19

Bird's Eye View of Practical Arguments

We have covered four approaches to the construction of practical SNARKs. In each of the four, an underlying information-theoretically secure protocol is combined with cryptography to yield an argument. The first approach is based on the interactive proof for arithmetic circuit evaluation of Section 4.6 (the GKR protocol), the second is based on the MIPs for circuit or R1CS satisfiability of Sections 8.2 and 8.4, the third is based on the constant-round polynomial IOP for circuit or R1CS satisfiability of Section 10.3, and the fourth is based on the linear PCP of Section 17.4. We presented a unified view of the first three approaches, and the pros and cons of each, in Section 10.6, via the lens of polynomial IOPs.

We have also covered a fifth approach to argument design, based on commit-and-prove techniques (Section 13.1), which can be viewed as combining a trivial static (i.e., **NP**) proof system with cryptographic commitments. Commit-and-prove based arguments have been studied in several works, e.g., [23], [107], [248]. These arguments are not succinct, and recent works on this approach yield interactive protocols; for both of these reasons, these arguments are not SNARKs.

For each of the first three approaches (IP-based, MIP-based, and constant-round-polynomial-IOP-based), one can combine the information-theoretically secure protocol with any extractable polynomial commitment scheme of the protocol designer's choosing to obtain a succinct argument (there is essentially just one technique to to turn linear PCPs into publicly-verifiable SNARKs, based on pairings and very similar to KZG polynomial commitments, see Section 17.5). For the IP-based and MIP-based argument systems, the polynomial commitment scheme must allow committing to *multilinear* polynomials. For the IOP-based argument system, the polynomial commitment scheme must allow committing to *univariate* polynomials. Of course, the resulting argument system will inherit the cryptographic and setup assumptions as well as any efficiency bottlenecks of the chosen polynomial commitment scheme.

We have in turn covered three broad approaches to polynomial commitment schemes in this survey, though some of these approaches have multiple instantiations with various cost tradeoffs. The first is via IOPs combined with Merkle hashing, where we saw FRI in Section 10.4.2 and Ligero- and Brakedown-commitments in Section 10.5. The second is based on transparent Σ-protocols that assume hardness of the discrete logarithm problem, where we saw Hyrax-commit (Section 14.3), Bulletproofs (Section 14.4), and Dory (Section 15.4).[1] The third is based on the approach of KZG [164] and uses pairings and a trusted setup (Section 15.2). Below, we call these respective approaches to polynomial commitments "IOP-based", "discrete-log-based", and "KZG-based". We discussed the pros and cons of the various polynomial commitment schemes in Section 16.3.

19.1 A Taxonomy of SNARKs

The research literature on practical succinct arguments is a veritable zoo of built systems and theoretical protocols. In this section, we attempt

[1] We also saw that it is possible to combine various commitment schemes to obtain different cost tradeoffs, e.g., Section 15.4. We omit such combinations from this section to avoid a combinatorial explosion of commitment schemes to discuss.

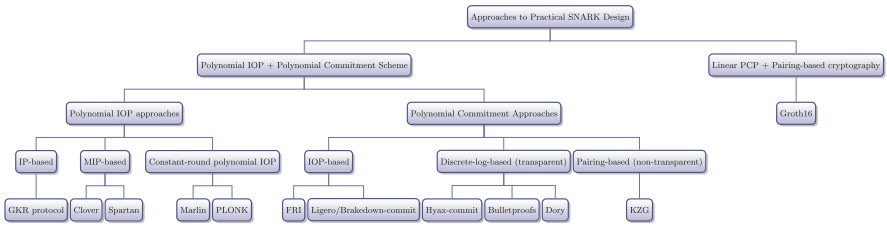

Figure 19.1: Our taxonomy of SNARK design. Leaves depict selected example protocols covered or discussed in this survey. Every combination of polynomial IOP and polynomial commitment scheme yields a SNARK, though constant-round polynomial IOPs need a commitment scheme for univariate polynomials, while IPs and MIPs need one for multilinear polynomials. SNARKs using transparent polynomial commitments are transparent. SNARKs using IOP-based polynomial commitments are plausibly post-quantum.

to tame this zoo with a coherent taxonomy of the primary approaches that have been pursued.

Outside of the linear-PCP-based SNARKs, most known SNARKs are obtained by combining some IP, MIP, or constant-round polynomial IOP with a polynomial commitment scheme. As we have covered at least six polynomial commitment schemes in this survey, this yields at least 18 possible SNARKs (even ignoring the fact that there are multiple MIPs and constant-round polynomial IOPs to choose from). Most of these combinations have been explored; below, we list which implemented systems use which combination.[2] Our taxonomy is depicted in Figure 19.1.

(a) IPs combined with FRI-based (multilinear) polynomial commitments (Section 10.4.5) were explored in [256], producing a system called Virgo.

(b) IPs combined with discrete-log-based (multilinear) polynomial commitments (Bulletproofs, and Hyrax-commit) were explored in [244], producing a system called Hyrax.

[2]This list is surely not exhaustive.

(c) IPs combined with KZG-based (multilinear) polynomial commitments were explored in [251], [257], [258], producing systems called zk-vSQL and Libra.

(d) MIPs combined with many different multilinear polynomial commitments were explored in [137], [226], [230], producing systems including Spartan, Xiphos, Kopis, Brakedown, and Shockwave. Spartan, Kopis, and Xiphos use various discrete-logarithm-based multilinear commitments, while Brakedown naturally uses the Brakedown-commitment and Shockwave the Ligero-commitment.

(e) Constant-round polynomial IOPs combined with FRI-based (univariate) polynomial commitments were explored in a series of works, most recently [43], [100], [165], producing systems called Aurora, Fractal, and Redshift. Other related works in this series include [33]–[36], [41], [221].

(f) Constant-round polynomial IOPs combined with KZG-based (univariate) polynomial commitments were explored in popular systems called Marlin [98] and PlonK [123]. Marlin uses the polynomial IOP for R1CS from Section 10.3, while PlonK gives a different polynomial IOP, for circuit-satisfiability. A predecessor to these works is Sonic [187].

"Halo 2"[3] combines the PlonK constant-round polynomial IOP with the Bulletproofs polynomial commitment scheme. "PlonKy2"[4] uses a FRI-based polynomial commitment scheme rather than Bulletproofs.

(g) Ligero [7] combines a constant-round polynomial IOP with the polynomial commitment of the same name, and Ligero++ [53] replaces the polynomial commitment with a "combination" of Ligero's commitment and FRI.

(h) A very large number of systems have been derived from the linear PCP of Genarro, Gentry, Parno, and Raykova [125] (Section 17.5).

[3] https://zcash.github.io/halo2/.
[4] https://github.com/mir-protocol/PlonKy2/blob/main/PlonKy2.pdf.

These include [42], [203]. The most popular variant of the SNARK derived from GGPR's linear PCP is due to Groth [142], who obtained a proof length of just 3 elements of a pairing-friendly group, and proved the SNARK secure in the Generic Group Model that was briefly discussed in Section 15.2 ([117] extended the security proof to the Algebraic Group Model). This variant is colloquially referred to as Groth16.

More SNARKs via composition. On top of the taxonomy of SNARKs delineated above, one can take any two SNARKs designed via one of the 18 above approaches, and compose them one or more times. As discussed in Section 18.1, by taking a "fast-prover, larger-proof" SNARK and composing it with a "slower prover-smaller proof" SNARK, one can in principle obtain a "best-of-both-worlds" SNARK with a fast prover and small proofs. Such compositions are growing increasingly popular and already yield state-of-the-art performance.

To name some recent examples, PlonKy2 self-composes the SNARK obtained by combining the PlonK polynomial IOP with the FRI polynomial commitment scheme. In the first SNARK application, FRI can be configured to have a fast prover but to generate a large proof π. Since π is large, it is not actually sent to the verifier. Rather, subsequent applications of the same SNARK are used to establish knowledge of π. Since these later applications of the SNARK are applied to a relatively small computation (the procedure for *verifying* π), FRI can be configured in these later applications to have a slower prover and smaller proofs.

Relatedly, Polygon Hermez is composing such a FRI-based SNARK with Groth16, in order to inherit Groth16's attractive verification costs, while keeping both the prover time and size of the trusted setup smaller than in a direct application of Groth16 to the original statement being proved [25] (the use of Groth16 does relinquish the plausible post-quantum security and transparency of the FRI-based SNARK).

As other examples, Orion [253] composes Brakedown with Virgo to reduce proof size. deVirgo [252] in turn composes Virgo with Groth16. Filecoin uses a technique called SnarkPack [124] to aggregate many Groth16 proofs into one; such aggregation of SNARK proofs can be

viewed as a form of SNARK composition (see Section 18.3). zkSync has similarly used recursive aggregation of PlonK proofs since 2020.

Other approaches. There are a handful of approaches to the design of arguments that do not necessarily fall into the categories above. One example is called MPC-in-the-head, which takes any secure multiparty computation (MPC) protocol[5] and transforms it into a (zero-knowledge) IOP [7], [127], [157].[6] The IOP can in turn be transformed into a non-interactive argument via Merkle-hashing and the Fiat-Shamir transformation, as we have described in this survey.

Arguments derived via MPC-in-the-head typically have a cost profile loosely analogous to commit-and-prove arguments: much larger proof sizes and higher verifier costs than the approaches above, but they can have good concrete costs on small instance sizes, and good prover runtimes. This has led, for example, to an interesting family of candidate post-quantum secure digital signatures, called Picnic [93], [106], [162], [163], [166].[7]

We do not cover MPC-in-the-head because, in our view, it is not truly distinct from the approaches covered in this survey. In particular, all known *succinct* arguments (i.e., with sublinear proof size) that were originally discovered or presented via the MPC-in-the-head framework in fact comprise a polynomial IOP combined with one of the IOP-based polynomial commitment schemes that we have covered [7].

[5] An MPC protocol allows $t \geq 2$ parties to compute some function f of their inputs, say, $f(x_1, \ldots, x_t)$ where x_i is the i'th party's input. Very roughly speaking, the guarantee of an MPC protocol is that each party i learns no other information about the other parties' inputs, other than $f(x_1, \ldots, x_t)$.

[6] Very roughly speaking, the IOP is obtained as follows. If the IOP prover claims to know a witness w such that $\mathcal{C}(w) = 1$, it simulates, in its own head, the secret-sharing of w amongst several parties. It then simulates an MPC protocol for evaluating $\mathcal{C}$ on w, using verifier-supplied randomness within the MPC protocol. The IOP proof string is then a (claimed) transcript of the MPC protocol. The IOP verifier inspects the proof string to try to ascertain whether it is indeed a valid transcript for the MPC protocol. The security properties of the MPC protocol are used to ensure that the resulting IOP is complete, sound, and zero-knowledge.

[7] Other related techniques also derive zero-knowledge proofs from MPC protocols [114], [149], [159], with broadly similar cost profiles to MPC-in-the-head (long proofs and verification time, but good prover runtime and small hidden constants).

Another example approach not covered in this survey is that linear PCPs can be combined with *non*-pairing-based cryptosystems to yield designated-verifier (i.e., non-publicly-verifiable) SNARKs, including some based on the assumed hardness of lattice problems that are plausibly post-quantum secure [56]. To date, this approach has not led to practical protocols.

19.2 Pros and Cons of the Approaches

Every combination of {IP, MIP, constant-round polynomial IOP} and polynomial commitment scheme naturally inherits the pros and cons of the two components of the combination. Sections 10.6 and 16.3 respectively discussed the pros and cons of the individual components. In this section, we aim to do the same for the various combinations, as well as for SNARKs derived from linear PCPs.

Approaches minimizing proof size. There are two approaches that achieve proofs consisting of a constant number of group elements, captured in items (f) and (h) of the previous section—namely, constant-round polynomial IOPs combined with KZG-based polynomial commitments, and linear PCPs (transformed into SNARKs using pairing-based cryptography). The linear PCP approach is the ultimate winner in proof size, as its proofs consist of as few as 3 group elements [142]. For comparison, Marlin [98] (which uses the former approach), produces proofs that are roughly 4 times larger than that of Groth's SNARK [142].

The downsides of the two approaches are also related. First, both require a trusted setup (as they make use of a structured reference string), which produces toxic waste (also called a trapdoor) that must be discarded to prevent forgeability of proofs. In the case of IOPs combined with KZG-based polynomial commitments, the downsides of the SRS are not as severe as for linear PCPs, for two reasons. First, the SRS for the former approach is *universal*: a single SRS can be used for *any* R1CS-satisfiability or circuit-satisfiability instance up to some specified size bound. This is because the SRS simply consists of encodings of powers of a random field element τ, and hence is

independent of the circuit or R1CS instance. In contrast, the SRS in the linear PCP approach is *computation-specific*: in addition to including encodings of powers-of-τ, the SRS in the linear PCP approach also has to include encodings of evaluations of univariate polynomials capturing the wiring pattern of the circuit or the matrix entries specifying the R1CS instance.[8] Second, the SRS for the former approach is *updatable* (see Section 16.3 for discussion of this notion), while the SRS for the linear PCP approach is not, again owing to the fact that the SRS contains elements other than encodings of powers-of-τ.

The second downside of both of these two approaches is that they are computationally expensive for the prover, for two reasons. First, in both approaches, the prover needs to perform FFTs or polynomial division over vectors or polynomials of size proportional to the circuit size S, or number K of nonzero entries of constraints in the R1CS instance. This is time-intensive as well as highly space intensive and difficult to parallelize and distribute [250]. Second, in both approaches the prover also needs to perform several multi-exponentiations of size $\Theta(S)$ or $\Theta(K)$ in a pairing-friendly group. See Section 16.3 for discussion of the concrete costs of these operations.

SNARKs such as Marlin and PlonK that are derived from constant-round polynomial IOPs have a significantly slower prover than Groth's SNARK for a given circuit or R1CS size.[9] This increased prover cost

[8] One can render universal a SNARK with a computation-specific SRS by applying the SNARK to a so-called universal circuit, which takes as input both a description of another circuit $\mathcal{C}$ and an input-witness pair (x, w) for $\mathcal{C}$ and evaluates $\mathcal{C}$ on (x, w). This introduces significant overhead, despite several mitigation efforts [46], [172]; see [243] for some concrete measurements of overhead.

[9] In Groth's SNARK, the prover performs three multi-exponentiations in $\mathbb{G}_1$ and one in $\mathbb{G}_2$, all of size linear in the number of gates of the circuit or number of constraints in the R1CS. With popular pairing-friendly groups such as BLS12-381, this cost is comparable to that of applying KZG polynomial commitments to roughly six polynomials of this size. Marlin and PlonK require the prover to commit to more and/or larger polynomials than this. This is especially so as these systems are typically applied in the holographic setting (Sections 10.3.2 and 16.2), whereby polynomials capturing the "wiring" of the circuit or R1CS are committed in pre-processing, to enable sublinear verification time. These pre-processing polynomials are larger than the others by a constant factor, and there are several of them, and the prover must reveal evaluations of these committed polynomials as part of the SNARK proof. In contrast, Groth's SNARK, with its circuit-specific pre-processing,

can be mitigated in certain applications by the fact that polynomial-IOP derived SNARKs, compared to linear-PCP derived SNARKs such as Groth's, have more flexibility in the intermediate representation used—see Section 19.3.3 for details.

In the remainder of this section, we describe broad tradeoffs of the remaining approaches.

Transparency. In contrast to the two approaches that minimize proof size, all of the remaining approaches are *transparent* unless they choose to use KZG-based polynomial commitments. That is, they use a uniform reference string (URS) rather than a structured reference string, and hence no toxic waste is produced. Transparency of the SNARK is totally determined by the polynomial commitment scheme used—if the commitment scheme uses a URS, then the entire SNARK uses a URS.

Post-quantum security. The approaches that are plausibly post-quantum secure are comprised of those that utilize an IOP-based polynomial commitment (FRI, Ligero, Brakedown).[10] That is, quantum security is determined entirely by the polynomial commitment scheme–IOPs are plausibly post-quantum, but the other two classes of polynomial commitments are not, due to their reliance on the hardness of discrete log.

Dominant contributor to cost: polynomial commitments. When MIPs and constant-round polynomial IOPs are combined with any polynomial commitment, it is the polynomial commitment that typically dominates the most relevant costs: prover time, proof length, and verifier time (the lone exception is that, if an MIP is combined with KZG commitments, it is the MIP and not the polynomial commitment that dominates verification costs).

This may or may not be the case for IPs for circuit-satisfiability as well—IPs have larger proofs "outside of the polynomial commitment"

"bakes" circuit wiring information into the SRS generation procedure, and thereby does not have to "pay" in prover efficiency to achieve holography.

[10]The MPC-in-the-head approach, by virtue of yielding IOPs, also gives plausibly post-quantum secure protocols [7], [127].

than MIPs and constant-round IOPs, and that may or may not dominate verification costs, depending on the polynomial commitment scheme used, how small the witness is relative to the rest of the circuit, and how deep the circuit is.

Detailed asymptotic costs of the transparent polynomial commitments covered in this survey were provided in Table 16.1. Here is a brief summary of how concrete costs compare. Broadly speaking, in terms of prover costs, FRI[11] and Bulletproofs[12] are the most expensive polynomial commitment schemes, followed by those using pairings (Dory and KZG commitments). Hyrax, Ligero and Brakedown's commitments all have similar prover costs, though Brakedown is slightly faster and applies over fields that the others do not (see Section 19.3.1 for details). In terms of the sum of commitment size and evaluation proof length (which is what ultimately determines SNARK proof length), Brakedown is the largest, followed by Ligero, followed by Hyrax—all three yield roughly square-root size proofs, but with different constant factors. After that is FRI (polylogarithmic proof size). Next is Dory and Bulletproofs (logarithmic size proofs, with Bulletproofs shorter than Dory by a significant constant factor). KZG-commitments for univariate polynomials are the smallest (constant size).

Recent work called Orion [253] reduces the size of Brakedown's evaluation proofs via depth-one SNARK composition, but in so doing it relinquishes the field-agnostic nature of Brakedown and the proofs remain large (megabytes). Hyperplonk [95] proposes to reduce the proof size much further, to under 10 KBs, by combining Brakedown or

[11] As discussed in detail in Section 16.3, FRI exhibits a strong tension between prover costs and verification costs, with the tradeoff between the two determined by the *rate* of the Reed-Solomon code used in the protocol (see Section 10.4.4 for details). It can be configured to have a relatively fast prover, but then proofs are large (multiple hundreds of KBs), or a slower prover with smaller proofs (though still close to 100 KBs when run at 100 bits of security under current analyses [145]).

[12] A principle reason for high prover costs in Bulletproofs prover is that evaluation proofs are concretely expensive to compute: they require a linear number of exponentiations, as opposed to the single multi-exponentiation of linear size that suffices to compute the commitment itself. This high prover cost can be mitigated in contexts where many polynomials are committed and then evaluated, owing to efficient batch-verification of homomorphic polynomial commitments (Section 16.1).

Orion with KZG commitments, though this relinquishes transparency in addition to field-agnosticism.

Constant-round IOPs vs. MIPs and IPs. Broadly speaking, SNARKs from constant-round IOPs tend to be much slower and more space intensive for the prover than SNARKs from MIPs and IPs. This is because constant-round IOPs require the prover to commit to many polynomials (often 10 or more), while in SNARKs from MIPs or IPs, the prover only needs to commit to a single polynomial (which is no bigger than each of the polynomials arising in known constant-round IOPs).[13] This leads to prover time and space costs that are often one or more orders of magnitude larger than for MIP- and IP-derived SNARKs. The large number of committed polynomials in SNARKs from constant-round IOPs does not effect verification costs as much, owing to techniques for efficiently "batch-verifying" the claimed evaluations of multiple committed polynomials at the same input.

On pre-processing and work-saving for the verifier. The approaches requiring an SRS (i.e., linear PCPs, or any approach using KZG-based polynomial commitments) inherently require a pre-processing phase to generate the SRS, and this takes time proportional to the size of the circuit or R1CS instance and must be performed by a trusted party.[14] But the other approaches (combining any IP, MIP, or IOP with IOP-based or discrete-log-based polynomial commitments) can achieve a work-saving verifier *without pre-processing*, if applied to computations with a "regular" structure. By work-saving verifier, we mean that $\mathcal{V}$ runs faster than the time required simply to check a witness—in particular, $\mathcal{V}$'s runtime is sublinear in the size of the circuit or R1CS instance

[13] These statistics about the number of polynomials committed ignore the cost of holography/computation commitments, which requires even more (and even larger) committed polynomials (see Sections 10.3.2 and 16.2 and Footnote 9.). Even in the holographic setting, the qualitative comparison is similar.

[14] A partial exception is that combining IPs for circuit-satisfiability with KZG-based polynomial commitments has a setup phase of cost proportional to the size of the witness w rather than the entire circuit $\mathcal{C}$. Also, [84] gives a variant of KZG-based polynomial commitments with square-root-sized SRS, but logarithmic- rather than constant-sized evaluation proofs.

under consideration. For example, the MIP of Section 8.2 achieves a work-saving verifier without pre-processing so long as the multilinear extensions $\widetilde{\text{add}}$ and $\widetilde{\text{mult}}$ of the circuit's wiring predicate can be efficiently evaluated at any input, and any RAM of runtime T can be transformed into such a circuit of size $\tilde{O}(T)$ (Section 6).

That said, not all implementations of these approaches seek to avoid pre-processing for the verifier. One reason for this is that guaranteeing that the intermediate representation (whether a circuit, R1CS instance, or other representation) has a sufficiently regular structure to avoid pre-processing can introduce large concrete overheads to the representation size. Another is that "paying for" an expensive pre-processing phase can enable improved verification costs in the online phase of the protocol. For example, a primary ethos of SNARKs derived from linear PCPs, as well as constant-round polynomial IOPs combined with KZG commitments, is that, while it is expensive to generate the (long) SRS and distribute it to all parties wishing to act as the prover, checking proofs is extremely fast (only a constant number of group operations and bilinear map (pairing) evaluations). We elaborate further on these points at the end of this section.

To give a few examples from the research literature, STARKs [35], [128], [235] implement an IOP specifically designed to avoid pre-processing and achieve a polylogarithmic time verifier for *any* computation, with considerable effort devoted to mitigating the resulting overheads in the size of the intermediate representation. Although STARKs achieve considerable improvements over earlier instantiations of this approach [33], the resulting intermediate representations remain very large in general. Meanwhile, many IP and MIP implementations avoid or minimize pre-processing in data parallel settings [237], [242], [244] (see Section 4.6.7). These systems are able to exploit data parallel structure to ensure that the verifier can efficiently compute the information it needs about the computation in order to check the proof. Specifically, the time required for the verifier is independent of the number of parallel instances executed. They achieve this without incurring large concrete overheads in the size of the intermediate representation (see Section 6.6.4 for a sketch of how such overheads can arise when supporting work-saving verifiers for arbitrary computations).

Still other systems, such as Marlin [98], RedShift [165], PlonK [123], and Spartan [226] implement IOPs and MIPs targeted for the pre-processing setting, where a party can commit to polynomials encoding the wiring of the circuit or R1CS instance during pre-processing,[15] and thereafter, every time the circuit or R1CS is evaluated on a new input, the verifier can run in time sublinear in the circuit size. This is sometimes referred to as *holography* or *computation commitments*.

Finally, many systems do not seek a work-saving verifier even after potential pre-processing—these include [7], [43], [66], [80], [120].

Prover time in holographic vs. non-holographic SNARKs. In systems that implement holography, producing evaluation proofs for the "wiring" polynomials committed in pre-processing is typically the dominant cost in terms of prover time, for reasons discussed in Footnote 9. Hence, care should be taken when comparing prover time of holographic SNARKs to non-holographic ones (i.e., SNARKs that either don't seek to save work for the verifier, or that do seek to save work but only for circuits with "regular" wiring patterns). Non-holographic systems may achieve faster prover time as measured on a per-gate basis, but they may have to use much bigger circuits to achieve a work-saving verifier in general, or else limit themselves to applications that naturally perform the same "small' computation many times, to ensure "regular" wiring without significant blowups in circuit size.

19.3 Other Issues Affecting Concrete Efficiency

There are many subtle or complicated issues that can affect the concrete efficiency of a SNARK. This section provides an overview of some of them.

19.3.1 Field Choice

A subtle aspect of the various approaches to SNARK design that can have a significant effect on practical performance is the many ways in

[15] The pre-processing may or may not be transparent, depending on the polynomial commitment scheme used.

which the designer's choice of field to work over can be limited. One reason this matters is that many cryptographic applications naturally work over fields that do not satisfy the properties required by many SNARKs. Examples include proofs regarding encryption or signature schemes, many of which work over elliptic curve groups that are defined over fields that are not FFT-friendly; this is problematic for the many SNARKs in which the prover needs to perform FFTs.

Another reason flexibility in field choice matters is that for certain fields, addition and multiplication are particularly efficient on modern computers. For example, when working over Mersenne-prime fields ($\mathbb{F}_p$ where p is a prime of the form $2^k - 1$ for some positive integer k), reducing an integer modulo p can be implemented with simple bit-shifts and addition operations, and field multiplication can be implemented with a constant number of native (integer) multiplications and additions, followed by modular reduction. Mersenne primes include $2^{61}-1$, $2^{127}-1$, and $2^{521}-1$. Similarly fast arithmetic can be implemented more generally using any pseudo-Mersenne prime, which are of the form $2^k - c$ for small odd constant c (e.g., $2^{224} - 2^{96} + 1$). In contrast, modular reduction in an arbitrary prime-order field potentially requires division by p, and this is typically slower than reduction modulo pseudo-Mersenne primes by a factor of at least 2.[16] As another example of fields with fast arithmetic, some modern CPUs have built-in instructions for arithmetic operations in fields of sizes including 2^{64} and 2^{192}.

Limitations on the choice of field size for SNARKs come in multiple ways. Here are the main examples.

Guaranteeing soundness. All of the IPs, IOPs, MIPs, and linear PCPs that we have covered have soundness error that is at least $1/|\mathbb{F}|$ (and often larger by significant factors). Of course, so long as the soundness error is at most, say, $1/2$, the soundness error can always be driven to $2^{-\lambda}$ by repeating the protocol λ times, but this is expensive (often, only certain "soundness bottleneck" components need to be repeated, and this can mitigate the blowup in some costs, see for example Section 10.4.4).

[16] More information on efficient techniques for modular reduction in arbitrary prime-order fields can be found, for example, at https://en.wikipedia.org/wiki/Montgomery_modular_multiplication.

Regardless, $|\mathbb{F}|$ must be chosen sufficiently large to ensure the desired level of soundness.

Limitations coming from discrete-logarithm- or KZG-based polynomial commitments. SNARKs making use of discrete-logarithm-based or KZG-based polynomial commitments (Section 14), or linear PCPs (which are compiled into SNARKs via pairings) must use a field of size equal to the order of the cryptographic group that the polynomial commitment is defined over.[17] In contrast, SNARKs using polynomial commitment schemes derived from IOPs do not suffer such limitations, as the only cryptographic primitive they make use of is a collision-resistant hash function (to build a Merkle-tree over the evaluations of the polynomial to be committed), and such hash functions can be applied to arbitrary data.

Limitations coming from FFTs. SNARKs derived from IOPs (Section 10) and linear PCPs (Section 17) require the prover to perform FFTs over large vectors, and different finite fields support FFT algorithms of different complexities. In particular, standard FFT algorithms running in time $\tilde{O}(n)$ on vectors of length n work only for prime fields $\mathbb{F}_p$ if $p-1$ has many small prime factors.

Many, but not all, desirable fields do support fast FFT algorithms.[18] As an example, all fields of characteristic 2 do have efficient FFT algorithms, though until relatively recently, the fastest known algorithm ran in time $O(n \log n \log\log n)$. The extra $\log\log n$ factor was removed only in 2014 by Lin *et al.* [182]. Very recent work [37], [38] does show how to obtain $O(n \log n)$-time FFT-like algorithms and associated argument systems over arbitrary fields (after a more expensive field-dependent

[17] For any large enough prime p, it is typically possible to identify an elliptic curve group $\mathbb{G}$ with scalar field $\mathbb{F}_p$, such that discrete logarithms are presumed difficult to compute in $\mathbb{G}$. See for example [236]. However, the curve may not support pairings, and it is generally undesirable to have to design a new elliptic curve group every time a SNARK protocol designer wishes to change the field order p.

[18] By desirable, we either mean that the field supports fast arithmetic and meets the other desiderata described in this section, or that a particular cryptographic application calls for use of the field, say, because a SNARK is being used to prove a statement about an existing cryptosystem that performs arithmetic over the field.

pre-processing phase), but at the time of writing, its concrete efficiency remains unclear.

A related issue is that known constant-round polynomial IOPs [98], [123] require the field to have multiplicative or additive subgroups of specified sizes. For example, the polynomial IOP in Section 10.3 requires $\mathbb{F}$ to have a subgroup H of size roughly the number variables of the R1CS system, and a second subgroup $L_0 \supset H$ of size a constant factor larger than H. This is one reason SNARK designers may choose to work over a field of size 2^k, as this field has additive subgroups of size $2^{k'}$ for every $k' \leq k$ (see Remark 10.1 in Section 10.3.1).

Limitations coming from program-to-circuit transformations. IOP-derived SNARKs that seek to emulate arbitrary computer programs (Random Access Machines (RAMs)) while being work-saving for the verifier and avoiding pre-processing typically use transformations from RAMs to circuits or other intermediate representations that only work over fields of characteristic 2. We saw an example of this in Section 9.4.1, and modern instantiations such as STARKs [33], [35] also have this property.

Other considerations in field choice. There are other considerations when choosing a field to work over, beyond the limitations described above. For example, as discussed in Section 6.5.4.1, a prime field of size p naturally simulates integer addition and multiplication so long as one is guaranteed that the values arising in the computation always lie in the range $[-p/2, p/2]$ (if the values grow outside of this range, then the field, by reducing all values modulo p, will no longer simulate integer arithmetic). Such an efficient simulation is not possible in fields of characteristic 2. Conversely, addition in fields of characteristic 2 is equivalent to bitwise-XOR. Hence, aspects of the computation being fed through the proof machinery will affect which choice of field is most desirable: arithmetic-heavy computations may be more efficiently simulated when working over prime fields, and computations heavy on bitwise operations may be better suited to fields of characteristic 2.

Example field choices. To give some examples from the literature: Aurora [43], which is based on IOPs, chooses to work over the field of size 2^{192}. This is large enough to provide good soundness error while supporting FFT algorithms requiring $O(n \log n)$ group operations, and some modern processors have built-in support for arithmetic over this field. Virgo [256] chooses to work over the field of size p^2 where $p = 2^{61}-1$ is a Mersenne prime, to take advantage of the fast field operations offered by such primes. [35] chooses to work of the field of size 2^{64}. This field is not large enough to ensure cryptographically-appropriate soundness error on its own, so aspects of the protocol are repeated several times to drive the soundness error lower. PlonKy2 works over (an extension of) the field of size $2^{64} - 2^{32} + 1$. This field has is increasingly popular due features such as its fast arithmetic, its ability to support FFTs of length up to 2^{32} or larger, and the fact that it is large enough to represent the product of two 32-bit unsigned integer data types (which is at most $(2^{32} - 1)^2 = 2^{64} - 2^{33} + 1$).

The three systems above use FRI-based polynomial commitments, meaning they do not have to work over a field of size equal to the order of some cryptographically-secure group (though they do need the field to support FFTs and have subgroups of specified sizes for other reasons as well). SNARKs based on pairings or discrete-logarithm-based polynomial commitments are not able to work over these fields.

Hyrax [244] and Spartan [226], both of which combine IPs or MIPs with discrete-logarithm-based polynomial commitments, work over the field whose size is equal to the order of (a subgroup of) the elliptic curve group Curve25519 [52] (see Section 12.1.2.2), with this group chosen for its fast group arithmetic and popularity.

Systems that use pairings (e.g., all linear-PCP-derived SNARKs, as well as any SNARK using KZG-based polynomial commitments or Dory) work over a field of size equal to the order of (a subgroup of) chosen pairing-friendly elliptic curves. There have been significant efforts to design such pairing-friendly curves with fast group arithmetic while ensuring, e.g., that the order of the chosen subgroup is a prime p such that the field $\mathbb{F}_p$ supports fast FFTs. The most popular such curve today is perhaps BLS12-381—see the references in Footnote 3 in Section 15.1.

The choice of field can make a significant concrete difference in the efficiency of field arithmetic. For example, experiments in [226], [256] suggests that the field used in Virgo (of size $(2^{61}-1)^2 \approx 2^{122}$) has arithmetic that is at least $4\times$ faster than the field used in Hyrax and Spartan (of size close to 2^{252}). Much of this $4\times$ difference can be attributed to the fact that Virgo's field is roughly square root of the size of Hyrax and Spartan's, and hence field multiplications operate over smaller data types. However, some of the difference can be attributed to extra structure in the Mersenne prime $2^{61}-1$ that is not present in the prime order field used by Hyrax and Spartan.

Currently, Brakedown [137], which combines an MIP with the Brakedown-commitment, is the only implemented SNARK that neither requires the field to support FFTs nor to match the order of a cryptographic group. The same would hold if combining an IP with the Brakedown commitment, but not a constant-round polynomial IOP.[19]

19.3.2 Relative Efficiency of Different Operations

Of course, the speed of field arithmetic is just one factor in determining overall runtime of a SNARK. In some SNARKs, the bottleneck for the prover is performing FFTs over the field, in others the bottleneck is group operations, and in still others the bottleneck may be processes that have nothing to do with the field choice (e.g., building a Merkle tree). To give one example, in SNARKs for R1CS-satisfiability derived from constant-round polynomial IOPs, the prover typically has to perform an FFT over a vector of length $\Theta(K)$, where K is the number of nonzero matrix entries of the R1CS system, and also must build one or more Merkle trees over vectors of length $\Theta(K)$. For reasonably large values of K, the $O(K \log K)$ runtime of the FFT will be larger than the time required to perform the $\Theta(K)$ evaluations of a cryptographic hash function that are needed to build the Merkle tree(s). But for very small values of K, the $\log K$ factor in the FFT runtime may be concretely smaller

[19] This is because Brakedown-commitment applies to univariate polynomials represented over the standard monomial basis, but the univariate polynomials arising in constant-round polynomial IOPs are specified via their evaluations over a subgroup H of $\mathbb{F}$. Efficiently converting from these evaluations to the standard monomial basis requires an FFT.

than the time required to evaluate a cryptographic hash evaluation, particularly if the field supports fast arithmetic, ensuring the hidden constant in the FFT runtime is small. So which part of the protocol is the bottleneck (FFT vs. Merkle-tree building) likely depends on how large a computation is being processed.

As another example, if an MIP is combined with many of the discrete-logarithm-based or KZG-based polynomial commitment schemes, the prover does not have to do any FFTs, and the bottleneck is typically in performing one multi-exponentation of size proportional to K. Via Pippenger's algorithm, the multiexponentiation can be done using $O(K \log(|\mathbb{G}|)/\log(K))$ group multiplications. In many other SNARKs the prover would have to at least perform an FFT over a vector of length at least K, and this will cost $O(K \log K)$ field operations.

For small R1CS instances, the FFT is likely to be faster than the multi-exponentation, for three reasons. First, each operation in a cryptographic group $\mathbb{G}$ is often an order of magnitude more expensive than a field multiplication. Second, when K is small, $\log(|\mathbb{G}|)/\log(K) \gg \log K$, so even ignoring differences in the relative cost of a group vs. field operation, $O(K \log(|\mathbb{G}|)/\log K)$ is larger than $O(K \log K)$. Third, if the SNARK uses IOP-based polynomial commitments, it has the flexibility to work over a field whose size is not the order of an elliptic-curve group, and these fields can potentially support faster arithmetic. However, once K is large enough that $\log|\mathbb{G}| \ll \log^2 K$, the $O(K \log K)$ field operations required by the FFT will take more time than the $O(K \log(|\mathbb{G}|)/\log(K))$ group multiplications required to perform the multiexponentiation.

19.3.3 Intermediate Representations (IRs) Other than Arithmetic Circuits and R1CS

This survey described a variety of SNARKs for arithmetic circuit-satisfiability and R1CS. The SNARKs for arithmetic circuits supported addition and multiplication gates of fan-in two. R1CS systems are conceptually similar to arithmetic circuits augmented to allow "linear combination" gates of arbitrary fan-in (see Section 8.4.1). In both cases, all gates compute degree-two operations, meaning the output of each

gate or constraint is a polynomial in the gate's inputs of total degree at most 2.

However, SNARKs based on polynomial IOPs can typically be modified to support more general intermediate representations. For example, they can typically be modified to handle circuits with gates computing operations of total degree up to some bound d, with an increase in prover time that grows linearly with d.[20]

The proof size will typically also grow with the degree bound d, but in many cases this growth will be a low-order effect. For example, in MIP-based SNARKs using certain polynomial commitment schemes, the proof length, when applied to a circuit of size S in which gates compute operations of degree at most d, will be $O(S^{1/2} + d \log S)$. The $d \log S$ term, which grows linearly with d, will be dominated by the $S^{1/2}$ term unless the circuit size S is tiny.

The use of such expanded gate sets can be fruitful. To give a simple example, suppose that allowing gates computing degree-3 operations rather than degree-2 operations reduces the size S of the resulting circuit by a factor of 2, while the prover's runtime as a function of S increases by a factor of only 4/3 due to the degree of gate operations increasing from 2 to 3. Then using the expanded gate set yields a faster prover: the total prover time decreases by a factor of $2 \cdot (3/4) > 1$.[21]

[20] Such modifications require understanding how the SNARK works and modifying it in a non-black-box manner. To give a very rough indication of how this might work: the MIP of Section 8.2 uses a $(3 \log S)$-variate polynomial $g(a, b, c) = \widetilde{\text{mult}}(a, b, c) \left(\widetilde{W}(a) - \widetilde{W}(b) \cdot \widetilde{W}(c) \right)$ to "check" that the value that $\widetilde{W}$ "assigns" to a multiplication gate a is the product of the values assigned to the two in-neighbors of a. To support multiplication gates of fan-in three rather than two, with the MIP one would replace g with the following modified polynomial defined over $4 \log S$ variables, with total degree four instead of three: $g(a, b, c, e) = \widetilde{\text{mult}}(a, b, c, e) \left(\widetilde{W}(a) - \widetilde{W}(b) \cdot \widetilde{W}(c) \cdot W(e) \right)$.

[21] Intermediate representations that are *more restrictive* than arithmetic circuits or R1CS can also be useful for some applications. This will be the case if one manages to design a SNARK with improved prover time or proof size for these limited IRs, and the improvements outweigh the negative effects of any resulting increase in representation size. For one example, see the notion of "R1CS-Lite" in [86]. Highly efficient SNARKs for incremental computation (Sections 18.4 and 18.5) can also be thought of as utilizing restricted IRs to obtain efficiency benefits, as can the various super-efficient IPs given for specific problems in Section 4.

The polynomial-IOP-derived SNARKs covered in this survey can all be modified in the manner sketched above to support operations of total degree higher than 2. This does *not* appear to be the case for linear-PCP derived SNARKs such as Groth's [142]: their use of pairing-based cryptography appears to rely heavily on the linear PCP verifier computing a degree-2 function of the proof string, which in turn relies on the circuit or R1CS instance computing only degree-2 operations. It is also not clear that recursive-composition-based SNARKs for iterative computation, e.g., Nova (Section 18.5) can be modified support operations with degree $d > 2$, at least not without a blowup in prover time or proof size that outweighs the benefits.

There has been particular recent interest in modifications of the PlonK SNARK to support expanded and modified IRs. For example, the cryptocurrency Zcash incorporates so-called "PlonKish arithmetization" into its Orchard protocol.[22] This refers to a modification of PlonK[23] to support an IR reminiscent of circuits with gates of degree up to 9.[24] There is also considerable interest in backends for a related IR called AIR [128], [235].

As SNARK protocol designers move beyond arithmetic circuits and R1CS to variant IRs, the line between "front-ends" (Section 6) and "back-ends" (i.e., SNARK proof machinery) becomes blurred. Protocol designers may tailor the chosen IR to the desired back-end, and in turn have to modify the chosen back-end to support the resulting IR.

There may be tradeoffs to such efforts. On the one hand, the use of more expressive or idiosyncratic IRs may yield important efficiency gains. On the other, it may increase the burden on protocol designers or render it more difficult to develop or reuse infrastructure. For example, protocol designers may find themselves painstakingly designing "circuits" in the modified IRs by hand to adequately take advantage of the expanded set of primitive operations supported. And if one decides to swap out

[22] https://zips.z.cash/zip-0224.

[23] More precisely, to a modification of the SNARK obtained by combining the constant-round polynomial IOP underlying PlonK with the Bulletproofs polynomial commitment scheme.

[24] See, for example, https://zcash.github.io/halo2/concepts/arithmetization.html and, in particular, sections on "gadgets" within the same document, such as https://zcash.github.io/halo2/design/gadgets/sha256/table16.html.

one back-end for another with a different cost profile, one may have to change the IR, and hence repeat the entire protocol design process, or at least alter the front-end.