Brakedown: Linear-Time and Field-Agnostic SNARKs for R1CS

Alexander Golovnev[1(✉)], Jonathan Lee[2], Srinath Setty[3], Justin Thaler[1], and Riad S. Wahby[4]

[1] Georgetown University, Washington, D.C., USA
alexgolovnev@gmail.com
[2] Nanotronics, New York, USA
[3] Microsoft Research, Cambridge, USA
[4] Stanford University, Stanford, USA

Abstract. This paper introduces a SNARK called *Brakedown*. Brakedown targets R1CS, a popular NP-complete problem that generalizes circuit-satisfiability. It is the first built system that provides a *linear-time* prover, meaning the prover incurs $O(N)$ finite field operations to prove the satisfiability of an N-sized R1CS instance. Brakedown's prover is faster, both concretely and asymptotically, than prior SNARK implementations. It does not require a trusted setup and may be post-quantum secure. Furthermore, it is compatible with *arbitrary* finite fields of sufficient size; this property is new among built proof systems with sublinear proof sizes. To design Brakedown, we observe that recent work of Bootle, Chiesa, and Groth (BCG, TCC 2020) provides a polynomial commitment scheme that, when combined with the linear-time interactive proof system of Spartan (CRYPTO 2020), yields linear-time IOPs and SNARKs for R1CS (a similar theoretical result was previously established by BCG, but our approach is conceptually simpler, and crucial for achieving high-speed SNARKs). A core ingredient in the polynomial commitment scheme that we distill from BCG is a linear-time encodable code. Existing constructions of such codes are believed to be impractical. Nonetheless, we design and engineer a new one that is practical in our context.

We also implement a variant of Brakedown that uses Reed-Solomon codes instead of our linear-time encodable codes; we refer to this variant as *Shockwave*. Shockwave is *not* a linear-time SNARK, but it provides shorter proofs and lower verification times than Brakedown, and also provides a faster prover than prior plausibly post-quantum SNARKs.

1 Introduction

A SNARK [18,37,47,55] is a cryptographic primitive that enables a *prover* to prove to a *verifier* the knowledge of a satisfying witness to an NP statement by producing a proof π such that the size of π and the cost to verify it are both sub-linear in the size of the witness. Given their many applications, constructing SNARKs with excellent asymptotics and concrete efficiency is a highly active area of research. Still, one of the key bottlenecks preventing application of existing SNARKs to large NP statements is the prover's asymptotic and concrete

H. Handschuh and A. Lysyanskaya (Eds.): CRYPTO 2023, LNCS 14082, pp. 193–226, 2023.
https://doi.org/10.1007/978-3-031-38545-2_7

cost. This has limited the use of SNARKs to practical applications in which NP statements of interest are relatively small (for example, cryptocurrencies).

As with much of the literature on SNARKs, we focus on rank-1 constraint satisfiability (R1CS) over a finite field $\mathbb{F}$, an NP-complete problem that generalizes arithmetic circuit satisfiability. An R1CS instance comprises a set of M constraints, with a vector w over $\mathbb{F}$ said to *satisfy* the instance if it satisfies all M constraints. The term "rank-1" means that the constraints should have a specific form. Specifically, each constraint asserts that the product of two specified linear combinations of the entries of w equals a third linear combination of those entries. See Definition 1 for details. R1CS is amenable to probabilistic checking and is highly expressive. For example, in theory, any nondeterministic random access machine running in time T can be transformed into an R1CS instance of size "close" to T. In practice, there exist efficient transformations and compiler toolchains to transform applications of interest to R1CS [13,25,50,58,59,64,66,73].

Our focus in this work is designing SNARKs for R1CS with the fastest possible prover. We also wish for the SNARK to be *transparent* (or be without a trusted setup): there should be no need to run a complex multi-party computation to generate a so-called *structured reference string* that is needed for proof generation.

Furthermore, we desire a verifier that runs in time sub-linear in the size of the R1CS instance. Since the verifier must at least read the statement that is being proven, we allow a one-time public preprocessing phase for general (unstructured) R1CS instances. In this phase, the verifier computes a *computation commitment*, a cryptographic commitment to the structure of a circuit or R1CS instance [63]. (For "structured" computations, our SNARKs, like several prior works, can avoid this pre-processing phase.) After the pre-processing phase, the verifier must run in time sub-linear in the size of the R1CS instance. Furthermore, the pre-processing phase should be at least as efficient as the SNARK prover. Subsequent works to Spartan [63] refer to such public preprocessing to achieve fast verification as leveraging *holography* [21,31,32].

A second focus of our work is designing SNARKs that can operate over arbitrary (sufficiently large) finite fields. Prior SNARKs apply over fields that are "discrete-log friendly"[1] or "FFT-friendly", or otherwise require one or many multiplicative or additive subgroups of specified sizes. Yet many cryptographic applications naturally work over fields that do not satisfy these properties. Examples include proofs regarding encryption or signature schemes that themselves work over fields that do not satisfy the properties needed by the SNARK. Indeed, most practically relevant elliptic curve groups are defined over fields that are not FFT-friendly. Even in applications where SNARK designers do have flexibility in field choice, field size restrictions can still create engineering challenges or

[1] It is possible to construct elliptic curves with specified group order [26], which suffices for many discrete log–based SNARKs. Unfortunately, the most efficient elliptic curve implementations are tailored to specific curves—so using a newly constructed curve may entail a performance or engineering cost.

inconveniences, as well as performance overheads. For example, they may limit the size of R1CS statements that can be handled over the chosen field, or force instance sizes to be padded to a length corresponding to the size of a subgroup.

In this work we design transparent SNARKs that asymptotically have the fastest possible prover, may be post-quantum secure, and work over arbitrary (sufficiently large) finite fields.[2] We refer to this latter property as being *field-agnostic*, and to the best of our knowledge, it is new amongst implemented arguments with sublinear proof size and even *quasi*linear runtime. We optimize and implement our new SNARKs, and demonstrate the fastest prover performance in the SNARK literature (even compared to SNARKs that require FFT-friendly or discrete-log–friendly fields).

Formalizing "fastest possible" Provers. How fast can we hope for the prover in a SNARK to be? Letting N denote the size of the R1CS or arithmetic-circuit-satisfiability instance over an *arbitrary* finite field $\mathbb{F}$, a lower bound on the prover's runtime is N operations in $\mathbb{F}$. Here, the size of an arithmetic-circuit-satisfiability instance is the number of gates in the circuit. The size of an R1CS instance of the form $Az \circ Bz = Cz$ is the number of non-zero entries in A, B, C, where $\circ$ denotes the Hadamard (entry-wise) product. This is because any prover that knows a witness w for the instance has to at least convince *itself* (much less the verifier) that w is valid. We refer to this procedure as *native evaluation* of the instance. So the natural goal, roughly speaking, is to achieve a SNARK prover that is only a constant factor slower than native evaluation. Such a prover is said to run in *linear-time*.

Achieving a linear-time prover may sound like a simple and well-defined goal, but it is in fact subtle to formalize, because one must be precise about what operations can be performed in one "time-step", as well as the soundness error achieved and the choice of the finite field.

In known SNARKs, the bottleneck for the prover (both asymptotically and concretely) is typically one or more of the following operations: (1) Performing an FFT over a vector of length $O(N)$. (2) Building a Merkle-hash tree over a vector consisting of $O(N)$ elements of $\mathbb{F}$. (3) Performing a multiexponentiation of size $O(N)$ in a (multiplicative) cryptographic group $\mathbb{G}$. In this case, the field $\mathbb{F}$ is of prime order p and $\mathbb{G}$ is typically an elliptic curve group (or subgroup) of order p. A multiexponentiation of size N in $\mathbb{G}$ refers to a product of N exponentiations, i.e., $\prod_{i=1}^{N} g_i^{c_i}$, where each $g_i \in \mathbb{G}$ and each $c_i \in \{0, \ldots, p-1\}$.

Should any of these operations count as "linear-time"?

FFTs. An FFT of length $\Theta(N)$ over $\mathbb{F}$ should *not* count as linear-time, because the fastest known algorithms require $\Theta(N \log N)$ operations over $\mathbb{F}$, which is a $\log N$ factor, rather than a constant factor, larger than native evaluation.

However, the remaining operations are trickier to render judgment upon, because they do not refer to field operations.

[2] For our SNARKs, a field of size $\exp(\lambda)$ is sufficient to achieve λ bits of security with a linear-time prover. More generally, our SNARK can work over any field $\mathbb{F}$ of size $|\mathbb{F}| \geq \Omega(N)$ with a prover runtime that is superlinear by a factor of $O(\lambda/\log|\mathbb{F}|)$, where N denotes instance size.

Merkle-Hashing. Build a Merkle tree over a vector of $O(N)$ elements of $\mathbb{F}$, computing $O(N)$ cryptographic hashes is necessary and sufficient, assuming the hash function takes as input $O(1)$ elements of $\mathbb{F}$. However, this is only "linear-time" if hashing $O(N)$ elements of $\mathbb{F}$ can be done in time comparable to $O(N)$ operations over $\mathbb{F}$. It is not clear whether or not applying a standard hash function such as SHA-256 to hash a field element should be considered comparable to performing a single field operation.

Theoretical work of Bootle et al. [20] sidesteps this issue by observing that (assuming the intractability of certain lattice problems over $\mathbf{F}_2$, specifically finding a low-Hamming vector in the kernel of a sparse matrix), a collision-resistant hash family of Applebaum et al. [5] is capable of hashing strings consisting of $k \gg \lambda$ bits in $O(k)$ bit operations, with security parameter λ. This means that a vector of $O(N)$ elements of $\mathbb{F}$ can be Merkle-hashed in $O(N \log |\mathbb{F}|)$ bit operations, which Bootle et al. [20] consider comparable to the cost of $O(N)$ operations in $\mathbb{F}$. The aforementioned hash functions appear to be of primarily theoretical interest because they can be orders of magnitude slower than standard hash functions (e.g., SHA-256). Hence, in this paper our implementations make use of standard hash functions, and with this choice, Merkle-hashing is not the concrete bottleneck in our implementations. Accordingly, and to simplify discussion, we consider our implemented Merkle-hashing procedure to be linear-time, even if this may not be strictly justified from a theoretical perspective.

Multiexponentiation. Pippenger's algorithm [60] (see also [16,44]) can perform an $O(N)$-sized multiexponentiation in a group $\mathbb{G}$ of size $\sim 2^\lambda$ by performing $O(N \cdot \lambda / \log(N \cdot \lambda))$ group operations (i.e., group multiplications). Typically, one thinks of the security parameter λ as $\omega(\log N)$ (so that 2^λ is superpolynomial in N, ensuring the intractability of problems such as discrete logarithm in $\mathbb{G}$), and so $O(N \cdot \lambda / \log(N \cdot \lambda))$ group operations is considered $\omega(N)$ group operations. Each group operation is at least as expensive (in fact, several times slower) than a field operation—typically, an operation in the elliptic-curve group $\mathbb{G}$ requires performing a constant number of field operations within a field that is of similar size to, but different than, then prime-order field $\mathbb{F}$ over which the circuit or R1CS instance is defined. Hence, we do *not* consider this to be linear time.

However, note that *for a fixed value of the security parameter* λ, the cost of a multiexponentiation of size N performed using Pippenger's algorithm scales only linearly (in fact, *sub*linearly) with N. That is, Pippenger's algorithm incurs $\Theta(N \cdot (\lambda / \log(N\lambda))) = \Theta_\lambda(N / \log N)$ group operations and in turn this cost is comparable up to a constant factor to the same number of operations over a field of size $\exp(\lambda)$. In practice, protocol designers fix a cryptographic group (and hence fix λ), and then apply the resulting protocol to R1CS instances of varying sizes N. For this reason, systems (e.g., Spartan [63]) whose dominant prover cost is a multiexponentiation of size N will scale (sub-)linearly as a function of N. Specifically, in the experimental results [63], Spartan's prover exhibits the behavior of a linear-time prover (as the cost of native evaluation of the instance also scales linearly as a function of N). Nonetheless, since λ should be thought of as $\omega(\log N)$, we do not consider a multiexponentiation of size N to be a linear-time operation.

In summary, we do *not* consider FFTs and multiexponentiations of size $O(N)$ to be linear-time operations, but *do* consider Merkle-hashing of vectors of size $O(N)$ to be linear-time.

Closely Related Work. We cover additional related work in the full version of this paper [40]. Building on Bootle et al. [20], Bootle, Chiesa, and Groth [21] give an *interactive oracle proof* (IOP) [15] with *constant* soundness error, in which the prover's work is $O(N)$ finite field operations for an N-sized R1CS instance over any finite field of size $\Omega(N)$. Here, an interactive oracle proof (IOP) [15,61] is a generalization of an interactive proof, where in each round, the prover sends a string as an oracle, and the verifier may read one or more entries in the oracle. To achieve soundness error that is exponentially small in the security parameter λ in the IOP of [21], one must restrict to R1CS instances over a "sufficiently large" finite field i.e., where $|\mathbb{F}| = 2^{\Theta(\lambda)}$, or else sacrifice the linear-time prover.

Applying standard transformations to their IOP, one can obtain a SNARG in the random oracle model with similar prover costs, or an interactive argument assuming linear-time computable cryptographic hash functions [5]. Unlike prior SNARGs (even those with a *quasi*-linear time prover), the resulting protocol does not require the field to be FFT-friendly nor discrete-log friendly.

Their IOP construction does not achieve zero-knowledge nor polylogarithmic proofs and verification times (the proof sizes and verification times are $O_\lambda(N^{1/t})$, where t is a constant, and not $O_\lambda(\log N)$ or $O_\lambda(1)$). Bootle, Chiesa, and Liu [23] address these issues by achieving zero-knowledge as well as polylogarithmic proof sizes and verification times (a more detailed discussion of the relationship between our results and those of [23] is in the full version of this paper [40]). Both [21,23] are theoretical in nature; they do not implement their schemes nor report performance results.

There is also very recent work related to our goal of working over arbitrary finite fields. Ben-Sasson et al. [11,12] improve the efficiency of FFT-like algorithms that apply over fields with no smooth order root of unity, by a factor of $\exp(\log^* N)$. An explicit motivation for their work is to improve the efficiency of known SNARKs that perform FFTs (e.g., Fractal [32]) when operating over "non-FFT-friendly" fields. These results do not eliminate the superlinearity of the prover's runtime in their target SNARKs. The algorithms given in [11,12] also perform significant pre-computation that is field-specific and have not to date yielded implemented SNARKs. We seek (and achieve) high-speed SNARKs that require only black-box access to the addition and multiplication operations of the field, with the only additional information required being a lower bound on the field size to ensure soundness.

In summary, prior works leave open the problem of achieving concretely efficient SNARKs that support arbitrary (sufficiently large) finite fields, much less one with a linear-time prover.

1.1 Results and Contributions

We address the above problems with *Brakedown*, a new linear-time field-agnostic SNARK that we design, implement, optimize, and experimentally evaluate.

Concretely, Brakedown achieves the fastest SNARK prover in the literature, even over fields to which prior SNARKs apply. We also implement and evaluate *Shockwave*, a variant of Brakedown that reduces proof sizes and verification times at the cost of sacrificing a linear-time prover, but nonetheless provides a faster prover than prior plausibly post-quantum SNARKs. Brakedown and Shockwave are unconditionally secure in the random oracle model.

SNARK Design Background. Modern SNARKs work by combining a type of interactive protocol called a *polynomial IOP* [28] with a cryptographic primitive called a *polynomial commitment scheme* [45]. The combination yields succinct *interactive* argument, which can then be rendered non-interactive via the Fiat-Shamir transformation [35], yielding a SNARK.

Roughly, a polynomial IOP is an interactive protocol where, in one or more rounds, the prover "sends" to the verifier a very large polynomial q. Because q is so large, one does not wish for the verifier to read a complete description of q. Instead, the verifier only "queries" q at one point (or a handful of points). This means that the only information the verifier needs about q to check that the prover is behaving honestly is one (or a few) evaluations of q.

In turn, a polynomial commitment scheme enables an untrusted prover to succinctly *commit* to a polynomial q, and later provide to the verifier any evaluation $q(r)$ for a point r chosen by the verifier, along with a proof that the returned value is indeed consistent with the committed polynomial.

Essentially, a polynomial commitment scheme is exactly the cryptographic primitive that one needs to obtain a succinct argument from a polynomial IOP. Rather than having the prover send a large polynomial q to the verifier as in the polynomial IOP, the argument system prover cryptographically commits to q and later reveals any evaluations of q required by the verifier to perform its checks.

Design of Our Linear-Time SNARK. We first distill from [21] a polynomial commitment scheme with a linear-time commitment procedure, and show that it satisfies extractability, a key property required in the context of SNARKs (the commitment scheme itself is little more than a rephrasing of the results in [21], though [21] did not analyze extractability). This improves over the prior state-of-the-art polynomial commitment schemes [28,45,49,65,74,78,79] by offering the first in which the time to commit to a polynomial is linear in the size of the polynomial. We focus on multilinear polynomials over the Lagrange basis, but the scheme generalizes to many other types of polynomials such as univariate polynomials over the standard monomial basis (see e.g., [49]).

To obtain linear-time SNARKs for R1CS, we first make explicit a polynomial IOP for R1CS from Spartan [63] and then use our new linear-time polynomial commitment scheme in conjunction with prior compilers [28,63] to transform it into a SNARK for R1CS.

A New and Concretely Fast Linear-Time Encodable Code. A major component in the linear-time polynomial commitment scheme that we distill from [21] is a linear-time encodable linear code. Unfortunately, to the best of our knowledge, existing linear-time encodable codes are highly impractical. We therefore design a new linear-time encodable code that is concretely fast in our context. Our code builds on classic results [34,36,67], but designing this code is involved and represents a significant technical contribution. We achieve a fast linear code that works over any (sufficiently large) field by leveraging the following four observations: (1) In our setting, to achieve sublinear sized proofs, it is sufficient for the code to achieve relative Hamming distance only a small constant, rather than very close to 1 (higher minimum distance would improve Brakedown's proof length by a constant factor, but would not meaningfully reduce the prover time); (2) Efficient decoding is not necessary for reasons elaborated upon below; (3) We can (and indeed want to) work over large fields, say, of size at least 2^{127}; and (4) We can use randomized constructions instead of deterministic constructions of pseudorandom objects, so long as the probability that the construction fails to satisfy the necessary distance properties is cryptographically small (e.g., $\leq 2^{-100}$).

Observation (2) holds for the following two reasons: (1) the prover and verifier only execute the code's encoding procedure (this observation also appears in prior work [21]); (2) we describe an efficient extractor for our polynomial commitment scheme that does not invoke the code's decoding procedure. Hence, efficient decoding is not even needed to establish that our SNARK is knowledge-sound.

Observations (1)–(4) together allow us to strip out much of the complexity of prior constructions. For example, Spielman's celebrated work [67] is focused on achieving both linear-time encoding and decoding, while Druk and Ishai [34] focus on improving the minimum distance of Spielman's code. On top of this, we further optimize and simplify the code construction, and provide a detailed, quantitative analysis to show that the probability our code fails to achieve the necessary minimum distance is cryptographically small.

Implementation, Optimization, and Experimental Results. We implement the aforementioned linear-time SNARK, yielding a system we call Brakedown. Because our linear-time code works over any (sufficiently large) field, and the polynomial IOP from Spartan does as well, Brakedown is field-agnostic. This is the first built SNARK to achieve this property. It is also the first built system with a linear-time prover and sub-linear proof sizes and verification times.

We also implement Shockwave, a variant of Brakedown that uses Reed-Solomon codes instead of our fast linear-time code. Since Shockwave uses Reed-Solomon codes, it is not a linear-time SNARK and requires an FFT-friendly finite field, but it provides concretely shorter proofs and lower verification times than Brakedown and is faster than prior plausibly post-quantum secure SNARKs.

Both Shockwave and Brakedown contain simple but crucial concrete optimizations to the polynomial commitment scheme to reduce proof sizes. Neither Shockwave's nor Brakedown's implementations are currently zero-knowledge. However, Shockwave can be rendered zero-knowledge using standard techniques with minimal overhead [4,30,76]. Brakedown could be rendered zero-knowledge while maintaining linear prover time by using one layer of recursive composition with zkShockwave (or another zkSNARK). Indeed, subsequent work, called Orion [77], uses Virgo [78] to prove in zero-knowledge the knowledge of valid proofs produced by (a variant of) Brakedown. It is also plausible that Brakedown could be rendered zero-knowledge more directly using techniques from [23].

In terms of experimental results, Brakedown achieves a faster prover than all prior SNARKs for R1CS. Its primary downside is that its proofs are on the larger side, but they are still far smaller than the size of the NP-witness for R1CS instance sizes beyond several million constraints. Shockwave reduces Brakedown's proof sizes and verification times by about a factor of 6×, at the cost of a slower prover (both asymptotically and concretely). Nonetheless, Shockwave already features a concretely faster prover than prior plausibly post-quantum SNARKs. Furthermore, although Shockwave's proof sizes are somewhat larger than most prior schemes with sublinear proof size, they are surprisingly competitive with prior post-quantum schemes such as Fractal [32] and Aurora [14] that have lower *asymptotic* proof size (polylog(N) rather than $\Theta_\lambda(\sqrt{N})$). Its verification times are competitive with discrete-logarithm based schemes, and in fact superior to prior plausibly post-quantum SNARKs.

Public Parameter Generation. The public parameters of Brakedown include a description of the encoding procedure of our error-correcting code. This involves randomly generating certain sparse matrices (we provide details of this in the full version of the paper [40]). Our implementation generates the matrices deterministically using a cryptographic PRG with a public, fixed seed, which could be chosen in a "nothing-up-my-sleeve" way (e.g., as in Bulletproofs [27]). Generating the matrices is concretely fast: our implementation takes under 700 milliseconds to sample parameters suitable for encoding inputs of length 2^{20}, and 22 seconds for encoding inputs of length 2^{25}. The latter setting is suitable for committing to polynomials of degree over 2^{40}, and for giving SNARKs for R1CS instances with roughly 2^{40} constraints. Note that any party acting as the prover or verifier in Brakedown need only generate these matrices once, no matter how many times the SNARK is used.

Subsequent Work on Linear-Time Provers. Xie et al. [77] improve the concrete parameters of the error-correcting code underlying Brakedown. They also compose Brakedown with a different SNARK called Virgo [78] that requires an FFT-friendly field but has smaller proofs. This asymptotically reduces the proof size from $\Theta_\lambda(\sqrt{N})$ to $\Theta_\lambda(\log^2 N)$. The resulting implementation, called Orion, requires an FFT-friendly field, but has substantially smaller proofs than Brakedown, and a slightly faster prover due to the improved code parameters. Orion+ [29] improves Brakedown (this work) and the work of Xie et al. [77] by providing proofs of ≈10 KB at the cost of requiring a (universal) trusted setup and giving up plausible post-quantum security. Vortex [7] builds on Brakedown and uses lattice-based hash functions for improved recursion capabilities.

Recent theoretical works have obtained interactive arguments with constant soundness error and a linear-time prover even over small fields [22,62].

2 Preliminaries

We use $\mathbb{F}$ to denote a finite field, λ to denote the security parameter, and $\mathsf{negl}(()\lambda)$ to denote a negligible function in λ. Unless we specify otherwise, $|\mathbb{F}| = 2^{\Theta(\lambda)}$.

Polynomials. We recall a few basic facts about polynomials. Detailed treatment of these facts can be found elsewhere [69].

– A polynomial over $\mathbb{F}$ is an expression consisting of a sum of *monomials* where each monomial is the product of a constant and powers of one or more variables (which take values from $\mathbb{F}$); all arithmetic is performed over $\mathbb{F}$.
– The degree of a monomial is the sum of the exponents of variables in the monomial; the (total) degree of a polynomial g is the maximum degree of any monomial in g. Also, the degree of a polynomial g in a particular variable x_i is the maximum exponent that x_i takes in any of the monomials in g.
– A *multivariate* polynomial is a polynomial with more than one variable; otherwise it is called a *univariate* polynomial. A multivariate polynomial is called a *multilinear* polynomial if the degree of the polynomial in each variable is at most one.

Rank-1 Constraint Satisfiability (R1CS)

Definition 1. *An R1CS instance is a tuple* $(\mathbb{F}, A, B, C, M, N, \mathsf{io})$*, where* $A, B, C \in \mathbb{F}^{M\times M}$*,* $M \geq |\mathsf{io}| + 1$*,* io *denotes the public input and output, and there are at most* $N = \Omega(M)$ *non-zero entries in each matrix.*

We denote the set of R1CS (instance, witness) pairs as $\mathcal{R}_{\text{R1CS}}$, defined as: $\{\langle(\mathbb{F}, A, B, C, \mathsf{io}, M, N), w\rangle : A\cdot(w, 1, \mathsf{io}) \circ B\cdot(w, 1, \mathsf{io}) = C\cdot(w, 1, \mathsf{io})\}$.

In the rest of the paper, WLOG, we assume that M and N are powers of 2, and that $M = |\mathsf{io}| + 1$. Throughout this paper, all logarithms are to base 2.

SNARKs. We adapt the definition provided in [48].

Definition 2. *Consider a relation* $\mathcal{R}$ *over public parameters, structure, instance, and witness tuples. A non-interactive argument of knowledge for* $\mathcal{R}$ *consists of PPT algorithms* $(\mathcal{G}, \mathcal{P}, \mathcal{V})$ *and deterministic* $\mathcal{K}$*, denoting the generator, the prover, the verifier and the encoder respectively with the following interface.*

– $\mathcal{G}(1^\lambda) \rightarrow \mathsf{pp}$*: On input security parameter* λ*, samples public parameters* pp*.*
– $\mathcal{K}(\mathsf{pp}, \mathsf{s}) \rightarrow (\mathsf{pk}, \mathsf{vk})$*: On input structure* s*, representing common structure among instances, outputs the prover key* pk *and verifier key* vk*.*
– $\mathcal{P}(\mathsf{pk}, u, w) \rightarrow \pi$*: On input instance* u *and witness* w*, outputs a proof* π *proving that* $(\mathsf{pp}, \mathsf{s}, u, w) \in \mathcal{R}$*.*
– $\mathcal{V}(\mathsf{vk}, u, \pi) \rightarrow \{0, 1\}$*: On input the verifier key* vk*, instance* u*, and a proof* π*, outputs 1 if the instance is accepting and 0 otherwise.*

A non-interactive argument of knowledge satisfies completeness if for any PPT adversary $\mathcal{A}$

$$\Pr\left[\mathcal{V}(\mathsf{vk}, u, \pi) = 1 \;\middle|\; \begin{array}{l} \mathsf{pp} \leftarrow \mathcal{G}(1^\lambda), \\ (\mathsf{s}, (u, w)) \leftarrow \mathcal{A}(\mathsf{pp}), \\ (\mathsf{pp}, \mathsf{s}, u, w) \in \mathcal{R}, \\ (\mathsf{pk}, \mathsf{vk}) \leftarrow \mathcal{K}(\mathsf{pp}, \mathsf{s}), \\ \pi \leftarrow \mathcal{P}(\mathsf{pk}, u, w) \end{array}\right] = 1.$$

pagebreak A non-interactive argument of knowledge satisfies knowledge soundness if for all PPT adversaries $\mathcal{A}$ there exists a PPT extractor $\mathcal{E}$ such that for all randomness ρ

$$\Pr\left[\begin{array}{c|l} \mathcal{V}(\mathsf{vk}, u, \pi) = 1, & \mathsf{pp} \leftarrow \mathcal{G}(1^\lambda), \\ (\mathsf{pp}, \mathsf{s}, u, w) \notin \mathcal{R} & (\mathsf{s}, u, \pi) \leftarrow \mathcal{A}(\mathsf{pp}; \rho), \\ & (\mathsf{pk}, \mathsf{vk}) \leftarrow \mathcal{K}(\mathsf{pp}, \mathsf{s}), \\ & w \leftarrow \mathcal{E}(\mathsf{pp}, \rho) \end{array}\right] = \mathsf{negl}(\lambda).$$

A non-interactive argument of knowledge is succinct if the size of the proof π and the time to verify it are at most polylogarithmic in the size of the statement proven, where a statement includes both the structure and the instance.

Remark 1. In this paper, we consider an argument system to be succinct as long as the proof sizes and verification times are sublinear in the size of the statement proven. We accept this weakening as proofs produced by such proof systems can be shortened (both asymptotically and concretely) without substantial overheads using depth-1 recursion (e.g., see a subsequent work called Orion [77]).

Polynomial Commitment Scheme. We adapt the definition from [28]. A polynomial commitment scheme for multilinear polynomials is a tuple of four protocols $\mathsf{PC} = (\mathsf{Gen}, \mathsf{Commit}, \mathsf{Open}, \mathsf{Eval})$:

– $pp \leftarrow \mathsf{Gen}(1^\lambda, \mu)$: takes as input μ (the number of variables in a multilinear polynomial); produces public parameters pp.
– $\mathcal{C} \leftarrow \mathsf{Commit}(pp, \mathcal{G})$: takes as input a μ-variate multilinear polynomial over a finite field $\mathcal{G} \in \mathbb{F}[\mu]$; produces a commitment $\mathcal{C}$.
– $b \leftarrow \mathsf{Open}(pp, \mathcal{C}, \mathcal{G})$: verifies the opening of commitment $\mathcal{C}$ to the μ-variate multilinear polynomial $\mathcal{G} \in \mathbb{F}[\mu]$; outputs $b \in \{0, 1\}$.
– $b \leftarrow \mathsf{Eval}(pp, \mathcal{C}, r, v, \mu, \mathcal{G})$ is a protocol between a PPT prover $\mathcal{P}$ and verifier $\mathcal{V}$. Both $\mathcal{V}$ and $\mathcal{P}$ hold a commitment $\mathcal{C}$, the number of variables μ, a scalar $v \in \mathbb{F}$, and $r \in \mathbb{F}^\mu$. $\mathcal{P}$ additionally knows a μ-variate multilinear polynomial $\mathcal{G} \in \mathbb{F}[\mu]$. $\mathcal{P}$ attempts to convince $\mathcal{V}$ that $\mathcal{G}(r) = v$. At the end of the protocol, $\mathcal{V}$ outputs $b \in \{0, 1\}$.

Definition 3. *A tuple of four protocols* $(\mathsf{Gen}, \mathit{Commit}, \mathit{Open}, \mathit{Eval})$ *is an extractable polynomial commitment scheme for multilinear polynomials over a finite field $\mathbb{F}$ if the following conditions hold.*

– ***Completeness.*** *For any μ-variate multilinear polynomial $\mathcal{G} \in \mathbb{F}[\mu]$,*

$$\Pr\left\{\begin{array}{c} pp \leftarrow \mathsf{Gen}(1^\lambda, \mu);\ \mathcal{C} \leftarrow \mathit{Commit}(pp, \mathcal{G}): \\ \mathit{Eval}(pp, \mathcal{C}, r, v, \mu, \mathcal{G}) = 1 \wedge v = \mathcal{G}(r) \end{array}\right\} \geq 1 - \mathsf{negl}(\lambda)$$

– ***Binding.*** *For any PPT adversary $\mathcal{A}$, size parameter $\mu \geq 1$,*

$$\Pr\left\{\begin{array}{c} pp \leftarrow \mathsf{Gen}(1^\lambda, m);\ (\mathcal{C}, \mathcal{G}_0, \mathcal{G}_1) = \mathcal{A}(pp); \\ b_0 \leftarrow \mathit{Open}(pp, \mathcal{C}, \mathcal{G}_0);\ b_1 \leftarrow \mathit{Open}(pp, \mathcal{C}, \mathcal{G}_1): \\ b_0 = b_1 \neq 0 \wedge \mathcal{G}_0 \neq \mathcal{G}_1 \end{array}\right\} \leq \mathsf{negl}(\lambda)$$

- ***Knowledge Soundness.*** *Eval is a succinct argument of knowledge for the following NP relation given $pp \leftarrow \mathsf{Gen}(1^\lambda, \mu)$.*

$$\mathcal{R}_{\mathsf{Eval}}(pp) = \{\langle(\mathcal{C}, r, v), (\mathcal{G})\rangle : \mathcal{G} \in \mathbb{F}[\mu] \wedge \mathcal{G}(r) = v \wedge \mathsf{Open}(pp, \mathcal{C}, \mathcal{G}) = 1\}$$

3 Linear-Time Polynomial Commitments

We distill from Bootle et al. [21] a result establishing the existence of a linear-time commitment scheme for multilinear polynomials over the Lagrange basis with proofs of size $O(N^{1/t})$ for any desired integer constant $t > 0$. Note that this result is implicit in their work.

We then explicitly describe the linear-time polynomial commitment scheme for the case when the parameter $t = 2$. We additionally describe optimizations and prove that the scheme satisfies knowledge soundness.

A General Result Distilled from Bootle et al. [21].

Theorem 1. *For security parameter λ and a positive integer t, given a hash function that can compute a Merkle-hash of N elements of $\mathbb{F}$ with the same time complexity as $O(N)$ $\mathbb{F}$-ops, there exists a linear-time polynomial commitment scheme for multilinear polynomials. Specifically, there exists an algorithm that, given as input the coefficient vector of an ℓ-variate multilinear polynomial over $\mathbb{F}$ over the Lagrange basis, with $N = 2^\ell$, commits to the polynomial, where:*

- *the size of the commitment is $O_\lambda(1)$; and*
- *the running time of the commit algorithm is $O(N)$ operations over $\mathbb{F}$.*

Furthermore, there exists a non-interactive argument of knowledge in the random oracle model to prove the correct evaluation of a committed polynomial with the following parameters:

- *the prover's running time is $O(N)$ operations over $\mathbb{F}$;*
- *the verifier's running time is $O_\lambda(N^{1/t})$ operations over $\mathbb{F}$; and*
- *the proof size is $O_\lambda(N^{1/t})$.*

A proof of this theorem is in the full version of this paper [40].

3.1 Polynomial Commitments for $t = 2$

Notation. g is a multilinear polynomial with n coefficients. We assume for simplicity that $n = m^2$ for some integer m. Let u denote the coefficient vector of g in the Lagrange basis (equivalently, u is the vector of all evaluations of g over inputs in $\{0,1\}^{\log n}$). Recalling that $[m] = \{1, \ldots, m\}$, we can naturally index entries of u by elements of the set $[m]^2$. It is well known that for any input r to g there exist vectors $q_1, q_2 \in \mathbb{F}^m$ such that $g(r) = \langle(q_1 \otimes q_2), u\rangle$.

For each $i \in [m]$, let us view u as an $m \times m$ matrix, and let u_i denote the ith row of this matrix, i.e., $u_i = \{u_{i,j}\}_{j \in [m]}$.

Let $N = \rho^{-1} \cdot m$, and let $\mathsf{Enc}\colon \mathbb{F}^m \to \mathbb{F}^N$ denote the encoding function of a linear code with constant rate $\rho > 0$ and constant relative distance $\gamma > 0$. We assume that Enc runs in time proportional to that required to perform $O(N)$ operations over $\mathbb{F}$. We assume for simplicity that Enc is systematic, since explicit systematic codes with the properties we require are known [67].

Commitment Phase. Let $\hat{u} = \{\mathsf{Enc}(u_i)\}_{i \in [m]} \in \left(\mathbb{F}^N\right)^m$ denote the vector obtained by encoding each row of u. In the IOP setting, the commitment to u is just the vector $\hat{u}$, i.e., the prover sends $\hat{u}$ to the verifier, and the verifier is given point query access to $\hat{u}$. In the derived polynomial commitment scheme in the plain or random oracle model, the commitment to u will be the Merkle-hash of the vector $\hat{u}$. As with u, we may view $\hat{u}$ as a matrix, with $\hat{u}_i \in \mathbb{F}^N$ denoting the ith row of $\hat{u}$ for $i \in [m]$.

Testing Phase. Upon receiving the commitment message, the IOP verifier will interactively test it to confirm that each "row" of u is indeed (close to) a codeword of Enc. We describe this process as occurring in a separate "testing phase" so as to keep the commitment size constant in the plain or random oracle models. In practice, the testing phase can occur during the commit phase, during the evaluation phase, or sometime in between the two.

The verifier sends the prover a random vector $r \in \mathbb{F}^m$, and the prover sends a vector $u' \in \mathbb{F}^m$ claimed to equal the random linear combination of the m rows of u, in which the coefficients of the linear combination are given by r. The verifier reads u' in its entirety.

Next, the verifier tests u' for consistency with $\hat{u}$. That is, the verifier will pick $\ell = \Theta(\lambda)$ random entries of the codeword $\mathsf{Enc}(u') \in \mathbb{F}^N$ and confirm that $\mathsf{Enc}(u')$ is consistent with $v \in \mathbb{F}^N$ at those entries, where v is:

$$\sum_{i=1}^{m} r_i \hat{u}_i \in \mathbb{F}^N. \tag{1}$$

Observe that, by definition of v (Eq. (1)), any individual entry v_j of v can be learned by querying m entries of $\hat{u}$ (we refer to these m entries as the "j'th column" of $\hat{u}$). Meanwhile, since the verifier reads u' in its entirety; $\mathcal{V}$ can compute $\mathsf{Enc}(u')_j$ for all desired $j \in [N]$ in $O(m)$ time.

Evaluation Phase. Let $q_1, q_2 \in \mathbb{F}^m$ be such that $g(r) = \langle (q_1 \otimes q_2), u \rangle$. The evaluation phase is identical to the testing phase, except that r is replaced with q_1 (and the verifier uses fresh randomness to choose the sets of coordinates used for consistency testing). Let $u'' \in \mathbb{F}^m$ denote the vector that the prover sends in this phase, which is claimed to equal $\sum_{i=1}^m q_{1,i} \cdot u_i$. If the prover is honest, then u'' satisfies $\langle u'', q_2 \rangle = \langle (q_1 \otimes q_2), u \rangle$. Hence, if the verifier's consistency tests all pass in the testing and evaluation phases, the verifier outputs $\langle u'', q_2 \rangle$ as $g(r)$.

Concrete Optimizations to the Commitment Scheme. We discuss optimizations to reduce proof sizes in the testing and evaluation phases by large constant factors without affecting the correctness guarantees of the commitment scheme.

Description of polynomial commitment in the language of IOPs. Following standard transformations [15,47,55,70], in the actual polynomial commitment scheme, vectors sent by the prover in the IOP may be replaced with a Merkle-commitment to that vector, and each query the verifier makes to a vector is answered by the prover along with Merkle-tree authentication path for the answer. Each phase of the scheme can be rendered non-interactive using the Fiat-Shamir transformation [35].

Commit phase.

- $\mathcal{P} \to \mathcal{V}$: a vector $\hat{u} = (\hat{u}_1, \ldots, \hat{u}_m) \in \left(\mathbb{F}^N\right)^m$. If $\mathcal{P}$ is honest, each "row" $\hat{u}_i$ of $\hat{u}$ contains a codeword in Enc.

Testing phase.

- $\mathcal{V} \to \mathcal{P}$: a random vector $r \in \mathbb{F}^m$.
- $\mathcal{P} \to \mathcal{V}$ sends a vector $u' \in \mathbb{F}^m$ claimed to equal $v = \sum_{i=1}^m r_i \cdot u_i \in \mathbb{F}^m$.
- //Now $\mathcal{V}$ probabilistically checks consistency between $\hat{u}$ and u' ($\mathcal{V}$ reads u' in entirety).
- $\mathcal{V}$: chooses Q to be a random set of size $\ell = \Theta(\lambda)$ with $Q \subseteq [N]$. For each $j \in Q$:
  - $\mathcal{V}$ queries all m entries of the corresponding "column" of $\hat{u}$, namely $\hat{u}_{1,j}, \ldots, \hat{u}_{m,j}$.
  - $\mathcal{V}$ confirms that $\mathsf{Enc}(u')_j = \sum_{i=1}^m r_i \cdot \hat{u}_{i,j}$, halting and rejecting if not.

Evaluation phase.

- Let $q_1, q_2 \in \mathbb{F}^m$ be such that $g(r) = \langle (q_1 \otimes q_2), z\rangle$.
- The evaluation phase is identical to the testing phase, except that r is replaced with q_1 (and fresh randomness is used to choose a set Q' of columns for use in consistency checking).
- If all consistency tests pass, then $\mathcal{V}$ outputs $\langle u', q_2\rangle$ as $g(r)$.

- In settings where the evaluation phase will only be run once, the testing phase and evaluation phase can be run in parallel and the same query set Q can be used for both testing and evaluation. This saves $\approx 2\times$ in proof sizes.
- For simplicity, we describe the commitment scheme in the setting where u is indexed by $[m]^2$, i.e., u was viewed as a square matrix, this is not a requirement, and the proof size in the testing and evaluation phases can be substantially reduced by exploiting this flexibility. Specifically, if r and c denote the number of rows and columns of u, so that the number of entries in u is $c \cdot r = N$, then the proof length of the commitment scheme is roughly $2c + r \cdot \ell$ field elements where ℓ is the number of columns of the encoded matrix opened by the verifier. Here, the $2c$ term comes from the prover sending two different linear combination of the rows of u, one in the commitment phase and one in the evaluation phase, while the $r \cdot \ell$ term comes from the

verifier querying ℓ different columns of u in the testing and evaluation phases. (This optimization appeared in Ligero [4] in the context of the Reed-Solomon code.) To minimize proof length, one should set $c \approx r\ell/2$, or equivalently, one should set $r \approx \sqrt{2/\ell} \cdot \sqrt{N}$ and $c \approx \sqrt{\ell/2} \cdot \sqrt{N}$. This reduces the proof length from roughly $\ell \cdot \sqrt{N}$ if a square matrix is used, to roughly $\sqrt{2\ell} \cdot \sqrt{N}$, a savings of a factor of $\sqrt{\ell/2}$. Asymptotically, this means the proof length falls from $\Theta(\lambda\sqrt{N})$ if a square matrix is used, down to $\Theta(\sqrt{\lambda N})$, a quadratic improvement in the dependence on λ. To achieve soundness error, say, 2^{-100}, ℓ will be on the order of hundreds or thousands depending on the relative Hamming distance of the code used, and hence this optimization will lead to a reduction in proof length relative to the use of square matrices by one or more orders of magnitude.
– In settings where the commitment is trusted (e.g., applying the polynomial commitment to achieve holography), the testing phase can be omitted. An additional concrete optimization that applies when working over fields of size smaller than $\exp(\lambda)$ is in the full version of the paper [40].
– If $\mathcal{P}$ commits to the vector $\hat{u} \in (\mathbb{F}^N)^m$ with a Merkle tree, then revealing ℓ columns of $\hat{u}$ in the Testing and Evaluation phases would require providing $m \cdot \ell$ Merkle-authentication paths. Naively, this may require $\mathcal{P}$ to send up to $\Theta(m \cdot \ell \cdot \log m)$ hash values. However, by arranging the vector $\hat{u}$ in column-major order before Merkle-hashing it, the communication cost of revealing ℓ columns of $\hat{u}$ can be reduced to just the $m \cdot \ell$ requested field elements plus $O(\log m)$ hash values (a similar optimization appears in prior work [9]).

Soundness Analysis for the Testing Phase. The following claim roughly states that if $\hat{u} = (\hat{u}_1, \ldots, \hat{u}_m) \in \left(\mathbb{F}^N\right)^m$, then if even a single $\hat{u}_i$ is far from all codewords in Enc, then a random linear combination of the $\hat{u}_i$'s is also far from all codewords with high probability.

Claim. (Ames, Hazay, Ishai, and Venkitasubramaniam [4], Roth and Zémor) Let $\hat{u} = (\hat{u}_1, \ldots, \hat{u}_m) \in \left(\mathbb{F}^N\right)^m$ and for each $i \in [m]$ let c_i be the closest codeword in Enc to $\hat{u}_i$. Let E with $|E| \le (\gamma/3)N$ be a subset of the columns $j \in [N]$ of $\hat{u}$ on which there is even one row $i \in [m]$ such that $\hat{u}_{i,j} \neq c_{i,j}$. With probability at least $1 - (|E| + 1)/|\mathbb{F}| > 1 - N/|\mathbb{F}|$ over the choice of $r \in \mathbb{F}^m$, $\sum_{i=1}^m r_i \cdot \hat{u}_i$ has distance at least $|E|$ from any codeword in Enc.

Lemma 1. *If the prover passes all of the checks in the testing phase with probability at least $N/|\mathbb{F}| + (1 - \gamma/3)^\ell$, then there is a sequence of m codewords $c_1, \ldots, c_m$ in Enc such that*

$$E \coloneqq |\{j \in [N] \colon \exists i \in [m] \text{ such that } c_{i,j} \neq \hat{u}_{i,j}\}| \le (\gamma/3)N. \tag{2}$$

Proof. Let $d(b, c)$ denote the relative Hamming distance between two vectors $b, c \in \mathbb{F}^N$. Assume by way of contradiction that Eq. (2) does not hold. We explain that the prover passes the consistency tests during the testing phase with probability less than $N/|\mathbb{F}| + (1 - \gamma/3)^\ell$.

Recall that v denotes $\sum_{i=1}^{m} r_i \hat{u}_i$. By Claim 3.1, the probability over the verifier's choice of r that there exists a codeword a satisfying $d(a, v) > \gamma/3$ is less than $N/|\mathbb{F}|$. If no such a exists, then $d(\mathsf{Enc}(u'), v) \geq \gamma/3$. In this event, all of the verifier's consistency tests pass with probability at most $(1 - \gamma/3)^{\ell}$.

Completeness and Binding. Completeness holds by design. To argue binding, recall from the analysis of the testing phase that c_i denotes the codeword in Enc that is closest to row i of $\hat{u}$, and let $w := \sum_{i=1}^{m} q_{1,i} \cdot c_i$. We show that, if the prover passes the verifier's checks in the testing phase with probability more than $N/|\mathbb{F}| + (1 - \gamma/3)^{\ell}$ and passes the verifier's checks in the evaluation phase with probability more than $(1 - (2/3)\gamma)^{\ell}$, then $w = \mathsf{Enc}(u'')$.

If $w \neq \mathsf{Enc}(u'')$, then w and $\mathsf{Enc}(u'')$ are two distinct codewords in Enc and hence they can agree on at most $(1 - \gamma) \cdot N$ coordinates. Denote this agreement set by A. The verifier rejects in the evaluation phase if there is any $j \in Q'$ such that $j \notin A \cup E$, where E is as in Eq. (2). $|A \cup E| \leq |A| + |E| \leq (1 - \gamma) \cdot N + (\gamma/3)N = (1 - (2/3)\gamma)N$, and hence a randomly chosen column $j \in [N]$ is in $A \cup E$ with probability at most $1 - (2/3)\gamma$. It follows that u'' will pass the verifier's consistency checks in the evaluation phase with probability at most $(1 - (2/3)\gamma)^{\ell}$.

In summary, if the prover passes the verifier's checks in the commitment phase with probability at least

$$N/|\mathbb{F}| + (1 - \gamma/3)^{\ell}, \tag{3}$$

then, in the following sense, the prover is *bound* to the polynomial g^* whose coefficients in the Lagrange basis are given by $c_{1,1}, \ldots, c_{m,m}$, where $c_i \in \mathbb{F}^N$ denotes the closest codeword to row i of the vector $\hat{u}$ sent in the commitment phase: on evaluation query r, the verifier either outputs $g^*(r)$, or else rejects in the evaluation phase with probability at least

$$1 - (1 - (2/3)\gamma)^{\ell}. \tag{4}$$

The polynomial commitment scheme provides standard extractability properties. We show this by giving two different extractors.

Extractability via Efficient Decoding. The first is a simple straight-line extractor that is efficient if the error-correcting code Enc has a polynomial-time decoding procedure that can correct up to a $\gamma/4$ fraction of errors. This is because with the IOP-to-succinct-argument transformation of [15,47,55,70], it is known that, given a prover $\mathcal{P}$ that convinces the argument-system verifier to accept with non-negligible probability, there is an efficient straight-line extractor capable of outputting IOP proof string π that "opens" the Merkle commitment sent by the argument system prover in the commitment phase. Moreover, there is an IOP prover strategy $\mathcal{P}'$ for the testing and evaluation phases by which $\mathcal{P}'$ can convince the IOP verifier in those phases to accept with non-negligible probability when the first IOP message is π ($\mathcal{P}'$ merely simulates $\mathcal{P}$ in those phases).

Our analysis of the testing phase of the polynomial commitment scheme (Lemma 1) then guarantees that each row of the extracted string π has relative Hamming distance at most $\gamma/3$ from some codeword. Hence, row-by-row decoding provides the coefficients of the multilinear polynomial that the prover is bound to. If the decoding procedure runs in polynomial time, the extractor is efficient.

Extractability Without Decoding. If the error-correcting code does not support efficient decoding, then even though one can efficiently extract the IOP proof string π underlying the Merkle-commitment sent in the commitment phase of the commitment scheme, one can not necessarily decode (each row of) the string to efficiently extract from π the polynomial that the commiter is bound to.

Instead, the extractor can proceed as follows. We assume throughout the below that Expressions (3) and (4) are negligible (say, exponentially small in the security parameter λ), which holds so long as $|\mathbb{F}| \geq \exp(\lambda)$ and the number of column openings is $\ell = \Theta(\lambda)$.

The testing phase of the commitment scheme can be viewed as a 3-move public-coin argument in which the verifier moves first. First, the verifier sends a challenge vector $r \in \mathbb{F}^m$. Second, the prover responds with a vector claimed to equal $\sum_{i=1}^{m} r_i u_i$. Third, the verifier chooses a set Q of random columns to use in the consistency test, and performs the consistency test by querying the committed proof string π at all entries of the columns in Q.

Given any efficient prover strategy that passes the verifier's checks in the testing phase with non-negligible probability, we show in the following lemma that there is a polynomial-time extraction procedure capable of outputting m linearly independent challenge vectors $r_1, \ldots, r_m \in \mathbb{F}^m$ from the testing phase of the protocol, and m response vectors $u'_1, \ldots, u'_m \in \mathbb{F}^m$ of the prover, each of which pass the verifier's consistency checks in the testing phase with non-negligible probability.

Lemma 2. *Suppose there is a deterministic prover strategy $\mathcal{P}$ that, following the commitment phase of the polynomial commitment scheme, passes the verifier's checks in the testing phase of the polynomial commitment scheme with probability ϵ. Then there is a randomized extraction procedure $\mathcal{E}$ that runs in expected time poly$(m, \lambda, 1/\epsilon)$ and such that the following holds. Given the ability to repeatedly rewind $\mathcal{P}$ to the start of the testing phase, with probability at least $1 - 2^{-\Omega(\lambda)}$, $\mathcal{E}$ outputs m linearly independent challenge vectors $r_1, \ldots, r_m \in \mathbb{F}^m$ from the testing phase, and m corresponding response vectors $u'_1, \ldots, u'_m \in \mathbb{F}^m$ of the prover, each of which pass the verifier's checks in the testing phase with probability at least ϵ.*

Before proving Lemma 2, we explain how to extract the desired polynomial given the extracted challenge vectors $r_1, \ldots, r_m \in \mathbb{F}^m$ and m response vectors $u'_1, \ldots, u'_m \in \mathbb{F}^m$. Observe that the testing phase and the evaluation phase of the polynomial commitment scheme are identical up to how the challenge vector is selected. In addition, for each challenge r_i the prover's response u'_i passes the verifier's consistency checks with non-negligible probability. Hence, the binding analysis for the commitment scheme implies that $u'_1, \ldots, u'_m$ are all consistent with the evaluations of a fixed multilinear polynomial g^*, i.e., for $i = 1, \ldots, m$, $u'_i = r_i^T \cdot C$ where C is the coefficient matrix of g^* in the Lagrange basis. Since the r_i vectors are linearly independent, these m linear equations uniquely specify C, and in fact C can be found in polynomial time using Gaussian elimination.

Proof. We begin the proof by assuming that the extractor knows ϵ. We later explain how to remove this assumption. We refer to the extraction procedure that depends on ϵ as the "base extraction procedure".

If $\epsilon < 1/\sqrt{|\mathbb{F}|}$, then the base extractor can simply abort, by assumption that the field size is at least $\exp(\lambda)$. Below, we assume that $\epsilon > 1/\sqrt{|\mathbb{F}|}$.

Fix the extracted proof string $\pi = (\pi_1, \ldots, \pi_m) \in (\mathbb{F}^N)^m$ that "opens" the Merkle-commitment sent by the committer during the commitment phase.

Observe that for any verifier challenge $r' \in \mathbb{F}^m$ and prover response u', one can efficiently compute the probability (over the random choice of column set Q) that u' will pass the verifier's consistency checks. Specifically, if η is the number of columns i such that $\left(\sum_{j=1}^m r'_j \pi_j\right)_i = u'_i$, and ℓ is the number of columns selected by the verifier, then this probability is $(\eta/N)^\ell$ (here, for simplicity let us assume columns are selected with replacement, but an exact expression can also be given when columns are selected without replacement).

Let T denote the set of all challenges r such that $\mathcal{P}$'s response u to r passes the consistency checks with probability at least $\epsilon/2$. By averaging, since $\mathcal{P}$ passes all checks in the testing phase with probability at least ϵ, $|T| \geq (\epsilon/2) \cdot |\mathbb{F}|^m$. The extractor's goal is to efficiently identify a subset $S = \{r_1, \ldots, r_m\}$ of T that spans $\mathbb{F}^m$.

The extractor $\mathcal{E}$ works by repeatedly picking challenge vectors r uniformly at random from $\mathbb{F}^m$, and running $\mathcal{P}$ on challenge r to get a response u; this enables $\mathcal{E}$ to determine whether $r \in T$, and if so, $\mathcal{E}$ adds r to S. The extractor tries $\ell = 18(m + \lambda)/\epsilon$ vectors r, aborting if it fails to identify at least m vectors in T. We claim that the probability the extractor fails to identify m vectors to add to S is at most $1 - 2^{-(m+\lambda)}$. To see this, model each choice of challenge vector r as a Poisson trial with success probability at least $\epsilon/2$. Let μ be the expected number of successes after ℓ Poisson trials each with success probability at least $\epsilon/2$. Then μ is at least $\ell \cdot \epsilon/2 \geq 9(m+\lambda)$. Let $\delta = 1/2$. Since $m \leq \mu/9 \leq (1-\delta) \cdot \mu$, standard Chernoff bounds (e.g., [56, Theorem 4.5]) upper bound the probability that the number of successes is less than m by $e^{-\mu\delta^2/2} \leq e^{-9(m+\lambda)/8} \leq 2^{-(m+\lambda)}$.

We now argue that with probability at least $1 - (m-1) \cdot (2/\epsilon) \cdot |\mathbb{F}|^{-1}$, the first m vectors that $\mathcal{E}$ adds to S are linearly independent (in the event that this is not the case, the extractor aborts). Denote these m vectors by $r_1, \ldots, r_m \in \mathbb{F}^m$. Observe that each vector r_i is a random element of T. We now explain that for each $i = 2, \ldots, m$, the probability that $r_i \in \operatorname{span}(r_1, \ldots, r_{i-1})$ is at most $(m-1) \cdot (2/\epsilon) \cdot |\mathbb{F}|^{-1}$. To see this, observe that since the dimension of $\operatorname{span}(r_1, \ldots, r_{i-1})$ is at most $i-1$, the span contains at most $|\mathbb{F}|^{i-1}$ vectors. Since r_i is a uniform random vector from T and $|T| \geq (\epsilon/2) \cdot \mathbb{F}^m$, the probability that $r_i \in \operatorname{span}(r_1, \ldots, r_{i-1})$ is at most $(2/\epsilon) \cdot |\mathbb{F}|^{i-1}/|\mathbb{F}|^m \leq (2/\epsilon) \cdot |\mathbb{F}|^{-1}$. The claim then follows by a union bound over all $m-1$ vectors $r_2, \ldots, r_m$. That is, the probability that $r_1, \ldots, r_m$ are not linearly independent is at most $(m-1) \cdot (2/\epsilon) \cdot |\mathbb{F}|^{-1}$.

Since $\epsilon > 1/\sqrt{|\mathbb{F}|}$, we conclude that the extractor aborts with probability at most $2^{-(m+\lambda)} + (m-1)/\sqrt{|\mathbb{F}|}$. This is a negligible function, by the assumption that $|\mathbb{F}|$ is at least $\exp(\lambda)$.

The above base extraction procedure depends on ϵ, because the extractor tries out $\ell = 18(m+\lambda)/\epsilon$ vectors r (aborting if it fails to identify m vectors in T within that many tries). The following modification eliminates this dependence. Iteratively run the base extraction procedure with ϵ set to the geometrically decreasing sequence of values $\epsilon' = 2^{-1}, 2^{-2}, \ldots, 2^{-\lambda/8}$ halting when the extraction procedure succeeds, and aborting if the extractor reaches $\epsilon' < 2^{-\lambda/8}$ without a witness being identified. The above analysis guarantees that when the extraction procedure is run with ϵ' less than or equal to ϵ, it successfully outputs a witness with probability at least $1 - 2^{-(m+\lambda)} - (m-1)/\sqrt{|\mathbb{F}|} \geq 1 - 2^{-\lambda/3}$. The expected runtime of this modified extraction procedure is at most $36(m+\lambda)/\epsilon + 2^{-\lambda/3} \cdot 18(m+\lambda) \cdot 2^{\lambda/8} \leq \text{poly}(m, \lambda, 1/\epsilon)$. $\square$

The extraction procedure given in Lemma 2 succeeds with overwhelming probability and runs in expected time $\text{poly}(m, \lambda, 1/\epsilon)$ given access to a prover that produces an accepting proof with probability at least ϵ. However, if ϵ is not inverse-polynomial in m and λ, this expected runtime is not polynomial in m and λ. The definition of knowledge soundness (Definition 2) and extractable polynomial commitments (Definition 3) requires that the extractor run in expected polynomial time *regardless of* ϵ, and that whenever the prover succeeds in outputting a convincing proof π, the extractor outputs a witness with all but negligible probability. The lemma below achieves this.

Lemma 3. *There is a randomized extraction procedure $\mathcal{E}$ that runs in expected time poly(m, λ) and such that the following holds. $\mathcal{E}$ first runs $\mathcal{P}$ once during the testing phase of the above polynomial commitment scheme. If $\mathcal{P}$ fails to pass the verifier's checks on the first run, the extractor aborts. Otherwise, with probability at least $1-2^{-\Omega(\lambda)}$, $\mathcal{E}$ outputs m linearly independent challenge vectors $r_1, \ldots, r_m \in \mathbb{F}^m$ from the testing phase, and m corresponding response vectors $u'_1, \ldots, u'_m \in \mathbb{F}^m$ of the prover, each of which pass the verifier's checks in the testing phase with probability at least ϵ.*

Proof. We follow the presentation of Hazay and Lindell [43, Theorem 6.5.6] of an extraction strategy originally due to Goldreich [38].

As described in the statement of the lemma, $\mathcal{E}$ first runs $\mathcal{P}$ once during the testing phase, and if $\mathcal{P}$ does not pass the verifier's checks, then $\mathcal{E}$ aborts. If $\mathcal{P}$ does pass the verifier's checks in the testing phase, then $\mathcal{E}$ proceeds to estimate the value ϵ (i.e., the probability that $\mathcal{P}$ indeed passes the verifier's checks in the testing phase). It does this by rewinding $\mathcal{P}$ to the start of the testing phase until $12 \cdot (m+\lambda)$ successful verifications occur. If T runs of $\mathcal{P}$ are required before $12 \cdot (m+\lambda)$ successful verifications occur, then the extractor uses $\epsilon' = 12 \cdot (m+\lambda)/T$ as an estimate of ϵ. The extractor then runs the base extraction procedure from the proof of Lemma 2 with ϵ set to $\epsilon'/2$. Throughout its entire execution, the extractor also keeps a counter of how many times it has run the prover through the testing phase of the polynomial commitment scheme, and if this number ever exceeds 2^m, it aborts. In this event, we say that the extractor has "timed out".

Let us analyze the success probability of the extractor under the assumption that $\epsilon > 2^{-m/2}$ (if $\epsilon \le 2^{-m/2}$, then ϵ is negligible, and hence it is acceptable for the extractor to succeed with probability 0). [43, Proof of Theorem 6.5.6] shows that with probability at least $1 - 2^{-m+\lambda}$, ϵ' is between $2\epsilon/3$ and 2ϵ. We call this the "good event". Since $\epsilon > 2^{-m/2}$, if the good event occurs, the extractor runs in fewer than 2^m steps and hence does not time out. And the proof of Lemma 2 shows that, conditioned on the good event occurring, the extractor succeeds with probability at least $1 - 2^{-(m+\lambda)} - (m-1)/\sqrt{|\mathbb{F}|}$. Hence, by a union bound, the extractor succeeds with probability at least $1-2^{-(m+\lambda)}-(m-1)/\sqrt{|\mathbb{F}|}-2^{-m+\lambda} \ge 1 - 2^{-\Omega(\lambda)}$.

We now explain that the above extractor runs in expected polynomial time, regardless of the value of ϵ. Let us first consider the case that $\epsilon \le 2^{-m}$. The probability that the prover passes the verifier's checks in the first prover execution is ϵ, and the extractor never runs for more than 2^m steps. So in this case the expected runtime of the extractor is at most $\epsilon \cdot 2^m \le 1$, plus the time required to run the verification procedure of the testing phase, which is polynomial in m and λ.

Now consider the case that $\epsilon > 2^{-m}$. The first run of the prover passes the verifier's tests with probability ϵ. If this does not happen, the extractor aborts. Otherwise, the extractor proceeds. If the extractor proceeds and the good event occurs, the extractor runs in time at most $O((m+\lambda)/\epsilon)$. The good event fails to occur with probability only at most $2^{-(m+\lambda)}$, and in this case the extractor still does not run for more than 2^m steps.

Hence, the expected runtime of the extractor is at most

$$O(\epsilon \cdot (m+\lambda)/\epsilon + 2^m \cdot 2^{-(m+\lambda)}) = O(m+\lambda).$$

□

The runtime of our knowledge extractor that does not perform decoding may imply reduced concrete security, but the effect is small. Compared to the extractor that uses efficient decoding, the rewinding extractor requires m "successful" executions of the prover rather than just one, where $m = \sqrt{n}$, for n equal to the degree of the committed polynomial. So, roughly speaking, the runtime of the rewinding extractor is worse by a factor of m, plus an additive term that accounts for the cost of Gaussian elimination.

In the context of Brakedown (our SNARK for R1CS that utilizes this polynomial commitment scheme, see Sect. 5), n is roughly equal to the number of R1CS constraints. As an example, when Brakedown is used to prove a statement about a cryptographic primitive, e.g., knowledge of pre-image of a hash function that is implemented in (say) $1000 \cdot \lambda$ R1CS constraints, then a factor-of-m increase in extractor time corresponds to a loss of roughly $(1/2) \cdot \log(1000\lambda)$ bits of security, which in practice is less than 10 bits.

4 Fast Linear Codes with Linear-Time Encoding

This section describes our construction of practical linear codes with linear-time encoding that we use in Brakedown's implementation of the polynomial commitment scheme from Sect. 3.1. We begin with a sketch of our encoding procedure and of the analysis of its minimum distance.

Overview of Encoding. In this overview we restrict our attention to a construction of a code with distance $\delta = 1/20$ and rate $\rho = 3/5$. The encoding procedure is recursive. For a message $x \in \mathbb{F}^n$ of length n, the codeword consists of three parts $\mathsf{Enc}(x) = (x, z, v)$. The first is the "systematic part" that just copies the message x of length n. The other parts (z, v) are obtained via the following three-step process. First, multiply x by a random sparse $n \times n/5$ matrix to "compress" x to a vector y of length $n/5$. Then obtain z of length $n/3$ by recursively encoding y, and finally obtain v of length $n/3$ by multiplying z by a random sparse $n/3 \times n/3$ matrix B.

Overview of Distance Analysis. The distance analysis proceeds in three cases. Since the code is linear, we merely need to show that the encoding of any non-zero message x has Hamming weight at least $\delta n/\rho = n/12$. We sketch the analysis for a fixed x, but the formal analysis in our full version of the paper [40] holds with overwhelming probability for all x simultaneously.

- If the Hamming weight of x is $> n/12$, then the systematic part x of $\mathsf{Enc}(x)$ already ensures that $\mathsf{Enc}(x)$ has a sufficiently large Hamming weight.
- Otherwise, we show that with overwhelming probability over the random choice of A, y will be non-zero. This, in turn, ensures by induction that $z = \mathsf{Enc}(y)$ has "reasonably large" Hamming weight, at least $n/60$. If the Hamming weight of z is in fact larger than $n/12$ then we are done because z is part of $\mathsf{Enc}(x)$.
- Otherwise, the Hamming weight of z is between $n/60$ and $n/12$. In this case, we show that, with overwhelming probability, B "mixes" the non-zero coordinates of z, and results in a $v = zB$ of Hamming weight at least $n/12$, completing the analysis.

A full version of our paper [40] provides details of our linear-time codes.

5 Linear-Time SNARKs for R1CS

The following theorem captures our main result.

Theorem 2. *Assuming that $|\mathbb{F}| = 2^{\Theta(\lambda)}$ there exists a preprocessing SNARK for $\mathcal{R}_{R1CS}$ in the random oracle model, with the following efficiency characteristics, where M denotes the dimensions of the R1CS matrices, N denotes the number of non-zero entries, and a fixed positive integer t:*

- *the preprocessing cost to the verifier is $O(N)$ $\mathbb{F}$-ops;*
- *the running time of the prover is $O(N)$ $\mathbb{F}$-ops;*

– *the running time of the verifier is* $O_\lambda(N^{1/t})$ $\mathbb{F}$*-ops; and*
– *the proof size is* $O_\lambda(N^{1/t})$*.*

A proof of Theorem 2 is in the full version of the paper [40]. In a nutshell, we obtain a SNARK with the claimed performance profile as follows. We first make explicit Spartan's polynomial IOP for R1CS. We then combine our polynomial commitment schemes with Spartan's polynomial IOP to obtain a succinct argument per prior compilers [28]. Finally, to achieve linear-time computation commitments (also called holography) we invoke prior techniques [63] to transform our polynomial commitment scheme from Theorem 1 into one that handles so-called *sparse* multilinear polynomials. This step is necessary because the polynomials used in Spartan's IOP to "capture" the R1CS matrices are sparse.

5.1 A Self-contained Description of Brakedown and Shockwave

For an R1CS instance, $\mathbb{X} = (\mathbb{F}, A, B, C, M, N, \mathsf{io})$ and a purported witness W, let $Z = (W, 1, \mathsf{io})$. For ease of notation, we assume that the vector W and the vector $(1, \mathsf{io})$ have the same length. We interpret the matrices A, B, C as functions mapping domain $\{0,1\}^{\log M} \times \{0,1\}^{\log M}$ to $\mathbb{F}$ in the natural way. That is, an input in $\{0,1\}^{\log M} \times \{0,1\}^{\log M}$ is interpreted as the binary representation of an index $(i, j) \in [M] \times [M]$, where $[M] := \{1, \ldots, M\}$ and the function outputs the (i, j)'th entry of the matrix. Similarly we interpret Z and $(1, \mathsf{io})$ as functions with the following respective signatures in the same manner: $\{0,1\}^s \to \mathbb{F}$ and $\{0,1\}^{s-1} \to \mathbb{F}$.

Note that the *multilinear extension (MLE) polynomial* $\widetilde{Z}$ of Z satisfies

$$\widetilde{Z}(X_1, \ldots, X_{\log M}) = (1 - X_1) \cdot \widetilde{W}(X_2, \ldots, X_{\log M}) + X_1 \cdot \widetilde{(1, \mathsf{io})}(X_2, \ldots, X_{\log M}). \tag{5}$$

Here, the MLE refers to the unique multilinear polynomial $\widetilde{Z}$ satisfying $\widetilde{Z}(x_1, \ldots, x_{\log M}) = Z(x_1, \ldots, x_{\log M})$ for all $(x_1, \ldots x_{\log M}) \in \{0,1\}^{\log M}$. Indeed, the RHS of Eq. (5) is a multilinear polynomial, and it is easily checked that $\widetilde{Z}(x_1, \ldots, x_{\log M}) = Z(x_1, \ldots, x_{\log M})$ for all $x_1, \ldots, x_{\log M}$ (since the first half of the evaluations of Z are given by W and the second half are given by the vector $(1, \mathsf{io})$).

From [63, Theorem 4.1], checking if $(\mathbb{X}, W) \in \mathcal{R}_{\text{R1CS}}$ is equivalent, except for a soundness error of $\log M/|\mathbb{F}|$ over the choice of $\tau \in \mathbb{F}^s$, to checking if the following identity holds:

$$0 \overset{?}{=} \sum_{x \in \{0,1\}^s} \widetilde{eq}(\tau, x) \cdot \left[\left(\sum_{y \in \{0,1\}^s} \widetilde{A}(x,y) \cdot \widetilde{Z}(y) \right) \cdot \left(\sum_{y \in \{0,1\}^s} \widetilde{B}(x,y) \cdot \widetilde{Z}(y) \right) - \sum_{y \in \{0,1\}^s} \widetilde{C}(x,y) \cdot \widetilde{Z}(y) \right] \tag{6}$$

where $\widetilde{eq}$ is the MLE of $eq : \{0,1\}^s \times \{0,1\}^s \to \mathbb{F}$:

$$eq(x,e) = \begin{cases} 1 & \text{if } x = e \\ 0 & \text{otherwise.} \end{cases}$$

That is, if $(\mathbb{X}, W) \in \mathcal{R}_{\text{R1CS}}$, then Eq. (6) holds with probability 1 over the choice of τ, and if $(\mathbb{X}, W) \notin \mathcal{R}_{\text{R1CS}}$, then Eq. (6) holds with probability at most $O(\log M/|\mathbb{F}|)$ over the random choice of τ.

Consider computing the right hand side of Eq. (6) by applying the well-known *sum-check protocol* [54] to the polynomial

$$g(x) := \widetilde{eq}(\tau, x) \cdot \left[\left(\sum_{y \in \{0,1\}^s} \widetilde{A}(x,y) \cdot \widetilde{Z}(y) \right) \cdot \left(\sum_{y \in \{0,1\}^s} \widetilde{B}(x,y) \cdot \widetilde{Z}(y) \right) - \sum_{y \in \{0,1\}^s} \widetilde{C}(x,y) \cdot \widetilde{Z}(y) \right]$$

From the verifier's perspective, this reduces the task of computing the right hand side of Eq. (6) to the task of evaluating g at a random input $r_x \in \mathbb{F}^s$. Note that the verifier can evaluate $\widetilde{eq}(\tau, r_x)$ unassisted in $O(\log M)$ field operations, as it is easily checked that $\widetilde{eq}(\tau, r_x) = \prod_{i=1}^{s} (\tau_i r_{x,i} + (1-\tau_i)(1-r_{x,i}))$. With $\widetilde{eq}(\tau, r_x)$ in hand, $g(r_x)$ can be computed in $O(1)$ time given the three quantities $\sum_{y \in \{0,1\}^s} \widetilde{A}(r_x, y) \cdot \widetilde{Z}(y)$, $\sum_{y \in \{0,1\}^s} \widetilde{B}(r_x, y) \cdot \widetilde{Z}(y)$, and $\sum_{y \in \{0,1\}^s} \widetilde{C}(r_x, y) \cdot \widetilde{Z}(y)$.

These three quantities can be computed by applying the sum-check protocol three more times in parallel, once to each of the following three polynomials (using the same random vector of field elements, $r_y \in \mathbb{F}^s$, in each of the three invocations): $\widetilde{A}(r_x, y) \cdot \widetilde{Z}(y)$, $\widetilde{B}(r_x, y) \cdot \widetilde{Z}(y)$, $\widetilde{C}(r_x, y) \cdot \widetilde{Z}(y)$.

To perform the verifier's final check in each of these three invocations of the sum-check protocol, it suffices for the verifier to evaluate each of the above 3 polynomials at the random vector r_y, which means it suffices for the verifier to evaluate $\widetilde{A}(r_x, r_y)$, $\widetilde{B}(r_x, r_y)$, $\widetilde{C}(r_x, r_y)$, and $\widetilde{Z}(r_y)$. We present the protocol in this section assuming that the verifier can evaluate $\widetilde{A}$, $\widetilde{B}$, and $\widetilde{C}$ at the point (r_x, r_y) in time $O(\sqrt{N})$—if this is not the case, then as discussed in Sect. 5.1 and detailed in the full version of the paper, these polynomials can each be committed in pre-processing and the necessary evaluations revealed by the prover, thereby achieving holography. $\widetilde{Z}(r_y)$ can be obtained from one query to $\widetilde{W}$ and one query to $\widetilde{(1, \mathsf{io})}$ via Eq. (5).

In summary, we have the following succinct interactive argument. It is public coin, and can be rendered non-interactive via the Fiat-Shamir transformation.

1. $\mathcal{P} \rightarrow \mathcal{V}$: a commitment to the $(\log M - 1)$-variate multilinear polynomial $\widetilde{W}$ using the polynomial commitment scheme of §3.1.
2. $\mathcal{V} \rightarrow \mathcal{P}$: $\tau \in_R \mathbb{F}^s$
3. $\mathcal{V} \leftrightarrow \mathcal{P}$: run the sum-check reduction to reduce the check in Equation (6) to checking if the following hold, where r_x, r_y are vectors in $\mathbb{F}^s$ chosen at random by the verifier over the course of the sum-check protocol:
   - $\widetilde{A}(r_x, r_y) \stackrel{?}{=} v_A$, $\widetilde{B}(r_x, r_y) \stackrel{?}{=} v_B$, and $\widetilde{C}(r_x, r_y) \stackrel{?}{=} v_C$; and
   - $\widetilde{Z}(r_y) \stackrel{?}{=} v_Z$.
4. $\mathcal{V}$:
   - check if $\widetilde{A}(r_x, r_y) \stackrel{?}{=} v_A$, $\widetilde{B}(r_x, r_y) \stackrel{?}{=} v_B$, and $\widetilde{C}(r_x, r_y) \stackrel{?}{=} v_C$, (recall that we have assumed that each of $\widetilde{A}, \widetilde{B}, \widetilde{C}$ can be evaluated in $O(\sqrt{N})$ time).
   - check if $\widetilde{Z}(r_y) \stackrel{?}{=} v_Z$ by checking if: $v_Z = (1 - r_y[1]) \cdot v_W + r_y[1] \cdot \widetilde{(io, 1)}(r_y[2..])$, where $r_y[2..]$ refers to a slice of r_y without the first element of r_y (Eq. (5)), and $v_W \leftarrow \widetilde{W}(r_y[2..])$ is obtained via the evaluation procedure of the polynomial commitment scheme (§3.1).

By composing the SNARK of Theorem 2 with known zkSNARKs [41,63,65], we obtain zkSNARKs with shorter proofs (the full version of this paper [40] provides detailed cost profiles of the resulting zkSNARKs). Specifically, the prover in the composed SNARKs proves that it knows a proof π that would convince the SNARK verifier in Theorem 2 to accept. Perfect zero-knowledge of the resulting composed SNARK is immediate from the zero-knowledge property of the SNARKs from these prior works [41,63,65]. Perfect completeness follows from the perfect completeness properties of these prior works and of Theorem 2. Knowledge soundness follows from a standard argument [19,70]: one composes the knowledge extractors of the two constituent SNARKs to get a knowledge extractor for the composed SNARK.

6 Implementation and Evaluation

We evaluate the performance of two polynomial commitment schemes and two SNARKs based on these schemes. Specifically, we evaluate two instantiations of the polynomial commitment scheme of Sect. 3.1: Ligero-PC, which uses the Reed-Solomon code (this scheme is implicit in Ligero [4]), and Brakedown-PC, which uses the new linear-time error-correcting code described in Sect. 4.

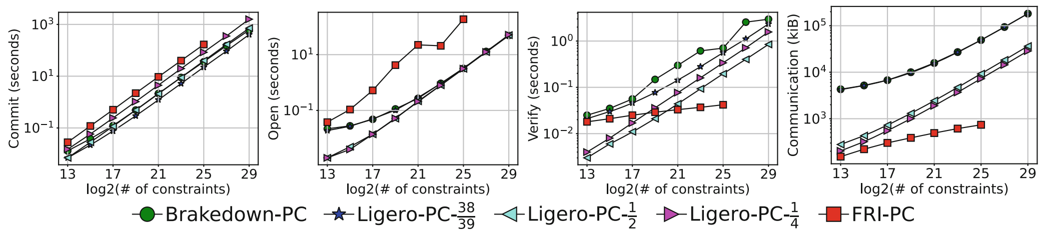

Fig. 1. Microbenchmark results (Sect. 6.1); lower is better. Brakedown-PC uses the parameters in the third line of [40, Figure 7]; Ligero-PC-$^{38}/_{39}$, Ligero-PC-$^{1}/_{2}$, Ligero-PC-$^{1}/_{4}$, and FRI-PC are instantiated with Reed-Solomon rates of $^{38}/_{39}$, $^{1}/_{2}$, $^{1}/_{4}$, and $^{1}/_{4}$, respectively. FRI-PC results are incomplete as the prover ran out of memory for larger instances.

Implementation. We implement Ligero-PC and Brakedown-PC in ≈3500 lines of Rust. This includes an implementation of the polynomial commitment of Sect. 3.1 that is generic over fields, error-correcting codes, and hash functions; implementations of the Reed-Solomon code and our new linear-time code; and a fast parallelized FFT. We also integrate our implementation with Spartan [3], yielding a SNARK library; this took less than 100 lines of glue Rust code.

All reported measurements of our implementation use the BLAKE3 hash function [57]. Because Ligero-PC needs to perform FFTs, our measurements use fields of characteristic p such that $p - 1$ is divisible by 2^{40}, which ensures that reasonably large FFTs are possible in the field; we choose p at random.

6.1 Evaluation of Polynomial Commitment Schemes

This section evaluates the concrete costs of our polynomial commitment schemes and a prior baseline, for univariate polynomials, over the standard monomial basis, of degree 2^{13} to 2^{29} (for our schemes, the costs for such univariate polynomials is identical to the costs for multilinear polynomials having 13 to 29 variables).

Baseline. The FRI protocol [8] underlies all prior IOP-based polynomial commitment schemes and all built post-quantum schemes. (Lattice-based Bulletproofs [24] could also be used to construct post-quantum schemes, but to our knowledge these have not been implemented.) Like Ligero-PC, FRI requires an FFT-friendly field for efficiency. Vlasov and Panarin [71] use FRI to construct a univariate polynomial commitment, which we call FRI-PC; Virgo [78] effectively extends FRI-PC to multilinear polynomials by invoking an interactive proof, increasing costs.[3] We evaluate the C++ FRI implementation in libiop [52]. Following the best known soundness analysis [10,71], we set the Reed-Solomon rate to $^{1}/_{4}$ and number of queries to 189.

[3] RedShift and ethStark [1,46] use FRI to construct a related primitive called a *list polynomial commitment* with a relaxed notion of soundness, and smaller opening proofs (by up to ≈30% using Reed-Solomon rate $^{1}/_{4}$ and existing analyses). That primitive, however, is not a drop-in replacement for polynomial commitments, so we restrict our focus to the latter.

Parameters of the Error-Correcting Codes. We instantiate Brakedown-PC with the parameters given on the third line of [40, Figure 7]. For Ligero-PC, the rate of the Reed-Solomon code trades between proving and verification costs: roughly, a lower rate gives a slower prover but smaller proofs and faster verification. To explore this tradeoff, we test Ligero-PC with three different code rates: $^{38}/_{39}$ gives proof sizes roughly matching Brakedown-PC, $^1/_2$ gives smaller proofs than Brakedown-PC at roughly comparable prover cost, and $^1/_4$ gives even smaller proofs at the cost of greater prover computation.[4]

Other Parameters. We evaluate all schemes over 255-bit prime fields. We set parameters of the commitment schemes to obtain 128 bits of security (the exception is that the randomized code generation procedure in Brakedown-PC is designed to have at most a 2^{-100} probability of failing to satisfy the requisite distance properties according to the analysis of Sect. 4; this is acceptable because code generation is done in public). To achieve this, we set the number of columns opened in Brakedown-PC to 6593, in Ligero-PC-$^{38}/_{39}$ to 7054, in Ligero-PC-$^1/_2$ to 309, and in Ligero-PC-$^1/_4$ to 189.

Setup and Method. Our testbed for this section is an Azure Standard F64s_v2 virtual machine (64 Intel Xeon Platinum 8272CL vCPUs, 128 GiB memory) with Ubuntu 18.04. We measure single-threaded speed for committing, opening, and verifying; and we report communication cost. For each experiment, we run the operation 10 times and report the average; in all cases, variation is negligible.

Results. Figure 1 reports the results. The FRI-PC prover ran out of memory for polynomials of degree greater than 2^{25}.

For Brakedown-PC and Ligero-PC, the dominant cost for the prover is committing to the polynomial. For large enough polynomials, Brakedown-PC's commitment computation is as fast or faster than Ligero-PC-$^1/_2$'s, and roughly 2–3× faster than Ligero-PC-$^1/_4$'s. Computing commitments in Ligero-PC-$^{38}/_{39}$ is faster than in Brakedown-PC, but Ligero-PC-$^{38}/_{39}$ does not support multilinear polynomials (see Footnote 4).

For FRI-PC, committing and opening have similar (high) costs: on the sizes where the FRI prover succeeded ($\leq 2^{25}$), committing and opening with FRI-PC is ≈2–9× slower than Brakedown-PC and ≈3–7× slower than the slowest Ligero-PC; for Brakedown-PC, the gap widens with increasing size.

Ligero-PC-$^1/_2$ and Ligero-PC-$^1/_4$ have lower verification cost than Ligero-PC-$^{38}/_{39}$ and Brakedown-PC, though this advantage shrinks as instances grow. FRI-PC's verification cost is up to 17× lower than the other schemes.

[4] Rate parameter $^{38}/_{39}$ *cannot* be used in the Ligero-PC scheme for multilinear polynomials (as required by Shockwave; Sect. 6.2). This is because Ligero-PC's FFT uses power-of-two–length codewords, and multilinear polynomial evaluation can only be decomposed into tensor products with power-of-two-sized tensors (see Sect. 3.1). Since ρ is the ratio of one tensor's size to codeword length, ρ^{-1} must be a power of two. As a result, $\rho = {}^1/_2$ is the highest rate Ligero-PC supports for multilinear polynomials.

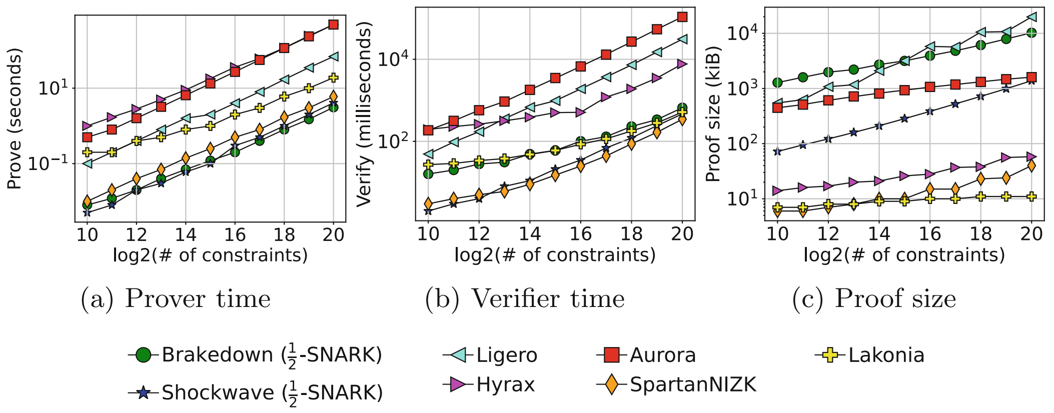

Fig. 2. $\frac{1}{2}$-SNARK results (Sect. 6.2); lower is better. Brakedown, Shockwave, Ligero, and Aurora are plausibly post-quantum secure.

Brakedown-PC and Ligero-PC-38/39 have nearly the same communication cost, by design. Ligero-PC-1/2 has ≈5–15× less communication than Brakedown-PC, and Ligero-PC-1/4 has ≈6–21× less than Brakedown-PC; like verification time, their proof size advantage shrinks as instance size grows. FRI-PC's communication is significantly lower: ≈22–66× less than Brakedown-PC or Ligero-PC-38/39, ≈2–13× less than Ligero-PC-1/2, and ≈1.3–10× less than Ligero-PC-1/4.

In sum, Brakedown-PC has a concretely and asymptotically fast prover (especially for multilinear polynomials) but gives large proofs and slower verification than the other schemes. Brakedown-PC's proofs are larger because higher ρ implies more column openings; for large polynomials, the overhead is $\approx \sqrt{\ell'}$, where ℓ' is the ratio of the number of column openings. For these instance sizes, Ligero-PC's prover is competitive with Brakedown-PC's and its proof size and verification cost are lower. FRI-PC gives much smaller proofs and lower verification cost but has a much slower prover. Of course, neither Ligero-PC nor FRI-PC is field-agnostic; Brakedown-PC is.

In our full paper [40] we report on Brakedown-PC and Ligero-PC experiments with 64 threads. For large polynomials, proving times improve by ≈16–34×.

6.2 Evaluation of Brakedown and Shockwave SNARKs

Metrics, Method, and Baselines. As is standard in the SNARKs literature, our metrics are: (1) the prover time to produce a proof; (2) the verifier time to verify a proof; (3) proof sizes; and (4) the verifier's preprocessing costs. As baselines, we consider two types of SNARKs: (1) schemes that achieve verification costs sub-linear in the size of the statement (which implies sub-linear proof sizes); and (2) schemes that only achieve proof sizes that are sub-linear in the size of the statement. We refer to the latter type of schemes as $\frac{1}{2}$-*SNARKs*. Additionally, we focus on schemes that do not require a trusted setup (we refer the reader to Spartan [63] for a comparison between our baselines with state-of-the-art SNARKs with trusted setup).

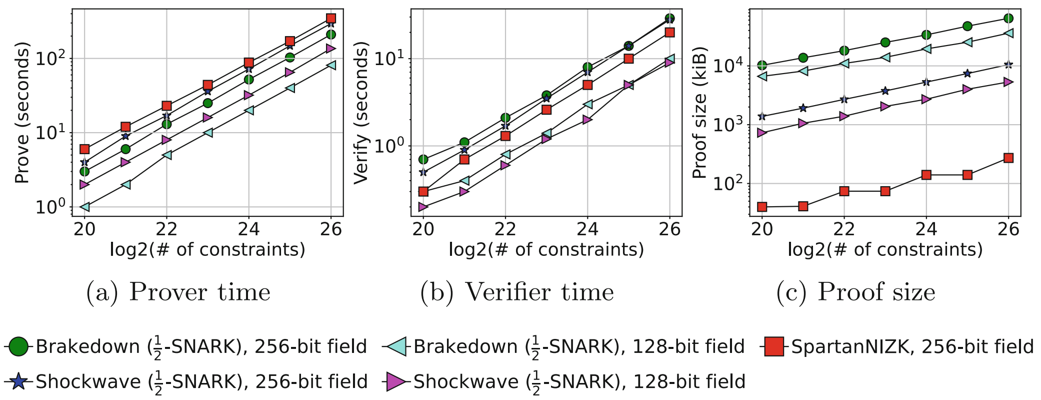

Fig. 3. $\frac{1}{2}$-SNARK results for varying field sizes (Sect. 6.2); lower is better. Brakedown and Shockwave are plausibly post-quantum secure.

The reader may wonder why we include results for our $\frac{1}{2}$-SNARK given that our implementation is not yet zero-knowledge; after all, the verifier runtime in any non-zero-knowledge $\frac{1}{2}$-SNARK is commensurate with that of the trivial proof system in which the prover explicitly sends the NP-witness to the verifier. The answer is three-fold. First, the proof length in our $\frac{1}{2}$-SNARK is smaller than the witness size for sufficiently large instances ($N \geq 2^{13}$ for Shockwave and $N \geq 2^{18}$ for Brakedown). Second, our $\frac{1}{2}$-SNARK actually *can* save the verifier time relative to the trivial proof system for structured R1CS instances. In particular, if the R1CS is data parallel, then the verifier can run in time proportional to the size of a single sub-computation, independent of the number of times the sub-computation is performed (this is entirely analogous to how prior proof systems save the verifier work for structured computations [9,68]). Third, we expect that the reported performance results are indicative of the performance of future zero-knowledge implementations.

Unless we specify otherwise, we run our experiments in this section on an Azure Standard F16s_v2 virtual machine (16 vCPUs, 32 GB memory) with Ubuntu 20.10. We report results from a single-threaded configuration since not all our baselines leverage multiple cores. As with prior work [14,32,63,65], we vary the size of the R1CS instance by varying the number of constraints and variables m and maintain the ratio n/m to approximately 1.

Performance of Shockwave's and Brakedown's $\frac{1}{2}$-SNARK Scheme. Prior state-of-the-art $\frac{1}{2}$-SNARK schemes include: Ligero [4], Bulletproofs [27], Aurora [14], SpartanNIZK [63], and Lakonia [65]. Note that for uniform computations (e.g., data-parallel circuits), Hyrax [74] and STARK [9] are SNARKs, but for computations without any structure, they are $\frac{1}{2}$-SNARKs. We do not compare with Ligero++ [17] since its source code is not public. Broadly speaking, Ligero++ has shorter proofs than Ligero at the cost of a slower prover, so its prover will be significantly slower than both Brakedown and Shockwave. We do not report results from Bulletproofs or STARK as they feature a more expensive prover than other baselines considered here [14,27]. Hyrax [74] supports only layered arithmetic circuits, so as used in prior work [63] for comparison purposes, we translate R1CS to depth-1 arithmetic circuits (without any structure). None of the $\frac{1}{2}$-SNARKs we consider require a preprocessing step for the verifier.

In the full version of the paper [40], we provide a rough comparison with Wolverine [75] and Mac'n'Cheese [6], which unlike schemes considered here do *not* support proof sizes sub-linear in the instance size. Another potential baseline is Virgo [78], which like Hyrax [74] applies only to low-depth circuits as they both share the same information-theoretic component [33,39,72,76].

For Aurora and Ligero, we use their open-source implementations from `libiop` [52], configured to provide provable security. For Hyrax, we use its reference (i.e., unoptimized) implementation [51]. For SpartanNIZK, we use its open-source implementation [3]. Unless we specify otherwise, we use 256-bit prime fields. Hyrax uses `curve25519` and SpartanNIZK uses `ristretto255` [2,42] for a group where the discrete-log problem is hard, so R1CS instances are defined over the scalar field of these curves. For Aurora and Ligero, we use the 256-bit prime field option in `libiop`. Finally, our schemes use the scalar field of `BLS12-381`, which supports FFTs (Brakedown does not need FFTs but Shockwave does). However, we note that none of these implementations leverages the specifics of the prime field to speed up scalar arithmetic.

We first experiment with Brakedown and Shockwave and their baselines with varying R1CS instance sizes up to 2^{20} constraints defined over a 256-bit prime field. Figures 2a, 2b, and 2c depict respectively the prover time, the verifier time, and the proof size from these experiments. We find the following.

- Brakedown's and Shockwave's provers are faster than prior work at all instance sizes we measure. Compared to baseslines that are plausibly post-quantum secure (Ligero and Aurora), Brakedown's and Shockwave's provers are over an order of magnitude faster.
- Brakedown's proof size is larger than other depicted systems except for Ligero. Still, its proofs are substantially smaller than the size of the NP-witness for instance sizes $N \geq 2^{18}$. Shockwave provides shorter proofs than Brakedown as well as prior post-quantum secure baselines (Ligero and Aurora). Note that Aurora has asymptotically shorter proofs than Shockwave, and hence the proof size comparison would "cross over" at larger instance sizes). Shockwave's proof sizes are smaller than that of the NP-witness for instance sizes $N \geq 2^{13}$.
- Despite their larger proofs, Brakedown's and Shockwave's verifiers are competitive with those of SpartanNIZK and Lakonia, and is well over an order of magnitude faster than the plausibly post-quantum secure baselines.

Performance for Larger Instance Sizes. To demonstrate Brakedown's and Shockwave's scalability to larger instance sizes, we experiment with them and SpartanNIZK for instance sizes beyond 2^{20} constraints.

For these larger-scale experiments, we use an Azure Standard F32s_v2 VM which has 32 vCPUs and 64 GB memory. Figures 3a, 3b, and 3c depict results from these larger-scale experiments. Our findings from these experiments are similar to results from the smaller-scale results.

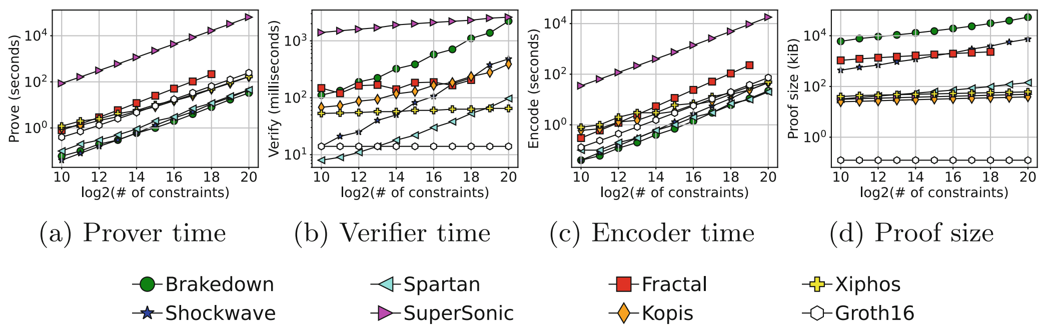

Fig. 4. SNARK results (Sect. 6.2); lower is better. Brakedown, Shockwave, and Fractal are plausibly post-quantum secure. Fractal results are incomplete because the prover and encoder ran out of memory for larger instances.

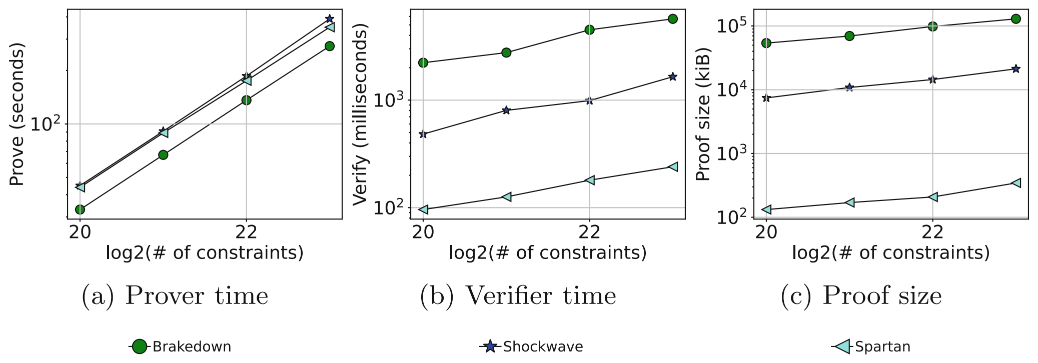

Fig. 5. SNARK results (Sect. 6.2); lower is better. Brakedown and Shockwave are plausibly post-quantum secure.

Performance over Small Fields. To demonstrate flexibility with different field sizes, we also run Brakedown and Shockwave with a prime field where the prime modulus is 128 bits. For the latter case, our choice of parameters achieve at least 100 bits security. We depict these results together with results from our larger-scale experiments (Figs. 3a, 3c, and 3b).

Recall that our asymptotic results require $|\mathbb{F}| > \exp(\Omega(\lambda))$ to achieve a linear-time prover, because if the field is smaller than this, certain parts of the protocol need to be repeated $\omega(1)$ times to drive the soundness error below $\exp(-\lambda)$. However, Brakedown and Shockwave are quite efficient over small fields. The reason is that only some parts of the protocol need to be repeated to drive the soundness error below $\exp(-\lambda)$ and those repetitions produce only low-order effects on the prover's runtime and the proof length. This means that for a fixed security level, our prover is faster over small fields than large fields, because the effect of faster field arithmetic dominates the overhead due to the need to repeat parts of the protocol to drive down soundness error. Similar observations appear in prior work [9].

Performance of Brakedown's and Shockwave's SNARK Scheme. Prior state-of-the-art schemes SNARKs include: Spartan [63], SuperSonic [28], Fractal [32], Kopis [65], and Xiphos [65]. We also give results for Groth16 [41], a SNARK whose preprocessing phase involves secret randomness and is therefore *not* transparent, but which gives very short proofs, fast verification, and fast proving times compared to other systems with similar properties.

For Fractal, we use its open-source implementations from `libiop` [52], configured to provide provable security. For SuperSonic, there is no prior implementation, so we use prior estimates of their costs based on microbenchmarks (See [65] for a detailed discussion of how they estimate these costs). For Spartan, we use its open-source implementation [3]. For Groth16, we benchmark the libsnark implementation using the BN254 elliptic curve [53]. For preprocessing costs, we ignore the use of "untrusted assistant" technique [65], which applies to all schemes considered here except Groth16.

Figures 4a, 4b, 4c, 4d depict respectively the prover time, the verifier time, the verifier's preprocessing time, and the proof size for varying R1CS instance sizes for our schemes and their baselines. We find the following from these results.

- Brakedown achieves the fastest prover at instance sizes we measure. Shockwave's prover is slower than Brakedown's, both asymptotically and concretely, but Shockwave's prover is still over an order of magnitude faster than prior plausibly post-quantum secure SNARKs (namely Fractal [32]).
- Brakedown and Shockwave have the largest proof sizes amongst the displayed proof systems, but for large enough R1CS instances their proof sizes are sublinear in the size of the NP-witness ($N > 2^{16}$ for Shockwave and $N \geq 2^{22}$ for Brakedown).
- Brakedown's verifier is slower than Shockwave and most other schemes, particularly Xiphos [65] which is specifically designed for achieving a fast verifier. However, Shockwave's verifier is competitive with prior plausibly post-quantum secure SNARKs.
- Brakedown's and Shockwave's preprocessing costs for the verifier are competitive with those of prior high-speed SNARKs such as Spartan [63] and Xiphos [65], and an order of magnitude faster than the prior post-quantum secure SNARK (Fractal).

Performance for Larger Instance Sizes. To demonstrate Brakedown's and Shockwave's scalability to larger instance sizes, we experiment with them and Spartan for instance sizes beyond 2^{20} constraints.

For these larger-scale experiments, we use an Azure Standard F64s_v2 VM which has 64 vCPUs and 128 GB memory. Figures 5a, 5c, and 5b depict results from these larger-scale experiments. Our findings from these experiments are similar to results from the smaller-scale results.

References

1. ethSTARK. https://github.com/starkware-libs/ethSTARK
2. The Ristretto group. https://ristretto.group/
3. Spartan: High-speed zkSNARKs without trusted setup. https://github.com/Microsoft/Spartan
4. Ames, S., Hazay, C., Ishai, Y., Venkitasubramaniam, M.: Ligero: lightweight sublinear arguments without a trusted setup. In: CCS (2017)
5. Applebaum, B., Haramaty, N., Ishai, Y., Kushilevitz, E., Vaikuntanathan, V.: Low-complexity cryptographic hash functions. In: ITCS (2017)
6. Baum, C., Malozemoff, A.J., Rosen, M., Scholl, P.: Mac'n'cheese: zero-knowledge proofs for arithmetic circuits with nested disjunctions. Cryptology ePrint Archive, Report 2020/1410 (2020)
7. Belling, A., Soleimanian, A.: Vortex: building a lattice-based snark scheme with transparent setup. Cryptology ePrint Archive, Paper 2022/1633 (2022)
8. Ben-Sasson, E., Bentov, I., Horesh, Y., Riabzev, M.: Fast reed-solomon interactive oracle proofs of proximity. In: ICALP (2018)
9. Ben-Sasson, E., Bentov, I., Horesh, Y., Riabzev, M.: Scalable zero knowledge with no trusted setup. In: Boldyreva, A., Micciancio, D. (eds.) CRYPTO 2019. LNCS, vol. 11694, pp. 701–732. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-26954-8_23
10. Ben-Sasson, E., Carmon, D., Ishai, Y., Kopparty, S., Saraf, S.: Proximity gaps for Reed-Solomon codes. In: FOCS (2020)
11. Ben-Sasson, E., Carmon, D., Kopparty, S., Levit, D.: Elliptic Curve Fast Fourier Transform (ECFFT) part I: fast polynomial algorithms over all finite fields. Electronic Colloquium on Computational Complexity, Report 2021/103 (2021)
12. Ben-Sasson, E., Carmon, D., Kopparty, S., Levit, D.: Scalable and transparent proofs over all large fields, via elliptic curves. Electronic Colloquium on Computational Complexity, Report 2022/110 (2022)
13. Ben-Sasson, E., Chiesa, A., Genkin, D., Tromer, E.: Fast reductions from RAMs to delegatable succinct constraint satisfaction problems: extended abstract. In: ITCS (2013)
14. Ben-Sasson, E., Chiesa, A., Riabzev, M., Spooner, N., Virza, M., Ward, N.P.: Aurora: transparent succinct arguments for R1CS. In: Ishai, Y., Rijmen, V. (eds.) EUROCRYPT 2019. LNCS, vol. 11476, pp. 103–128. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-17653-2_4
15. Ben-Sasson, E., Chiesa, A., Spooner, N.: Interactive oracle proofs. In: Hirt, M., Smith, A. (eds.) TCC 2016. LNCS, vol. 9986, pp. 31–60. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53644-5_2
16. Bernstein, D.J., Doumen, J., Lange, T., Oosterwijk, J.-J.: Faster batch forgery identification. Cryptology ePrint Archive, Paper 2012/549 (2012)
17. Bhadauria, R., Fang, Z., Hazay, C., Venkitasubramaniam, M., Xie, T., Zhang, Y.: Ligero++: a new optimized sublinear IOP. In: CCS, pp. 2025–2038 (2020)
18. Bitansky, N., Canetti, R., Chiesa, A., Tromer, E.: From extractable collision resistance to succinct non-interactive arguments of knowledge, and back again. In: ITCS (2012)
19. Bitansky, N., Canetti, R., Chiesa, A., Tromer, E.: Recursive composition and bootstrapping for SNARKs and proof-carrying data. In: STOC (2013)
20. Bootle, J., Cerulli, A., Ghadafi, E., Groth, J., Hajiabadi, M., Jakobsen, S.K.: Linear-time zero-knowledge proofs for arithmetic circuit satisfiability. In: Takagi,

T., Peyrin, T. (eds.) ASIACRYPT 2017. LNCS, vol. 10626, pp. 336–365. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-70700-6_12

21. Bootle, J., Chiesa, A., Groth, J.: Linear-time arguments with sublinear verification from tensor codes. In: Pass, R., Pietrzak, K. (eds.) TCC 2020. LNCS, vol. 12551, pp. 19–46. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-64378-2_2
22. Bootle, J., Chiesa, A., Guan, Z., Liu, S.: Linear-time probabilistic proofs over every field. Cryptology ePrint Archive, Paper 2022/1056 (2022)
23. Bootle, J., Chiesa, A., Liu, S.: Zero-knowledge succinct arguments with a linear-time prover. ePrint Report 2020/1527 (2020)
24. Bootle, J., Lyubashevsky, V., Nguyen, N.K., Seiler, G.: A non-PCP approach to succinct quantum-safe zero-knowledge. In: Micciancio, D., Ristenpart, T. (eds.) CRYPTO 2020. LNCS, vol. 12171, pp. 441–469. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-56880-1_16
25. Braun, B., Feldman, A.J., Ren, Z., Setty, S., Blumberg, A.J., Walfish, M.: Verifying computations with state. In: SOSP (2013)
26. Bröker, R., Stevenhagen, P.: Efficient CM-constructions of elliptic curves over finite fields. Math. Comp. **76**(260), 2161–2179 (2007)
27. Bünz, B., Bootle, J., Boneh, D., Poelstra, A., Wuille, P., Maxwell, G.: Bulletproofs: short proofs for confidential transactions and more. In: S&P (2018)
28. Bünz, B., Fisch, B., Szepieniec, A.: Transparent SNARKs from DARK compilers. In: Canteaut, A., Ishai, Y. (eds.) EUROCRYPT 2020. LNCS, vol. 12105, pp. 677–706. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-45721-1_24
29. Chen, B., Bünz, B., Boneh, D., Zhang, Z.: Hyperplonk: plonk with linear-time prover and high-degree custom gates. In: Hazay, C., Stam, M. (eds.) EUROCRYPT 2023. LNCS, vol. 14005, pp. 499–530. Springer, Cham (2023). https://doi.org/10.1007/978-3-031-30617-4_17
30. Chiesa, A., Forbes, M.A., Spooner, N.: A zero knowledge sumcheck and its applications. CoRR, abs/1704.02086 (2017)
31. Chiesa, A., Hu, Y., Maller, M., Mishra, P., Vesely, N., Ward, N.: Marlin: preprocessing zkSNARKs with universal and updatable SRS. In: Canteaut, A., Ishai, Y. (eds.) EUROCRYPT 2020. LNCS, vol. 12105, pp. 738–768. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-45721-1_26
32. Chiesa, A., Ojha, D., Spooner, N.: Fractal: post-quantum and transparent recursive proofs from holography. In: Canteaut, A., Ishai, Y. (eds.) EUROCRYPT 2020. LNCS, vol. 12105, pp. 769–793. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-45721-1_27
33. Cormode, G., Mitzenmacher, M., Thaler, J.: Practical verified computation with streaming interactive proofs. In: ITCS (2012)
34. Druk, E., Ishai, Y.: Linear-time encodable codes meeting the Gilbert-Varshamov bound and their cryptographic applications. In: ITCS, pp. 169–182 (2014)
35. Fiat, A., Shamir, A.: How To prove yourself: practical solutions to identification and signature problems. In: Odlyzko, A.M. (ed.) CRYPTO 1986. LNCS, vol. 263, pp. 186–194. Springer, Heidelberg (1987). https://doi.org/10.1007/3-540-47721-7_12
36. Gelfand, S.I., Dobrushin, R.L., Pinsker, M.S.: On the complexity of coding. pp. 177–184 (1973)
37. Gentry, C., Wichs, D.: Separating succinct non-interactive arguments from all falsifiable assumptions. In: STOC, pp. 99–108 (2011)
38. Goldreich, O., Kahan, A.: How to construct constant-round zero-knowledge proof systems for NP. J. Cryptol. **9**(3), 167–189 (1996)

39. Goldwasser, S., Kalai, Y.T., Rothblum, G.N.: Delegating computation: Interactive proofs for muggles. In: STOC (2008)
40. Golovnev, A., Lee, J., Setty, S., Thaler, J., Wahby, R.S.: Brakedown: linear-time and post-quantum snarks for r1cs. Cryptology ePrint Archive, Paper 2021/1043 (2021)
41. Groth, J.: On the size of pairing-based non-interactive arguments. In: Fischlin, M., Coron, J.-S. (eds.) EUROCRYPT 2016. LNCS, vol. 9666, pp. 305–326. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49896-5_11
42. Hamburg, M.: Decaf: eliminating cofactors through point compression. In: Gennaro, R., Robshaw, M. (eds.) CRYPTO 2015. LNCS, vol. 9215, pp. 705–723. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-47989-6_34
43. Hazay, C., Lindell, Y.: Efficient Secure Two-Party Protocols: Techniques and Constructions. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-14303-8
44. Housni, Y.E., Botrel, G.: EdMSM: multi-scalar-multiplication for SNARKs and faster montgomery multiplication. Cryptology ePrint Archive, Paper 2022/1400 (2022)
45. Kate, A., Zaverucha, G.M., Goldberg, I.: Constant-size commitments to polynomials and their applications. In: Abe, M. (ed.) ASIACRYPT 2010. LNCS, vol. 6477, pp. 177–194. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-17373-8_11
46. Kattis, A., Panarin, K., Vlasov, A.: RedShift: transparent SNARKs from list polynomial commitment IOPs. Cryptology ePrint Archive, Report 2019/1400 (2019)
47. Kilian, J.: A note on efficient zero-knowledge proofs and arguments (extended abstract). In: STOC (1992)
48. Kothapalli, A., Setty, S., Tzialla, I.: Nova: recursive zero-knowledge arguments from folding schemes. In: Dodis, Y., Shrimpton, T. (eds.) CRYPTO 2022. LNCS, vol. 13510, pp. 359–388. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-15985-5_13
49. Lee, J.: Dory: efficient, transparent arguments for generalised inner products and polynomial commitments. Cryptology ePrint Archive, Report 2020/1274 (2020)
50. Lee, J., Nikitin, K., Setty, S.: Replicated state machines without replicated execution. In: S&P (2020)
51. libfennel. Hyrax reference implementation. https://github.com/hyraxZK/fennel
52. libiop. A C++ library for IOP-based zkSNARK. https://github.com/scipr-lab/libiop
53. libsnark. A C++ library for zkSNARK proofs. https://github.com/scipr-lab/libsnark
54. Lund, C., Fortnow, L., Karloff, H., Nisan, N.: Algebraic methods for interactive proof systems. In: FOCS, October 1990
55. Micali, S.: CS proofs. In: FOCS (1994)
56. Mitzenmacher, M., Upfal, E.: Probability and Computing: Randomization and Probabilistic Techniques in Algorithms and Data Analysis. Cambridge University Press, Cambridge (2017)
57. O'Connor, J., Aumasson, J.-P., Neves, S., Wilcox-O'Hearn, Z.: BLAKE3: one function, fast everywhere, February 2020. https://github.com/BLAKE3-team/BLAKE3-specs/blob/master/blake3.pdf
58. Ozdemir, A., Brown, F., Wahby, R.S.: Unifying compilers for SNARKs, SMT, and more. Cryptology ePrint Archive, Report 2020/1586 (2020)
59. Parno, B., Gentry, C., Howell, J., Raykova, M.: Pinocchio: nearly practical verifiable computation. In: S&P, May 2013

60. Pippenger, N.: On the evaluation of powers and related problems. In: SFCS (1976)
61. Reingold, O., Rothblum, G.N., Rothblum, R.D.: Constant-round interactive proofs for delegating computation. In: STOC, pp. 49–62 (2016)
62. Ron-Zewi, N., Rothblum, R.D.: Proving as fast as computing: succinct arguments with constant prover overhead. In: STOC (2022)
63. Setty, S.: Spartan: efficient and general-purpose zkSNARKs without trusted setup. In: Micciancio, D., Ristenpart, T. (eds.) CRYPTO 2020. LNCS, vol. 12172, pp. 704–737. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-56877-1_25
64. Setty, S., Angel, S., Gupta, T., Lee, J.: Proving the correct execution of concurrent services in zero-knowledge. In: OSDI, October 2018
65. Setty, S., Lee, J.: Quarks: quadruple-efficient transparent zkSNARKs. Cryptology ePrint Archive, Report 2020/1275 (2020)
66. Setty, S., Vu, V., Panpalia, N., Braun, B., Blumberg, A.J., Walfish, M.: Taking proof-based verified computation a few steps closer to practicality. In: USENIX Security, August 2012
67. Spielman, D.A.: Linear-time encodable and decodable error-correcting codes. IEEE Trans. Inf. Theory **42**(6), 1723–1731 (1996)
68. Thaler, J.: Time-optimal interactive proofs for circuit evaluation. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013. LNCS, vol. 8043, pp. 71–89. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40084-1_5
69. Thaler, J.: Proofs, arguments, and zero-knowledge (2020). http://people.cs.georgetown.edu/jthaler/ProofsArgsAndZK.html
70. Valiant, P.: Incrementally verifiable computation or proofs of knowledge imply time/space efficiency. In: Canetti, R. (ed.) TCC 2008. LNCS, vol. 4948, pp. 1–18. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-78524-8_1
71. Vlasov, A., Panarin, K.: Transparent polynomial commitment scheme with polylogarithmic communication complexity. Cryptology ePrint Archive, Report 2019/1020 (2019)
72. Wahby, R.S., et al.: Full accounting for verifiable outsourcing. In: CCS (2017)
73. Wahby, R.S., Setty, S., Ren, Z., Blumberg, A.J., Walfish, M.: Efficient RAM and control flow in verifiable outsourced computation. In: NDSS (2015)
74. Wahby, R.S., Tzialla, I., Shelat, A., Thaler, J., Walfish, M.: Doubly-efficient zkSNARKs without trusted setup. In: S&P (2018)
75. Weng, C., Yang, K., Katz, J., Wang, X.: Wolverine: fast, scalable, and communication-efficient zero-knowledge proofs for Boolean and arithmetic circuits. Cryptology ePrint Archive, Report 2020/925 (2020)
76. Xie, T., Zhang, J., Zhang, Y., Papamanthou, C., Song, D.: Libra: succinct zero-knowledge proofs with optimal prover computation. In: Boldyreva, A., Micciancio, D. (eds.) CRYPTO 2019. LNCS, vol. 11694, pp. 733–764. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-26954-8_24
77. Xie, T., Zhang, Y., Song, D.: Orion: zero knowledge proof with linear prover time. In: Dodis, Y., Shrimpton, T. (eds.) CRYPTO 2022. LNCS, vol. 13510, pp. 299–328. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-15985-5_11
78. Zhang, J., Xie, T., Zhang, Y., Song, D.: Transparent polynomial delegation and its applications to zero knowledge proof. In: S&P (2020)
79. Zhang, Y., Genkin, D., Katz, J., Papadopoulos, D., Papamanthou, C.: vSQL: Verifying arbitrary SQL queries over dynamic outsourced databases. In: S&P (2017)