ and each product gate

	protocol type	**round complexity**	**proof length** (field elts)	**query complexity**	**prover time** (field ops)	**verifier time** (field ops)
Ligero	IPCP †	2	$O(N)$	$O(\sqrt{N})$	$O(N \log N)$	$O(N)$
Stark	IOP	$O(\log N)$	$O(N \log N)$	$O(\log N)$	$O(N \log^2 N)$	$O(N)$
Aurora	IOP	$O(\log N)$	$O(N)$	$O(\log N)$	$O(N \log N)$	$O(N)$

Fig. 1: Asymptotic comparison of the information-theoretic proof systems underlying Ligero, Stark, and Aurora, when applied to an N-gate arithmetic circuit.
† An IPCP [56] is a PCP oracle that is checked via an Interactive Proof; it is a special case of an IOP.

	name	**setup**	**post quantum?**	**argument size** asymptotic	$N = 10^6$	**verifier time**	**non-interactivity technology**
[50][45] [60][29]...	various	private	no	$O_\lambda(1)$	128 B	$O_\lambda(k)$ †	linear PCP + linear encoding
[83]	ZK-vSQL	private	no	$O_\lambda(d \log N)$	N/A	$O_\lambda(N)$	apply [41]-transform to doubly-efficient IP [48, 39]
[79]	Hyrax	public	no	$O_\lambda(d \log N)$ ‡	50 kB	$O_\lambda(N)$	as above (but using a different polynomial commitment)
[31] [35]	Bulletproofs	public	no	$O_\lambda(\log N)$	1.5 kB	$O_\lambda(N)$	recursive inner product argument
[4]	Ligero	public	yes	$O_\lambda(\sqrt{N})$	4.0 MB	$O_\lambda(N)$	apply [21]-transform to IPCP
[13]	Stark	public	yes	$O_\lambda(\log^2 N)$	3.2 MB	$O_\lambda(N)$	apply [21]-transform to IOP
this work	Aurora	public	yes	$O_\lambda(\log^2 N)$	220 kB	$O_\lambda(N)$	apply [21]-transform to IOP

Fig. 2: Comparison of some non-interactive zero knowledge arguments for proving statements of the form "there exists a secret w such that $\mathcal{C}(x, w) = 1$" for a given arithmetic circuit $\mathcal{C}$ of N gates (and depth d) and public input x of size k. The table is grouped by "technology", and for simplicity assumes that the circuit's underlying field has size $2^{O(\lambda)}$ where λ is the security parameter. Approximate argument sizes are given for $N = 10^6$ gates over a cryptographically-large field, and a security level of 128 bits; some argument sizes may differ from those reported in the cited works because size had to be re-computed for the security level and N used here; also, [83] reports no implementation.
† Given a per-circuit preprocessing step.
‡ A tradeoff between argument size and verifier time is possible; see [79].

We describe the high-level structure of our IOP protocol for R1CS, which has linear proof length and logarithmic query complexity. The protocol tests satisfaction by relying on two building blocks, one for testing the entry-wise vector product and the other for testing the linear transformations induced by the matrices A, B, C. Informally, we thus consider protocols for the following two problems.

- **Rowcheck:** given vectors $x, y, z \in \mathbb{F}^m$, test whether $x \circ y = z$, where "$\circ$" denotes entry-wise product.
- **Lincheck:** given vectors $x \in \mathbb{F}^m, y \in \mathbb{F}^n$ and a matrix $M \in \mathbb{F}^{m \times n}$, test whether $x = My$.

One can immediately obtain an IOP for R1CS when given IOPs for the rowcheck and lincheck problems. The prover first sends four oracles to the verifier: the satisfying assignment z and its linear transformations $y_A := Az, y_B := Bz, y_C := Cz$. Then the prover and verifier engage in four IOPs in parallel:
– An IOP for the lincheck problem to check that "$y_A = Az$". Likewise for y_B and y_C.
– An IOP for the rowcheck problem to check that "$y_A \circ y_B = y_C$".
Finally, the verifier checks that z is consistent with the public input v. Clearly, there exist z, y_A, y_B, y_C that yield valid rowcheck and lincheck instances if and only if (A, B, C, v) is a satisfiable R1CS instance.

The foregoing reduces the goal to designing IOPs for the rowcheck and lincheck problems.

As stated, however, the rowcheck and lincheck problems only admit "trivial" protocols in which the verifier queries all entries of the vectors in order to check the required properties. In order to allow for sublinear query complexity, we need the vectors x, y, z to be *encoded* via some error-correcting code. We use the Reed–Solomon (RS) code because it ensures constant distance with constant rate while at the same time it enjoys efficient IOPs of Proximity [14].

Given an evaluation domain $L \subseteq \mathbb{F}$ and rate parameter $\rho \in [0, 1]$, $\mathrm{RS}[L, \rho]$ is the set of all codewords $f \colon L \to \mathbb{F}$ that are evaluations of polynomials of degree less than $\rho|L|$. Then, the encoding of a vector $v \in \mathbb{F}^S$ with $S \subseteq \mathbb{F}$ and $|S| < \rho|L|$ is $\hat{v}|_L \in \mathbb{F}^L$ where $\hat{v}$ is the unique polynomial of degree $|S| - 1$ such that $\hat{v}|_S = v$. Given this encoding, we consider "encoded" variants of the rowcheck and lincheck problems.

- **Univariate rowcheck**: given a subset $H \subseteq \mathbb{F}$ and codewords $f, g, h \in \mathrm{RS}[L, \rho]$, check that $\hat{f}(a) \cdot \hat{g}(a) - \hat{h}(a) = 0$ for all $a \in H$. (This is a special case of the definition that we use later.)
- **Univariate lincheck**: given subsets $H_1, H_2 \subseteq \mathbb{F}$, codewords $f, g \in \mathrm{RS}[L, \rho]$, and a matrix $M \in \mathbb{F}^{H_1 \times H_2}$, check that $\hat{f}(a) = \sum_{b \in H_2} M_{a,b} \cdot \hat{g}(b)$ for all $a \in H_1$.

Given IOPs for the above problems, we can now get an IOP protocol for R1CS roughly as before. Rather than sending z, Az, Bz, Cz, the prover sends their encodings $f_z, f_{Az}, f_{Bz}, f_{Cz}$. The prover and verifier then engage in rowcheck and lincheck protocols as before, but with respect to these encodings.

$x_i \leftarrow x_j \cdot x_k$ is mapped to the quadratic constraint $x_i = x_j \cdot x_k$. The resulting system of equations can be written as $A \cdot ((1, x) \otimes (1, x)) = b$ for suitable $A \in \mathbb{F}^{m \times (n+1)^2}$ and $b \in \mathbb{F}^m$. However, this reduction results in a quadratic blowup in the instance size. There is an alternative reduction due to [62, 45] that avoids this.

For these encoded variants, we achieve IOP protocols with linear proof length and logarithmic query complexity, as required. We obtain a protocol for rowcheck via standard techniques from the probabilistic checking literature [27]. As for lincheck, we do not use any routing and instead use a technique (dating back at least to [9]) to reduce the given testing problem to a *sumcheck instance*. However, since we are not working with multivariate polynomials, we cannot rely on the usual (multivariate) sumcheck protocol. Instead, we present a novel protocol that realizes a univariate analogue of the classical sumcheck protocol, and use it as the testing "core" of our IOP protocol for R1CS. We discuss univariate sumcheck next.

Remark 1. The verifier receives as input an explicit (non-uniform) description of the set of constraints, namely, the matrices A, B, C. In particular, the verifier runs in time that is at least linear in the number of non-zero entries in these matrices (if we consider a sparse-matrix representation for example).

2.2 A sumcheck protocol for univariate polynomials

A key ingredient in our IOP protocol is a *univariate* analogue of the classical (multivariate) sumcheck protocol [61]. Recall that the classical sumcheck protocol is an IP for claims of the form "$\sum_{\boldsymbol{a} \in H^m} f(\boldsymbol{a}) = 0$", where f is a given polynomial in $\mathbb{F}[X_1, \ldots, X_m]$ of individual degree d and H is a subset of $\mathbb{F}$. In this protocol, the verifier runs in time $\mathrm{poly}(m, d, \log |\mathbb{F}|)$ and accesses f at a single (random) location. The sumcheck protocol plays a fundamental role in computational complexity (it underlies celebrated results such as $\mathsf{IP} = \mathsf{PSPACE}$ [72] and $\mathsf{MIP} = \mathsf{NEXP}$ [10]) and in efficient proof protocols [48, 39, 75, 73, 74, 76, 77, 82, 83, 79].

We work with univariate polynomials instead, and need a univariate analogue of the sumcheck protocol (see previous subsection): *how can a prover convince the verifier that "$\sum_{a \in H} f(a) = 0$" for a given polynomial $f \in \mathbb{F}[X]$ of degree d and subset $H \subseteq \mathbb{F}$?* Designing a "univariate sumcheck" is not straightforward because univariate polynomials (the Reed–Solomon code) do not have the tensor structure used by the sumcheck protocol for multivariate polynomials (the Reed–Muller code). In particular, the sumcheck protocol has m rounds, each of which reduces a sumcheck problem to a simpler sumcheck problem with one variable fewer. When there is only one variable, however, it is not clear to what simpler problems one can reduce.

Using different ideas, we design a natural protocol for univariate sumcheck in the cases where H is an additive or multiplicative coset in $\mathbb{F}$ (i.e., a coset of an additive or multiplicative subgroup of $\mathbb{F}$).

Theorem 2 (informal). *The univariate sumcheck protocol over additive or multiplicative cosets has a $O(\log d)$-round IOP with proof complexity $O(d)$ over alphabet $\mathbb{F}$ and query complexity $O(\log d)$. The IOP prover uses $O(d \log |H|)$ field operations and the IOP verifier uses $O(\log d + \log^2 |H|)$ field operations.*

We now provide the main ideas behind the protocol, when H is an *additive* coset in $\mathbb{F}$.

Suppose for a moment that the degree d of f is less than $|H|$ (we remove this restriction later). A theorem of Byott and Chapman [36] states that the sum of f over

(an additive coset) H is zero if and only if the coefficient of $X^{|H|-1}$ in f is zero. In particular, $\sum_{a \in H} f(a)$ is zero if and only if f has degree less than $|H| - 1$. Thus, the univariate sumcheck problem over H when $d < |H|$ is equivalent to low-degree testing.

The foregoing suggests a natural approach: test that f has degree less than $|H| - 1$. Without any help from the prover, the verifier would need at least $|H|$ queries to f to conduct such a test, which is as expensive as querying all of H. However, the prover can help by engaging with the verifier in an IOP of Proximity for the Reed–Solomon code. For this we rely on the recent construction of Ben-Sasson et al. [14], which has proof length $O(d)$ and query complexity $O(\log d)$.

In our setting, however, we need to also handle the case where the degree d of f is larger than $|H|$. For this case, we observe that we can split any polynomial f into two polynomials g and h such that $f(x) \equiv g(x) + \prod_{\alpha \in H}(x - \alpha) \cdot h(x)$ with $\deg(g) < |H|$ and $\deg(h) < d - |H|$; in particular, f and g agree on H, and thus so do their sums on H. This observation suggests the following extension to the prior approach: the prover sends g (as an oracle) to the verifier, and then the verifier performs the prior protocol with g in place of f. Of course, a cheating prover may send a polynomial g that has nothing to do with f, and so the verifier must also ensure that g is consistent with f. To facilitate this, we actually have the prover send h rather than g; the verifier can then "query" $g(x)$ as $f(x) - \prod_{\alpha \in H}(x - \alpha) \cdot h(x)$; the prover then shows that f, g, h are all of the correct degrees.

A similar reasoning works when H is a multiplicative coset in $\mathbb{F}$. It remains an interesting open problem to establish whether the foregoing can be extended to any subset H in $\mathbb{F}$.

2.3 Efficient zero knowledge from algebraic techniques

The ideas discussed thus far yield an IOP protocol for R1CS with linear proof length and logarithmic query complexity. However these by themselves do not provide zero knowledge.

We achieve zero knowledge by leveraging recent algebraic techniques [17]. Informally, we adapt these techniques to achieve efficient zero knowledge variants of key sub-protocols, including the univariate sumcheck protocol and low-degree testing, and combine these to achieve a zero knowledge IOP protocol for R1CS.

We summarize the basic intuition for how we achieve zero knowledge in our protocols.

First, we use *bounded independence*. Informally, rather than encoding a vector $z \in \mathbb{F}^H$ by the unique polynomial of degree $|H| - 1$ that matches z on H, we instead sample uniformly at random a polynomial of degree, say, $|H| + 9$ conditioned on matching z on H. Any set of 10 evaluations of such a polynomial are independently and uniformly distributed in $\mathbb{F}$ (and thus reveal no information about z), *provided these evaluations are outside of* H. To ensure this latter condition, we choose the evaluation domain L of Reed–Solomon codewords to be disjoint from H. Thus, for example, if H is a linear space (an additive subgroup of $\mathbb{F}$) then we choose L to be an affine subspace (a coset of some additive subgroup), since the underlying machinery for low-degree testing (e.g., [14]) requires codewords to be evaluated over algebraically-structured domains. All of our protocols are robust to these variations.

Bounded independence alone does not suffice, though. For example, in the sumcheck protocol, consider the case where the input vector $z \in \mathbb{F}^H$ is all zeroes. The prover samples a random polynomial $\hat{f}$ of degree $|H| + 9$, such that $\hat{f}(a) = 0$ for all $a \in H$, and sends its evaluation f over L disjoint from H to the verifier. As discussed, any ten queries to f result in ten independent and uniformly random elements in $\mathbb{F}$. Observe, however, that when we run the sumcheck protocol on f, the polynomial g (the remainder of $\hat{f}$ when divided by $\prod_{\alpha \in H}(x - \alpha)$) is the zero polynomial: all randomness is removed by the division.

To remedy this, we use *self-reducibility* to reduce a sumcheck claim about the polynomial f to a sumcheck claim about a random polynomial. The prover first sends a random Reed–Solomon codeword r, along with the value $\beta := \sum_{a \in H} r(a)$. The verifier sends a random challenge $\rho \in \mathbb{F}$. Then the prover and verifier engage in the univariate sumcheck protocol with respect to the new claim "$\sum_{a \in H} \rho f(a) + r(a) = \beta$". Since r is uniformly random, $\rho f + r$ is uniformly random for any ρ, and thus the sumcheck protocol is performed on a random polynomial, which ensures zero knowledge. Soundness is ensured by the fact that if f does not sum to 0 on H then the new claim is true with probability $1/|\mathbb{F}|$ over the choice of ρ.

2.4 Perspective on our techniques

A linear-length logarithmic-query IOP for a "circuit-like" NP-complete relation like R1CS (Theorem 1) may come as a surprise. We wish to shed some light on our IOP construction by connecting the ideas behind it to prior ideas in the probabilistic checking literature, and use these connections to motivate our construction.

A significant cost in all known PCP constructions with good proof length is using *routing networks* to reduce combinatorial objects (circuits, machines, and so on) to structured algebraic ones;[8] routing also plays a major role in many IOPs [17, 15, 12, 13]. While it is plausible that one could adapt routing techniques to route the constraints of an R1CS instance (similarly to [68]), such an approach would likely incur logarithmic-factor overheads, precluding *linear*-size IOPs.

A recent work [16] achieves linear-length constant-query IOPs for boolean circuit satisfaction *without routing the input circuit*. Unfortunately, [16] relies on other expensive tools, such as algebraic-geometry (AG) codes and quasilinear-size PCPs of proximity [27]; moreover, it is not zero knowledge. Informally, [16] tests arbitrary (unstructured) constraints by invoking a sumcheck protocol [61] on a $O(1)$-wise tensor product of AG codes; this latter is then locally tested via tools in [26, 27].

One may conjecture that, to achieve an IOP for R1CS like ours, it would suffice to merely replace the AG codes in [16] with the Reed–Solomon code, since both codes have constant rate. But taking a tensor product exponentially deteriorates rate, and testing proximity to that tensor product would be expensive.

[8] Polishchuk and Spielman [68] reduce boolean circuit satisfaction to a trivariate algebraic coloring problem with "low-degree" neighbor relations, by routing the circuit's wires over an arithmetized routing network. Ben-Sasson and Sudan [27] reduce nondeterministic machine computations to a univariate algebraic satisfaction problem by routing the machine's memory accesses over another arithmetized routing network. Routing is again a crucial component in the linear-size sublinear-query PCPs of [24].

An alternative approach is to solve a sumcheck problem *directly* on the Reed–Solomon code. Existing protocols are not of much use here: the multivariate sumcheck protocol relies on a tensor structure that is *not* available in the Reed–Solomon code, and recent IOP implementations either use routing [12, 13] or achieve only sublinear query complexity [4].

Instead, we design a completely new IOP for a sumcheck problem on the Reed–Solomon code. We then combine this solution with ideas from [16] (to avoid routing) and from [17] (to achieve zero knowledge) to obtain our linear-length logarithmic-query IOP for R1CS. Along the way, we rely on recent efficient proximity tests for the Reed–Solomon code [14].

3 Roadmap

In Section 4 we evaluate Aurora, and compare it to other IOP-based SNARGs. In Section 5 we describe the implementation. In Section 6 we present the underlying IOP for R1CS. Fig. 3 summarizes the structure of this protocol. For details of this construction, including proofs of theorems, we refer the reader to the full version.

Throughout, we focus on the case where all relevant domains are *additive* cosets (affine subspaces) in $\mathbb{F}$. The case where domains are *multiplicative* cosets is similar, with only minor modifications. Moreover, while for convenience we limit our discussions to establishing soundness, all protocols described in this paper are easily seen to satisfy the stronger notion of proof of knowledge. Informally, this is because we prove soundness by showing that oracles sent by convincing provers can be decoded to valid witnesses.

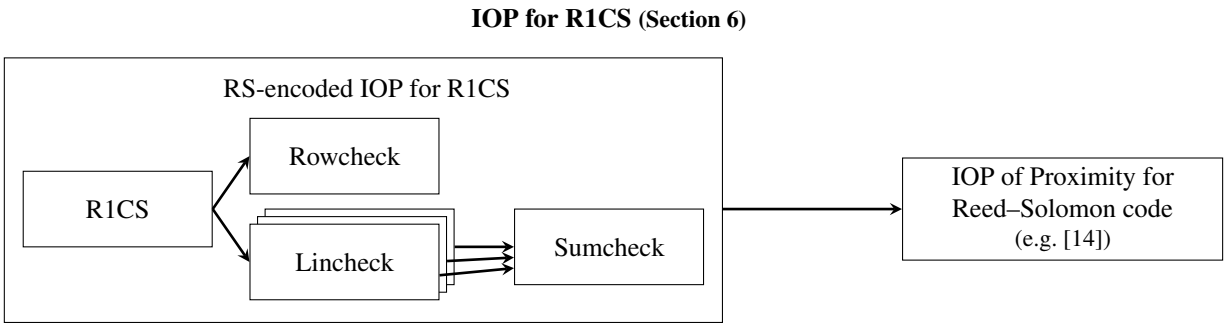


Fig. 3: Structure of our IOP for R1CS in terms of key sub-protocols.

4 Evaluation

In Section 4.1 we evaluate the performance of Aurora. Then, in Section 4.2 we compare Aurora with Ligero [4] and Stark [13], two other IOP-based SNARGs. Our experiments not only demonstrate that Aurora's performance matches the theoretical predictions

implied by the protocol but also that Aurora achieves the smallest argument size of any IOP-based SNARG, *by more than an order of magnitude*.

That said, there is still a sizable gap between the argument sizes of IOP-based SNARGs and other SNARGs that use public-key cryptographic assumptions vulnerable to quantum adversaries; see Fig. 2 for how argument sizes vary across these. It remains an exciting open problem to close this gap.

Experiments ran on a machine with an Intel Xeon W-2155 3.30GHz 10-core processor and 64GB of RAM.

4.1 Performance of Aurora

We consider Aurora at the standard security level of 128 bits, over the binary field $\mathbb{F}_{2^{192}}$. We report data on key efficiency measures of a SNARG: the time to generate a proof (running time of the prover), the length of a proof, and the time to check a proof (running time of the verifier). We also indicate how much of each cost is due to the IOP protocol, and how much is due to the BCS transformation [21].

In Aurora, all of these quantities depend on the number of constraints m in an R1CS instance.[9] Our experiments report how these quantities change as we vary m over the range $\{2^{10}, 2^{11}, \ldots, 2^{20}\}$.

Prover running time. In Fig. 4 we plot the running time of the prover, as absolute cost (top graph) and as relative cost when compared to native execution (bottom graph). For R1CS, *native execution* is the time that it takes to check that an assignment satisfies the constraint system. The plot confirms the quasilinear complexity of the prover; proving times range from fractions of a second to several minutes. Proving time is dominated by the cost of running the underlying IOP prover.

Argument size. In Fig. 5 we plot argument size, as absolute cost (top graph) and as relative cost when compared to native witness size (bottom graph). For R1CS, *native witness size* is the number of bytes required to represent an assignment to the constraint system. The plot shows that compression (argument size is smaller than native witness size) occurs for $m \geq 4000$. The plot also shows that argument size ranges from $50\,\mathrm{kB}$ to $250\,\mathrm{kB}$, and is dominated by the cryptographic digests to authenticate query answers.

Verifier running time. In Fig. 6 we plot the running time of the verifier, as absolute cost (top graph) and as relative cost when compared to native execution (bottom graph). The plot shows that verification times range from milliseconds to seconds, and confirms that our implementation incurs a constant multiplicative overhead over native execution.

4.2 Comparison of Ligero, Stark, and Aurora

In Figs. 7 to 9 we compare costs (proving time, argument size, and verification time) on R1CS instances for three IOP-based SNARGs: Ligero [4], Stark [13], and Aurora (this work). As in Section 4.1, we plot costs as the number of constraints m increases (and

[9] The number of variables n also affects performance, but it is usually close to m and so we take $n \approx m$ in our experiments. The number of inputs k in an R1CS instance is at most n, and in typical applications it is much smaller than n, so we do not focus on it.

with $n \approx m$ variables as explained in Footnote 9); we also set security to the standard level of 128 bits and use the binary field $\mathbb{F}_{2^{192}}$.

Comparison of Ligero and Aurora. Ligero natively supports R1CS so a comparison with Aurora is straightforward. Fig. 8 shows that argument size in Aurora is much smaller than in Ligero, even for a relatively small number of constraints. The gap between the two grows bigger as the number of constraints increases, as Aurora's argument size is polylogarithmic while Ligero's is only sublinear (an exponential gap).

Comparison of Stark and Aurora. Stark does not natively support the NP-complete relation R1CS but instead natively supports an NEXP-complete relation known as *Algebraic Placement and Routing* (APR). These two relations are quite different,and so to achieve a meaningful comparison, we consider an APR instance that *simulates* a given R1CS instance. We thus plot the costs of Stark on a hand-optimized APR instance that simulates R1CS instances. Relying on the reductions described in [13], we wrote an APR instance that realizes a simple abstract computer that checks that a variable assignment satisfies each one of the rank-1 constraints in a given R1CS instance.

Fig. 8 shows that argument size in Aurora is much smaller than in Stark, even if both share the same asymptotic growth. This is due to the fact that R1CS and APR target different computation models (explicit circuits vs. uniform computations), so Stark incurs significant overheads when used for R1CS. Fig. 9 shows that verification time in Stark grows linearly with the number of constraints (like Ligero and Aurora); indeed, the verifier must read the description of the statement being proved, which is the entire constraint system.

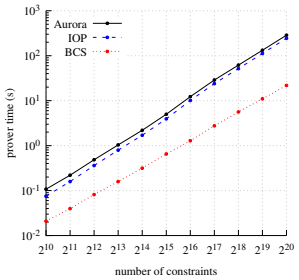


Fig. 4: Proving time in Aurora.

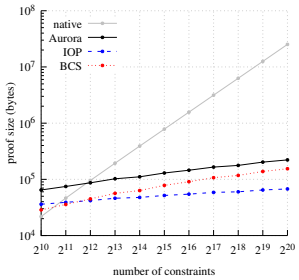


Fig. 5: Argument size in Aurora.

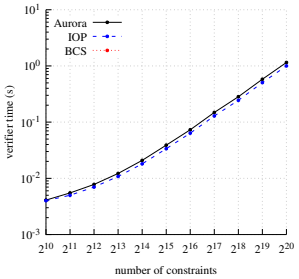
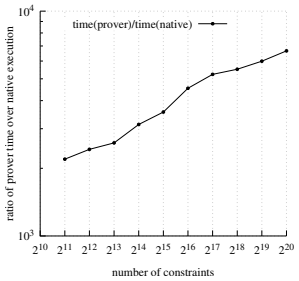
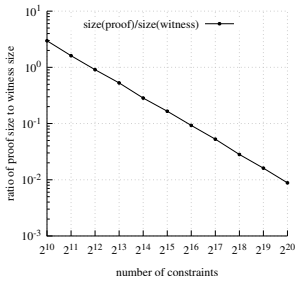
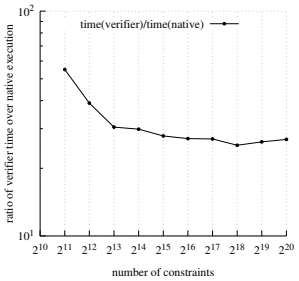


Fig. 6: Verification time in Aurora.







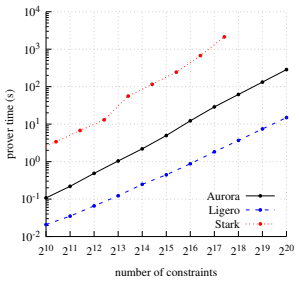


Fig. 7: Proving time in Aurora, Ligero, Stark.

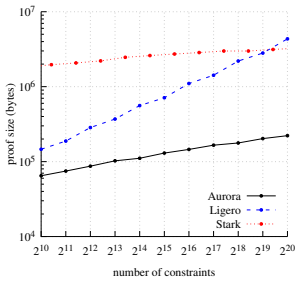


Fig. 8: Argument size in Aurora, Ligero, Stark.

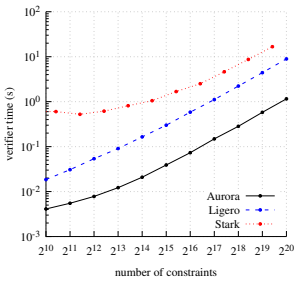


Fig. 9: Verification time in Aurora, Ligero, Stark.

5 `libiop`: a library for IOP-based non-interactive arguments

We provide `libiop`, a codebase that enables the design and implementation of IOP-based non-interactive arguments. The codebase uses the C++ language and has three main components: (1) a library for writing IOP protocols; (2) a realization of the [21] transformation, mapping any IOP written with our library to a corresponding non-interactive argument; (3) a portfolio of IOP protocols, including our new IOP protocol for R1CS and IOP protocols from [4] and [13]. We discuss each of these components in turn.

5.1 Library for IOP protocols

We provide a library that enables a programmer to write IOP protocols. Informally, the programmer provides a blueprint of the IOP by specifying, for each round, the number and sizes of oracle messages (and non-oracle messages) sent by the prover, as well as the number of random bytes subsequently sent by the verifier. For the prover, the programmer specifies how each message is to be computed. For the verifier, the programmer specifies how oracle queries are generated and, also, how the verifier's decision is computed based on its random choices and information received from the prover. Notable features of our library include:

- Support for writing new IOPs by using other IOPs as sub-protocols. This includes juxtaposing or interleaving selected rounds of these sub-protocols. This latter feature not only facilitates reducing round complexity in complex IOP constructions but also makes it possible to take advantage of optimizations such as column hashing (discussed in Section 5.2) when constructing a non-interactive argument.
- A realization of the transformation described in the full version, which constructs an IOP by combining an 'encoded' IOP and a low-degree test. This is a powerful paradigm (it applies to essentially all published IOP protocols) that reduces the task of writing an IOP to merely providing suitable choices of these two simpler ingredients.

5.2 BCS transformation

We realize the transformation of [21], by providing code that maps any IOP written in our library into a corresponding non-interactive argument (which consists of a prover algorithm and a verifier algorithm).

We use BLAKE2b [8] to instantiate the random oracle in the [21] transformation (our code allows to conveniently specify alternative instantiations). This hash function is an improvement to BLAKE (a finalist in the SHA-3 competition) [7], and its performance on all recent x86 platforms is competitive with the most performant (and often hardware-accelerated) hash functions [42]. Moreover, BLAKE2b can be configured to output digests of any length between 1 and 64 bytes (between 8 and 512 bits in multiples of 8). When aiming for a security level of λ bits, we only need the hash function to output digests of 2λ bits, and our code automatically sets this length.

Our code incorporates additional optimizations that, while simple, are generic and effective.

One is *column hashing*, which informally works as follows. In many IOP protocols (essentially all published ones, including Ligero [4] and Stark [13]), the prover sends multiple oracles over the same domain in the same round, and the verifier accesses all of them at the same index in the domain. The prover can then build a Merkle tree over *columns* consisting of corresponding entries of the oracles, rather than building separate Merkle trees for each or a single Merkle tree over their concatenation. This reduces a non-interactive proof's length, because the proof only has to contain a *single* authentication path for the desired column, rather than authentication paths corresponding to the indices across all the oracles.

Another optimization is *path pruning*. When providing multiple authentication paths relative to the same root (in the non-interactive argument), some digests become redundant and can thus be omitted. For example, if one considers the authentication paths for all leaves in a particular sub-tree, then one can simple provide the authentication path for the root of the sub-tree. A simple way to view this optimization is to provide the smallest number of digests to authenticate a *set* of leaves.

5.3 Portfolio of IOP protocols and sub-components

We use our library to realize several IOP protocols:

- **Aurora:** our IOP protocol for R1CS (specifically, the one provided in Fig. 12 in Section 6).
- **Ligero:** an adaptation of the IOP protocol in [4] to R1CS. While the protocol(s) in [4] are designed for (boolean or arithmetic) circuit satisfiability, the same ideas can be adapted to support R1CS *at no extra cost*. This simplifies comparisons with R1CS-based arguments, and confers additional expressivity.
- **Stark:** the IOP protocol in [13] for *Algebraic Placement and Routing* (APR), a language that is a "succinct" analogue of algebraic satisfaction problems such as R1CS. (See [13] for details.)

Each of the above IOPs is obtained by specifying an encoded IOP and a low-degree test. As explained in Sections 5.1 and 5.2, our library compiles these into an IOP protocol, and the latter into a non-interactive argument. This toolchain enables specifying protocols with few lines of code (see Fig. 10), and also enhances code auditability.

The IOP protocols above benefit from several algebraic components that our library also provides.

- *Finite field arithmetic.* We support prime and binary fields. Our prime field arithmetic uses Montgomery representation [64]. Our binary field arithmetic uses the carryless multiplication instructions [53]; these are ubiquitous in x86 CPUs and, being used in AES-GCM computations, are highly optimized.
- *FFT algorithms.* The choice of FFT algorithm depends on whether the R1CS instance (and thus the rest of the protocol) is defined over a prime or binary field. In the former case, we use the radix-2 FFT (whose evaluation domain is a multiplicative coset of order 2^a for some a) [38]. In the latter case, we use an additive FFT (whose evaluation domain is an affine subspace of the binary field) [37, 44, 28, 59, 58]. We also provide the respective inverse FFTs, and variants for cosets of the base domains.

Remark 2. Known techniques can be used to reduce given programs or general machine computations to low-level representations such as R1CS and APR (see, e.g., [23, 78, 13]). Such techniques have been compared in prior work, and our library does not focus on these.

encoded IOP protocol	lines of code
Stark	321
Ligero	1281
Aurora	1165

low-degree test	lines of code
FRI	416
direct	212

Fig. 10: Lines of code to express various sub-components in our library.

6 Aurora: an IOP for rank-one constraint satisfaction (R1CS)

We describe the IOP for R1CS that comprises the main technical contribution of this paper, and also underlies the SNARG for R1CS that we have designed and built (more about this in Section 5).

For the discussions below, we introduce notation about the low-degree test in [14], known as "Fast Reed–Solomon IOPP" (FRI): given a subspace L of a binary field $\mathbb{F}$ and rate $\rho \in (0,1)$, we denote by $\varepsilon_{\mathsf{i}}^{\mathrm{FRI}}(\mathbb{F}, L)$ and $\varepsilon_{\mathsf{q}}^{\mathrm{FRI}}(L, \rho, \delta)$ the soundness error of the interactive and query phases in FRI (respectively) when testing proximity of a δ-far function to $\mathrm{RS}\,[L, \rho]$.

We first provide a "barebones" statement with constant soundness error and no zero knowledge.

Theorem 3. *There is an IOP for the R1CS relation over binary fields $\mathbb{F}$ that, given an R1CS instance having n variables and m constraints, letting $\rho \in (0,1)$ be a constant and L be any subspace of $\mathbb{F}$ such that $2\max(m, n+1) \leq \rho|L|$, has the following parameters:*

$$\left|\begin{array}{lll} \textit{alphabet} & \Sigma & = \mathbb{F} \\ \textit{number of rounds} & \mathsf{k} & = O(\log|L|) \\ \textit{proof length} & \mathsf{p} & = (5+\frac{1}{3})|L| \\ \textit{query complexity} & \mathsf{q}_\pi & = O(\log|L|) \\ \textit{randomness} & (\mathsf{r}_\mathsf{i}, \mathsf{r}_\mathsf{q}) & = (O(\log|L| \cdot \log|\mathbb{F}|), O(\log|L|)) \\ \textit{soundness error} & (\varepsilon_\mathsf{i}, \varepsilon_\mathsf{q}) & = \left(\frac{m+1}{|\mathbb{F}|} + \frac{|L|}{|\mathbb{F}|} + \varepsilon_\mathsf{i}^{\mathrm{FRI}}(\mathbb{F}, L), \varepsilon_\mathsf{q}^{\mathrm{FRI}}(L, \rho, \delta)\right) \\ \textit{prover time} & \mathsf{t}_\mathsf{P} & = O(|L| \cdot \log(n+m) + \|A\| + \|B\| + \|C\|) + 17 \cdot \mathrm{FFT}(\mathbb{F}, |L|) \\ \textit{verifier time} & \mathsf{t}_\mathsf{V} & = O(\|A\| + \|B\| + \|C\| + n + m + \log|L|) \end{array}\right|,$$

where $\delta := \min(\frac{1-2(\rho/2)}{2}, \frac{1-(\rho/2)}{3}, 1-\rho)$.

Next, we provide a statement that additionally has parameters for controlling the soundness error, is zero knowledge, and includes other (whitebox) optimizations; the

proof is analogous except that we use zero knowledge components. The resulting IOP protocol, fully specified in Fig. 12, underlies our SNARG for R1CS (see Section 5).

Theorem 4. *There is an IOP for the R1CS relation over binary fields $\mathbb{F}$ that, given an R1CS instance having n variables and m constraints, letting $\rho \in (0,1)$ be a constant and L be any subspace of $\mathbb{F}$ such that $2\max(m, n+1) + 2\mathsf{b} \leq \rho|L|$, is zero knowledge against b queries and has the following parameters:*

$$\left|\begin{array}{lll} \textit{alphabet} & \Sigma & = \mathbb{F} \\ \textit{number of rounds} & \mathsf{k} & = O(\log|L|) \\ \textit{proof length} & \mathsf{p} & = (4 + 2\lambda_{\mathsf{i}} + \lambda'_{\mathsf{i}}\lambda_{\mathsf{i}}^{\mathrm{FRI}}/3)|L| \\ \textit{query complexity} & \mathsf{q}_\pi & = O(\lambda_{\mathsf{i}}\lambda_{\mathsf{i}}^{\mathrm{FRI}}\lambda_{\mathsf{q}}^{\mathrm{FRI}}\log|L|) \\ \textit{randomness} & (\mathsf{r}_{\mathsf{i}}, \mathsf{r}_{\mathsf{q}}) & = \big(O((\lambda_{\mathsf{i}}\lambda'_{\mathsf{i}} + \lambda_{\mathsf{i}}^{\mathrm{FRI}}\log|L|)\log|\mathbb{F}|), O(\lambda_{\mathsf{q}}^{\mathrm{FRI}}\log|L|)\big) \\ \textit{soundness error} & (\varepsilon_{\mathsf{i}}, \varepsilon_{\mathsf{q}}) & = \Big((\tfrac{m+1}{|\mathbb{F}|})^{\lambda_{\mathsf{i}}} + (\tfrac{|L|}{|\mathbb{F}|})^{\lambda'_{\mathsf{i}}} + \varepsilon_{\mathsf{i}}^{\mathrm{FRI}}(\mathbb{F}, L)^{\lambda_{\mathsf{i}}^{\mathrm{FRI}}}, \varepsilon_{\mathsf{q}}^{\mathrm{FRI}}(L, \rho, \delta)^{\lambda_{\mathsf{q}}^{\mathrm{FRI}}}\Big) \\ \textit{prover time} & \mathsf{t}_{\mathsf{P}} & = \lambda_{\mathsf{i}} \cdot (O(|L| \cdot (\log(n+m) + \|A\| + \|B\| + \|C\|) + 18 \cdot \mathrm{FFT}(\mathbb{F}, |L|)) + O(\lambda'_{\mathsf{i}}\lambda_{\mathsf{i}}^{\mathrm{FRI}}|L|) \\ \textit{verifier time} & \mathsf{t}_{\mathsf{V}} & = \lambda_{\mathsf{i}} \cdot O(\|A\| + \|B\| + \|C\| + n + m + \log|L|) + O(\lambda'_{\mathsf{i}}\lambda_{\mathsf{i}}^{\mathrm{FRI}}\lambda_{\mathsf{q}}^{\mathrm{FRI}}\log|L|) \end{array}\right|,$$

where $\delta := \min(\frac{1-2\rho}{2}, \frac{1-\rho}{3}, 1-\rho)$. Setting $\mathsf{b} \geq \mathsf{q}_\pi$ ensures honest-verifier zero knowledge.

Given an R1CS instance $(\mathbb{F}, k, n, m, A, B, C, v)$, we fix subspaces $H_1, H_2 \subseteq \mathbb{F}$ such that $|H_1| = m$ and $|H_2| = n+1$ (padding to the nearest power of 2 if necessary) with $H_1 \subseteq H_2$ or $H_2 \subseteq H_1$, and a sufficiently large affine subspace $L \subseteq \mathbb{F}$ such that $L \cap (H_1 \cup H_2) = \varnothing$. We let $t := |H_1 \cup H_2| = \max(m, n+1)$. Fig. 11 below gives polynomials and codewords used in Fig. 12. We also define $\xi := \sum_{a \in H_1 \cup H_2} a^{t-1}$.

polynomial	**degree**	**values that define the polynomial**
p_α	$t-1$	$\hat{p}_\alpha(a) = \begin{cases} \alpha^{\gamma(a)} & \text{for } a \in H_1 \\ 0 & \text{for } a \in (H_1 \cup H_2) \setminus H_1 \end{cases}$
$p_\alpha^{(M)}$	$t-1$	$\hat{p}_\alpha^{(M)}(b) = \begin{cases} \sum_{a \in H_1} M_{a,b} \cdot \alpha^{\gamma(a)} & \text{for } b \in H_2 \\ 0 & \text{for } b \in (H_1 \cup H_2) \setminus H_2 \end{cases}$
codeword	**code**	**polynomial that defines the codeword**
f_w	$\mathrm{RS}\left[L, \frac{n-k+\mathsf{b}}{\lvert L\rvert}\right]$	random polynomial $\bar{f}_w$ of degree less than $n-k+\mathsf{b}$ such that, for all $b \in H_2$ with $k < \gamma(b) \leq n$, $\bar{f}_w(b) = \dfrac{w_{\gamma(b)-k} - \hat{f}_{(1,v)}(b)}{\mathbb{Z}_{H_2^{\leq k}}(b)}$
f_{Mz}	$\mathrm{RS}\left[L, \frac{m+\mathsf{b}}{\lvert L\rvert}\right]$	random polynomial $\bar{f}_{Az}$ of degree less than $m+\mathsf{b}$ such that, for all $a \in H_1$, $\bar{f}_{Az}(a) = \sum_{b \in H_2} M_{a,b} \cdot z_{\gamma(b)} = (Mz)_a$

Fig. 11: Polynomials and codewords used in the IOP protocol given in Fig. 12.

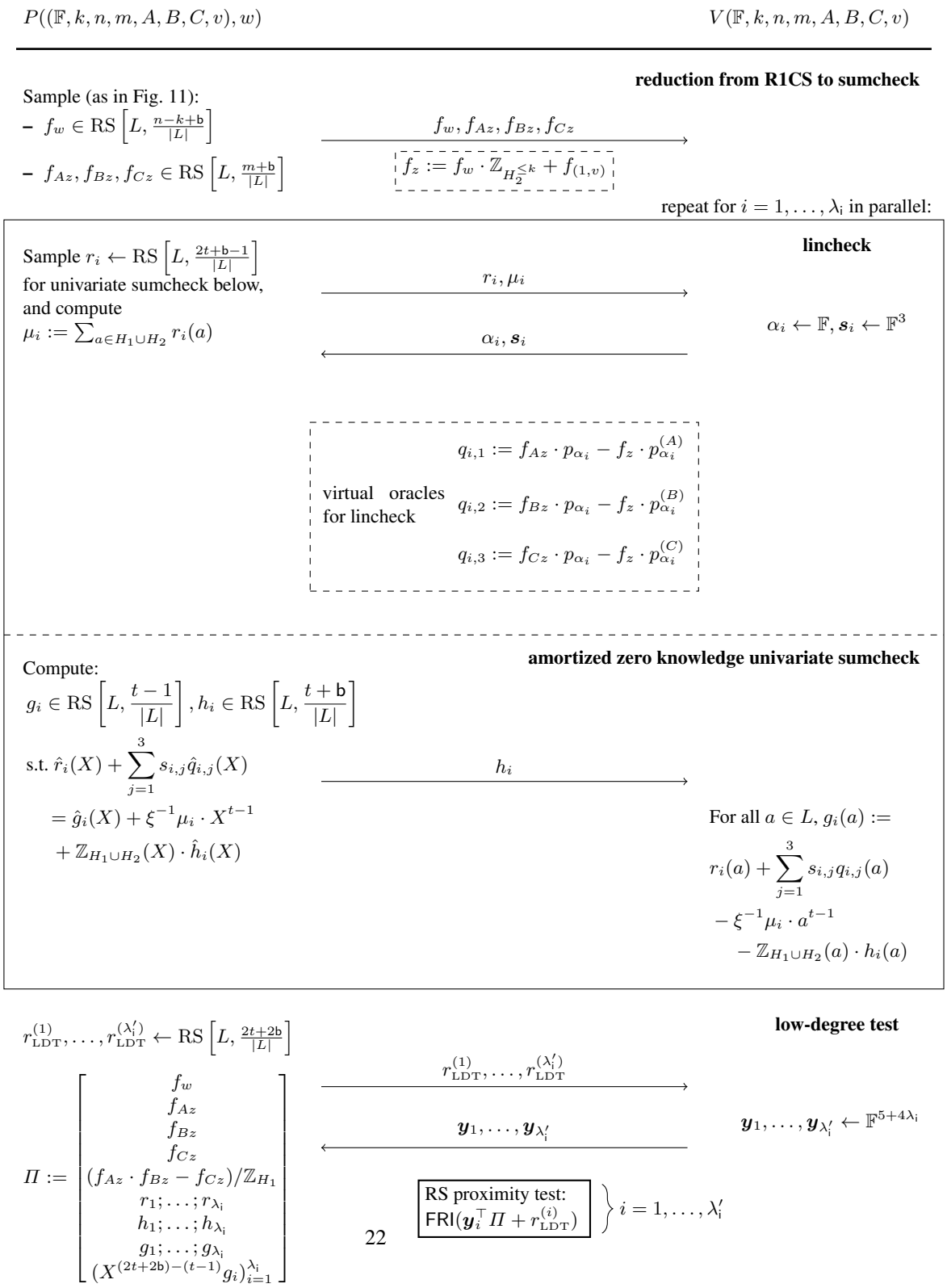


Fig. 12: Diagram of the zero knowledge IOP for R1CS that proves Theorem 4.

References

1. ZCash Company (2014), https://z.cash/
2. The Zcash Ceremony (2016), https://z.cash/blog/the-design-of-the-ceremony.html
3. Zero knowledge proof standardization (2017), https://zkproof.org/
4. Ames, S., Hazay, C., Ishai, Y., Venkitasubramaniam, M.: Ligero: Lightweight sublinear arguments without a trusted setup. In: Proceedings of the 24th ACM Conference on Computer and Communications Security. pp. 2087–2104. CCS '17 (2017)
5. Arora, S., Lund, C., Motwani, R., Sudan, M., Szegedy, M.: Proof verification and the hardness of approximation problems. Journal of the ACM **45**(3), 501–555 (1998), preliminary version in FOCS '92.
6. Arora, S., Safra, S.: Probabilistic checking of proofs: a new characterization of NP. Journal of the ACM **45**(1), 70–122 (1998), preliminary version in FOCS '92.
7. Aumasson, J.P., Meier, W., Phan, R., Henzen, L.: The Hash Function BLAKE. Springer-Verlag Berlin Heidelberg (2014)
8. Aumasson, J.P., Neves, S., Wilcox-O'Hearn, Z., Winnerlein, C.: BLAKE2: simpler, smaller, fast as MD5 (2013), https://blake2.net/blake2.pdf
9. Babai, L., Fortnow, L., Levin, L.A., Szegedy, M.: Checking computations in polylogarithmic time. In: Proceedings of the 23rd Annual ACM Symposium on Theory of Computing. pp. 21–32. STOC '91 (1991)
10. Babai, L., Fortnow, L., Lund, C.: Non-deterministic exponential time has two-prover interactive protocols. Computational Complexity **1**, 3–40 (1991), preliminary version appeared in FOCS '90.
11. Baum, C., Bootle, J., Cerulli, A., Pino, R.d., Groth, J., Lyubashevsky, V.: Sub-linear lattice-based zero-knowledge arguments for arithmetic circuits. In: Proceedings of the 38th Annual International Cryptology Conference. pp. 669–699. CRYPTO '18 (2018)
12. Ben-Sasson, E., Bentov, I., Chiesa, A., Gabizon, A., Genkin, D., Hamilis, M., Pergament, E., Riabzev, M., Silberstein, M., Tromer, E., Virza, M.: Computational integrity with a public random string from quasi-linear pcps. In: Proceedings of the 36th Annual International Conference on Theory and Application of Cryptographic Techniques. pp. 551–579. EUROCRYPT '17 (2017)
13. Ben-Sasson, E., Bentov, I., Horesh, Y., Riabzev, M.: Scalable, transparent, and post-quantum secure computational integrity. Cryptology ePrint Archive, Report 2018/046 (2018)
14. Ben-Sasson, E., Bentov, I., Horesh, Y., Riabzev, M.: Fast Reed–Solomon interactive oracle proofs of proximity. In: Proceedings of the 45th International Colloquium on Automata, Languages and Programming. pp. 14:1–14:17. ICALP '18 (2018)
15. Ben-Sasson, E., Chiesa, A., Forbes, M.A., Gabizon, A., Riabzev, M., Spooner, N.: Zero knowledge protocols from succinct constraint detection. In: Proceedings of the 15th Theory of Cryptography Conference. pp. 172–206. TCC '17 (2017)
16. Ben-Sasson, E., Chiesa, A., Gabizon, A., Riabzev, M., Spooner, N.: Interactive oracle proofs with constant rate and query complexity. In: Proceedings of the 44th International Colloquium on Automata, Languages and Programming. pp. 40:1–40:15. ICALP '17 (2017)
17. Ben-Sasson, E., Chiesa, A., Gabizon, A., Virza, M.: Quasilinear-size zero knowledge from linear-algebraic PCPs. In: Proceedings of the 13th Theory of Cryptography Conference. pp. 33–64. TCC '16-A (2016)
18. Ben-Sasson, E., Chiesa, A., Garman, C., Green, M., Miers, I., Tromer, E., Virza, M.: Zerocash: Decentralized anonymous payments from Bitcoin. In: Proceedings of the 2014 IEEE Symposium on Security and Privacy. pp. 459–474. SP '14 (2014)

19. Ben-Sasson, E., Chiesa, A., Genkin, D., Tromer, E., Virza, M.: SNARKs for C: Verifying program executions succinctly and in zero knowledge. In: Proceedings of the 33rd Annual International Cryptology Conference. pp. 90–108. CRYPTO '13 (2013)
20. Ben-Sasson, E., Chiesa, A., Green, M., Tromer, E., Virza, M.: Secure sampling of public parameters for succinct zero knowledge proofs. In: Proceedings of the 36th IEEE Symposium on Security and Privacy. pp. 287–304. S&P '15 (2015)
21. Ben-Sasson, E., Chiesa, A., Spooner, N.: Interactive oracle proofs. In: Proceedings of the 14th Theory of Cryptography Conference. pp. 31–60. TCC '16-B (2016)
22. Ben-Sasson, E., Chiesa, A., Tromer, E., Virza, M.: Scalable zero knowledge via cycles of elliptic curves. In: Proceedings of the 34th Annual International Cryptology Conference. pp. 276–294. CRYPTO '14 (2014), extended version at `http://eprint.iacr.org/2014/595`.
23. Ben-Sasson, E., Chiesa, A., Tromer, E., Virza, M.: Succinct non-interactive zero knowledge for a von Neumann architecture. In: Proceedings of the 23rd USENIX Security Symposium. pp. 781–796. Security '14 (2014), extended version at `http://eprint.iacr.org/2013/879`.
24. Ben-Sasson, E., Kaplan, Y., Kopparty, S., Meir, O., Stichtenoth, H.: Constant rate PCPs for Circuit-SAT with sublinear query complexity. In: Proceedings of the 54th Annual IEEE Symposium on Foundations of Computer Science. pp. 320–329. FOCS '13 (2013)
25. Ben-Sasson, E., Kopparty, S., Saraf, S.: Worst-case to average case reductions for the distance to a code. In: Proceedings of the 33rd ACM Conference on Computer and Communications Security. pp. 24:1–24:23. CCS '18 (2018)
26. Ben-Sasson, E., Sudan, M.: Robust locally testable codes and products of codes. Random Structures and Algorithms **28**(4), 387–402 (2006)
27. Ben-Sasson, E., Sudan, M.: Short PCPs with polylog query complexity. SIAM Journal on Computing **38**(2), 551–607 (2008), preliminary version appeared in STOC '05.
28. Bernstein, D.J., Chou, T.: Faster binary-field multiplication and faster binary-field MACs. In: Proceedings of the 21st International Conference on Selected Areas in Cryptography. pp. 92–111. SAC '14 (2014)
29. Bitansky, N., Chiesa, A., Ishai, Y., Ostrovsky, R., Paneth, O.: Succinct non-interactive arguments via linear interactive proofs. In: Proceedings of the 10th Theory of Cryptography Conference. pp. 315–333. TCC '13 (2013)
30. Boneh, D., Ishai, Y., Sahai, A., Wu, D.J.: Lattice-based SNARGs and their application to more efficient obfuscation. In: Proceedings of the 36th Annual International Conference on Theory and Applications of Cryptographic Techniques. pp. 247–277. EUROCRYPT '17 (2017)
31. Bootle, J., Cerulli, A., Chaidos, P., Groth, J., Petit, C.: Efficient zero-knowledge arguments for arithmetic circuits in the discrete log setting. In: Proceedings of the 35th Annual International Conference on Theory and Application of Cryptographic Techniques. pp. 327–357. EUROCRYPT '16 (2016)
32. Bowe, S., Gabizon, A., Green, M.: A multi-party protocol for constructing the public parameters of the Pinocchio zk-SNARK. Cryptology ePrint Archive, Report 2017/602 (2017)
33. Bowe, S., Gabizon, A., Miers, I.: Scalable multi-party computation for zk-SNARK parameters in the random beacon model. Cryptology ePrint Archive, Report 2017/1050 (2017)
34. Brassard, G., Chaum, D., Crépeau, C.: Minimum disclosure proofs of knowledge. Journal of Computer and System Sciences **37**(2), 156–189 (1988)
35. Bünz, B., Bootle, J., Boneh, D., Poelstra, A., Wuille, P., Maxwell, G.: Bulletproofs: Short proofs for confidential transactions and more. In: Proceedings of the 39th IEEE Symposium on Security and Privacy. pp. 315–334. S&P '18 (2018)
36. Byott, N.P., Chapman, R.J.: Power sums over finite subspaces of a field. Finite Fields and Their Applications **5**(3), 254–265 (Jul 1999)

37. Cantor, D.G.: On arithmetical algorithms over finite fields. Journal of Combinatorial Theory, Series A **50**(2), 285–300 (1989)
38. Cooley, J.W., Tukey, J.W.: An algorithm for the machine calculation of complex Fourier series. Mathematics of Computation **19**, 297–301 (1965)
39. Cormode, G., Mitzenmacher, M., Thaler, J.: Practical verified computation with streaming interactive proofs. In: Proceedings of the 4th Symposium on Innovations in Theoretical Computer Science. pp. 90–112. ITCS '12 (2012)
40. Costello, C., Fournet, C., Howell, J., Kohlweiss, M., Kreuter, B., Naehrig, M., Parno, B., Zahur, S.: Geppetto: Versatile verifiable computation. In: Proceedings of the 36th IEEE Symposium on Security and Privacy. pp. 250–273. S&P '15 (2015)
41. Cramer, R., Damgård, I.: Zero-knowledge proofs for finite field arithmetic; or: Can zero-knowledge be for free? In: Proceedings of the 18th Annual International Cryptology Conference. pp. 424–441. CRYPTO '98 (1998)
42. of Cryptographic Systems, e.E.B.: Measurements of hash functions, indexed by machine (2017), `https://bench.cr.yp.to/results-hash.html`
43. Feige, U., Goldwasser, S., Lovász, L., Safra, S., Szegedy, M.: Interactive proofs and the hardness of approximating cliques. Journal of the ACM **43**(2), 268–292 (1996), preliminary version in FOCS '91.
44. Gao, S., Mateer, T.: Additive fast fourier transforms over finite fields. IEEE Transactions on Information Theory **56**(12), 6265–6272 (2010)
45. Gennaro, R., Gentry, C., Parno, B., Raykova, M.: Quadratic span programs and succinct NIZKs without PCPs. In: Proceedings of the 32nd Annual International Conference on Theory and Application of Cryptographic Techniques. pp. 626–645. EUROCRYPT '13 (2013)
46. Gentry, C., Wichs, D.: Separating succinct non-interactive arguments from all falsifiable assumptions. In: Proceedings of the 43rd Annual ACM Symposium on Theory of Computing. pp. 99–108. STOC '11 (2011)
47. Goldreich, O., Håstad, J.: On the complexity of interactive proofs with bounded communication. Information Processing Letters **67**(4), 205–214 (1998)
48. Goldwasser, S., Kalai, Y.T., Rothblum, G.N.: Delegating computation: Interactive proofs for muggles. Journal of the ACM **62**(4), 27:1–27:64 (2015)
49. Goldwasser, S., Micali, S., Rackoff, C.: The knowledge complexity of interactive proof systems. SIAM Journal on Computing **18**(1), 186–208 (1989), preliminary version appeared in STOC '85.
50. Groth, J.: Short pairing-based non-interactive zero-knowledge arguments. In: Proceedings of the 16th International Conference on the Theory and Application of Cryptology and Information Security. pp. 321–340. ASIACRYPT '10 (2010)
51. Groth, J.: On the size of pairing-based non-interactive arguments. In: Proceedings of the 35th Annual International Conference on Theory and Applications of Cryptographic Techniques. pp. 305–326. EUROCRYPT '16 (2016)
52. Groth, J., Maller, M.: Snarky signatures: Minimal signatures of knowledge from simulation-extractable SNARKs. In: Proceedings of the 37th Annual International Cryptology Conference. pp. 581–612. CRYPTO '17 (2017)
53. Gueron, S.: Intel carry-less multiplication instruction and its usage for computing the GCM mode (2011), `https://software.intel.com/en-us/articles/intel-carry-less-multiplication-instruction-and-its-usage-for-computing-the-gcm-mode`
54. Ishai, Y., Kushilevitz, E., Ostrovsky, R.: Efficient arguments without short PCPs. In: Proceedings of the Twenty-Second Annual IEEE Conference on Computational Complexity. pp. 278–291. CCC '07 (2007)
55. Ishai, Y., Mahmoody, M., Sahai, A., Xiao, D.: On zero-knowledge PCPs: Limitations, simplifications, and applications (2015), available at `http://www.cs.virginia.edu/~mohammad/files/papers/ZKPCPs-Full.pdf`

56. Kalai, Y., Raz, R.: Interactive PCP. In: Proceedings of the 35th International Colloquium on Automata, Languages and Programming. pp. 536–547. ICALP '08 (2008)
57. Kilian, J.: A note on efficient zero-knowledge proofs and arguments. In: Proceedings of the 24th Annual ACM Symposium on Theory of Computing. pp. 723–732. STOC '92 (1992)
58. Lin, S., Al-Naffouri, T.Y., Han, Y.S.: FFT algorithm for binary extension finite fields and its application to Reed–Solomon codes. IEEE Transactions on Information Theory **62**(10), 5343–5358 (2016)
59. Lin, S., Chung, W.H., Han, Y.S.: Novel polynomial basis and its application to Reed–Solomon erasure codes. In: Proceedings of the 55th Annual IEEE Symposium on Foundations of Computer Science. pp. 316–325. FOCS '14 (2014)
60. Lipmaa, H.: Succinct non-interactive zero knowledge arguments from span programs and linear error-correcting codes. In: Proceedings of the 19th International Conference on the Theory and Application of Cryptology and Information Security. pp. 41–60. ASIACRYPT '13 (2013)
61. Lund, C., Fortnow, L., Karloff, H.J., Nisan, N.: Algebraic methods for interactive proof systems. Journal of the ACM **39**(4), 859–868 (1992)
62. Meir, O.: Combinatorial PCPs with short proofs. In: Proceedings of the 26th Annual IEEE Conference on Computational Complexity. CCC '12 (2012)
63. Micali, S.: Computationally sound proofs. SIAM Journal on Computing **30**(4), 1253–1298 (2000), preliminary version appeared in FOCS '94.
64. Montgomery, P.L.: Modular multiplication without trial division. Mathematics of Computation **44**(170), 519–521 (1985)
65. Nakamoto, S.: Bitcoin: a peer-to-peer electronic cash system (2009), `http://www.bitcoin.org/bitcoin.pdf`
66. NIST: Post-quantum cryptography (2016), `https://csrc.nist.gov/Projects/Post-Quantum-Cryptography`
67. Parno, B., Gentry, C., Howell, J., Raykova, M.: Pinocchio: Nearly practical verifiable computation. In: Proceedings of the 34th IEEE Symposium on Security and Privacy. pp. 238–252. Oakland '13 (2013)
68. Polishchuk, A., Spielman, D.A.: Nearly-linear size holographic proofs. In: Proceedings of the 26th Annual ACM Symposium on Theory of Computing. pp. 194–203. STOC '94 (1994)
69. Reingold, O., Rothblum, R., Rothblum, G.: Constant-round interactive proofs for delegating computation. In: Proceedings of the 48th ACM Symposium on the Theory of Computing. pp. 49–62. STOC '16 (2016)
70. SCIPR Lab: libsnark: a C++ library for zkSNARK proofs, `https://github.com/scipr-lab/libsnark`
71. Setty, S., Braun, B., Vu, V., Blumberg, A.J., Parno, B., Walfish, M.: Resolving the conflict between generality and plausibility in verified computation. In: Proceedings of the 8th EuoroSys Conference. pp. 71–84. EuroSys '13 (2013)
72. Shamir, A.: IP = PSPACE. Journal of the ACM **39**(4), 869–877 (1992)
73. Thaler, J.: Time-optimal interactive proofs for circuit evaluation. In: Proceedings of the 33rd Annual International Cryptology Conference. pp. 71–89. CRYPTO '13 (2013)
74. Thaler, J.: A note on the GKR protocol. `http://people.cs.georgetown.edu/jthaler/GKRNote.pdf` (2015)
75. Thaler, J., Roberts, M., Mitzenmacher, M., Pfister, H.: Verifiable computation with massively parallel interactive proofs. CoRR **abs/1202.1350** (2012)
76. Wahby, R.S., Howald, M., Garg, S.J., Shelat, A., Walfish, M.: Verifiable ASICs. In: Proceedings of the 37th IEEE Symposium on Security and Privacy. pp. 759–778. S&P '16 (2016)

77. Wahby, R.S., Ji, Y., Blumberg, A.J., Shelat, A., Thaler, J., Walfish, M., Wies, T.: Full accounting for verifiable outsourcing. In: Proceedings of the 24th ACM Conference on Computer and Communications Security. pp. 2071–2086. CCS '17 (2017)
78. Wahby, R.S., Setty, S., Ren, Z., Blumberg, A.J., Walfish, M.: Efficient RAM and control flow in verifiable outsourced computation. In: Proceedings of the 22nd Annual Network and Distributed System Security Symposium. NDSS '15 (2015)
79. Wahby, R.S., Tzialla, I., Shelat, A., Thaler, J., Walfish, M.: Doubly-efficient zksnarks without trusted setup. Cryptology ePrint Archive, Report 2017/1132 (2017)
80. Walfish, M., Blumberg, A.J.: Verifying computations without reexecuting them. Communications of the ACM **58**(2), 74–84 (Jan 2015)
81. Wee, H.: On round-efficient argument systems. In: Proceedings of the 32nd International Colloquium on Automata, Languages and Programming. pp. 140–152. ICALP '05 (2005)
82. Zhang, Y., Genkin, D., Katz, J., Papadopoulos, D., Papamanthou, C.: vSQL: Verifying arbitrary SQL queries over dynamic outsourced databases. In: Proceedings of the 38th IEEE Symposium on Security and Privacy. pp. 863–880. S&P '17 (2017)
83. Zhang, Y., Genkin, D., Katz, J., Papadopoulos, D., Papamanthou, C.: A zero-knowledge version of vsql. Cryptology ePrint Archive, Report 2017/1146 (2017)

Acknowledgments

We thank Alexander Chernyakhovsky and Tom Gur for helpful discussions, and Aleksejs Popovs for help in implementing parts of `libiop`. This work was supported in part by: the Ethics and Governance of Artificial Intelligence Fund; a Google Faculty Award; the Israel Science Foundation (grant 1501/14); the UC Berkeley Center for Long-Term Cybersecurity; the US-Israel Binational Science Foundation (grant 2015780); and donations from the Interchain Foundation and Qtum.