The Small-Field Turn in Succinct Proofs:
A Systematization of Finite-Field Choice in Modern SNARKs and STARKs

Christos Skatharoudis
PhD Candidate, Department of Information and Communication Systems Engineering
University of the Aegean, Karlovasi, Samos, Greece
ORCID: 0009-0001-6398-556X

Abstract

Over the past half-decade, transparent succinct arguments have migrated off the 256-bit scalar fields of pairing-friendly elliptic curves and onto small fields: the 64-bit Goldilocks prime, the 31-bit primes BabyBear and KoalaBear, the Mersenne prime $2^{31} - 1$ reached through the circle construction, and binary tower fields down to $\mathbb{F}_2$. We call this movement the *small-field turn* and systematize it with the finite field, rather than the proof system or the virtual machine, as the unit of analysis. We organize the fields in use by the structural properties that drive their selection: machine-word fit, two-adicity, reduction cost, and the S-box automorphism structure exploited by algebraic hashes; and we show how each is realized in a production prover (Plonky2, Plonky3, Stwo, Binius, and their descendants). We then assemble, across systems that state it only individually, the relationship between base-field width and the extension degree that Fiat–Shamir soundness requires, and we set that relationship against the measured gap between conjectured and provable soundness for non-interactive FRI. Finally, we separate the peer-reviewed results on embedding and arithmetization overhead from the vendor benchmarks that dominate the topic, and identify the controlled cross-field comparison whose absence is the area's sharpest empirical gap. No prior work takes field choice as its organizing object across this design space; the nearest systematization treats it as one dimension among many within a zero-knowledge virtual machine taxonomy. We frame the turn as the exploration of a single trade, cheaper arithmetic against repurchased soundness and simulated non-native operations, and argue from the provenance of the fields where the frontier is likely to move next.

Keywords: zero-knowledge proofs, SNARKs, STARKs, finite fields, FRI, polynomial commitment schemes, arithmetization.

1. Introduction

A succinct non-interactive argument lets a verifier check a long computation by reading a short proof. For most of the last decade the arithmetic underneath these arguments lived in a single kind of place: the scalar field of a pairing-friendly elliptic curve, roughly 256 bits wide, as in BN254 or BLS12-381. That choice was not arbitrary. Pairing-based commitments such as KZG need it, and a 256-bit field is comfortably large enough that a single random challenge drawn from it makes a cheating prover's success probability negligible. The field was, in effect, free security.

It was also a poor fit for the machine doing the work. A CPU adds and multiplies 32- and 64-bit words; a GPU lane is happiest on the same. Representing one 256-bit field element consumes four or more machine words, and every field multiplication becomes a small schoolbook routine with carry propagation. When the value being encoded is a single bit or a byte of a hash computation, most of those 256 bits carry no information at all. The proof system pays, in memory bandwidth and in multiplier width, for headroom it never uses. A recent accounting of one deployed prover makes the waste concrete: committing to the trace of a single RISC-V

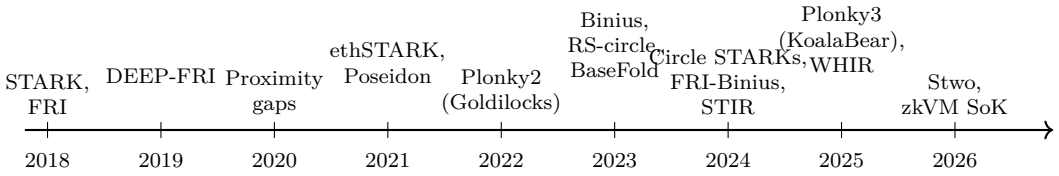

Figure 1: A timeline of the small-field turn (Section 3). Foundational low-degree-testing results (STARK/FRI, DEEP-FRI, proximity gaps) precede the field constructions and the provers that adopted them.

step costs on the order of 275 31-bit field elements, which is roughly equivalent to 34 full 256-bit elements per step once the smaller field is accounted for [1]. Read the other way, the large field would have cost several times more for the same work.

The response, worked out across a cluster of proof systems over roughly the past five years, was to shrink the field. Transparent, hash-based arguments in the STARK family do not need a pairing, and their low-degree test (the Fast Reed–Solomon Interactive Oracle Proof of Proximity, or FRI) places no large-characteristic requirement on the field [2, 3]. That freedom opened a design space. Plonky2 encoded its witnesses in the 64-bit Goldilocks field, chosen so that elements fit a machine word and reduction needs no multiplication [4]. A subsequent generation pushed to 31-bit primes: BabyBear and KoalaBear in the Plonky3 toolkit [5], and the Mersenne prime $2^{31} - 1$, whose arithmetic is about as cheap as prime-field arithmetic gets, reached through the circle-group construction of Circle STARKs [6, 7]. A more radical branch abandoned prime fields altogether: Binius builds its arguments over towers of binary fields, down to the two-element field itself, so that a bit is stored as a bit [8, 9].

We call this collective movement the *small-field turn*, and this paper systematizes it.

1.1 The recurring trade-off

Shrinking the field relocates cost rather than removing it. Three tensions recur across every system we survey, and they organize the paper.

The first is soundness. A challenge drawn from a 31-bit field can be guessed by a cheating prover with probability around 2^{-31}, which is nowhere near a cryptographic target. Every small-field system therefore draws its Fiat–Shamir randomness not from the base field but from an extension of it, large enough to restore a negligible soundness error. ethSTARK states the principle plainly: to be secure "one must use a larger field in some places" [10]. The DEEP method and the proximity-gap machinery make this precise, and, as we show in Section 4, the required extension degree turns out to depend on the base field, not on a single universal rule.

The second is the security foundation itself. The soundness of non-interactive FRI at deployed parameters is conjectural: it assumes that the best known attack is the best possible attack. A recent analysis quantifies the exposure, showing that provable security for most deployed parameter sets sits 21 to 63 bits below the conjectured level [11]. A survey of field choice that quoted "128-bit security" without this qualifier would be misleading.

The third is embedding. A smaller field narrows, but does not close, the gap between the field element and the datum being encoded. Thirty-two-bit computation still does not sit cleanly inside a 31-bit prime; range checks and non-native arithmetic reappear as arithmetization overhead. Only the binary-tower branch closes the gap entirely, and it does so by construction: Binius commits to small-field polynomials "with no embedding overhead," a property none of the prime-field systems can claim [8].

1.2 Contributions

This paper makes four contributions.

- **A taxonomy of the field families** now in production use (64-bit Goldilocks, the 31-bit primes BabyBear and KoalaBear, the Mersenne prime $2^{31} - 1$, and binary tower fields), organized by the structural properties that drive each choice: word fit, two-adicity, reduction cost, and the automorphism structure exploited by algebraic hashes (Section 3).
- **A cross-system tabulation of extension degree against security target** (Table 2), together with the deployed FRI parameters that realize it (Table 3). The underlying relationship, that achievable security is capped at $\log_2 |K|$ so a wider base field needs a smaller extension degree, is a one-line consequence of the Fiat–Shamir bound (Section 2.5); the contribution is the assembly of what deployments actually choose, the observation that they meet the cap at different degrees (two for the 61- and 64-bit fields, four for the 31-bit fields), and the placement of the conjectured-versus-provable soundness gap alongside these numbers (Section 4). The unified law that would predict the minimal degree from base-field width, rate, and query budget across both prime and binary families remains open (Section 7.1).
- **A synthesis of the embedding and arithmetization overhead** arguments, separating the peer-reviewed claims from the vendor benchmarks that dominate this part of the literature, and stating where no rigorous cross-field comparison yet exists (Section 5).
- **A prover-to-field map** that ties each production system to its field and the rationale its designers gave, correcting a common misattribution along the way (Section 6).

1.3 Relation to prior systematization

The closest prior work is the recent systematization of zero-knowledge virtual machines by Yang et al. [12]. That work takes the *zkVM* as its unit of analysis, decomposing systems into instruction-set, virtual-machine, and proving layers. It is not silent on fields: it includes a comparison table of the common finite fields in zkVM provers and a subsection on small-field soundness concerns, the latter citing Fenzi and Sanso's result that small-field hash-based SNARGs are less sound than conjectured [13]. But these enter as attributes of the proving backend inside a zkVM taxonomy, not as the organizing object, and the one-line summary of the trade-off (that small fields such as BabyBear and Goldilocks require degree-four extensions for 128-bit security at a reported 9–16× per-operation overhead, against the large pairing-friendly field BN254) is imprecise in a way Section 4.4 corrects: a 64-bit field such as Goldilocks reaches a 2^{128} challenge space with a degree-*two* extension, and the degree-four requirement is specific to the 31-bit fields. Our survey instead takes the *field choice itself* as the organizing object, and goes deeper on both axes that work touches: we give a structural taxonomy of field families rather than a feature table, treat the extension-degree-versus-security relationship and its conjectured-versus-provable gap in detail, and add the arithmetization-overhead axis. We treat Circle STARKs over the Mersenne prime and binary-tower constructions as *field-choice objects* rather than interchangeable proving backends, and we extend beyond virtual machines to raw STARK and polynomial-commitment constructions, which a zkVM-centered systematization does not cover. The two general zero-knowledge-SNARK systematizations we are aware of are further still from this topic: one surveys more than forty schemes and eleven libraries without treating small fields at all [14], and the other is an empirical taxonomy of implementation vulnerabilities rather than of design [15]. To our knowledge, no prior work takes finite-field choice as its unit of analysis across the small-field turn; the nearest, a zkVM systematization, treats it as one dimension among many.

1.4 A note on how these fields arrived

One structural observation runs through the survey. The field families divide by provenance. Mersenne-31 and the binary towers each arrived with a construction paper, a piece of mathematics whose purpose was to make that field usable for proving [6–8]. Goldilocks and BabyBear did

not. They entered the field through prover engineering: Goldilocks is introduced in a section of the Plonky2 technical report justifying it on reduction cost and word fit [4], and BabyBear is defined by its implementation in a toolkit [5]. The distinction is not only historical. It marks two different ways a design community converges on a primitive: by proving a theorem that unlocks a field, or by measuring that a field is fast and standardizing it. We return to it in Section 7 as a lens on where the next field is likely to come from.

1.5 Roadmap and scope

Section 2 fixes the background: interactive oracle proofs, FRI, polynomial commitments, arithmetization, and the soundness primitives on which everything rests. Section 3 presents the taxonomy of field families. Section 4 treats extension-field soundness and assembles the extension-degree-versus-security account. Section 5 synthesizes the overhead arguments. Section 6 maps provers to fields. Section 7 states the open problems, and Section 8 concludes.

Two scope limits apply throughout. We survey field *choice*, not proof systems in full; a system appears only insofar as its field is instructive. And we run no experiments beyond one bounded illustrative micro-measurement (Appendix A): every other quantitative figure is reported from its source and labeled by the strength of that source, with vendor and blog measurements marked as such and distinguished from peer-reviewed results.

---

2. Background

This section fixes notation and recalls the machinery on which the small-field turn depends. A reader fluent in FRI-based proof systems can skim it; we include it so that the soundness arguments of Section 4 and the overhead arguments of Section 5 can be stated precisely. Throughout, $\mathbb{F}_q$ denotes a finite field with q elements, and for a prime p we write $\mathbb{F}_p$ for the prime field and $\mathbb{F}_{p^e}$ for its degree-e extension.

2.1 Interactive oracle proofs and the STARK paradigm

The proof systems we survey are instances of the *interactive oracle proof* (IOP) model, in which a prover sends oracles (long strings the verifier may query at a few positions) rather than messages read in full. A STARK compiles an IOP into a non-interactive argument by two standard steps: the oracles are replaced by Merkle commitments, so that a query becomes an authenticated opening, and the interaction is removed by the Fiat–Shamir transform, so that each verifier challenge is derived by hashing the transcript so far [3]. The result is transparent (it needs no trusted setup), plausibly post-quantum (its only cryptographic assumption is the collision resistance of a hash function), and (the property that matters here) indifferent to the algebraic structure of the field beyond what the low-degree test requires [3]. It is this indifference that makes field choice a free variable to be optimized.

An arithmetized computation is expressed as a set of low-degree polynomial constraints over an execution trace. Proving reduces to two tasks: convincing the verifier that committed column polynomials satisfy the constraints, and convincing it that those polynomials have low degree. The first is handled by a constraint-aggregation step; the second is the province of FRI.

2.2 FRI and low-degree testing

The Fast Reed–Solomon Interactive Oracle Proof of Proximity (FRI) is a protocol by which a verifier, given oracle access to a function $f : D \to \mathbb{F}_q$ on an evaluation domain D, becomes convinced that f is close to the Reed–Solomon code $\mathrm{RS}[\mathbb{F}_q, D, \rho]$ of rate ρ, that is, close to the

evaluations of some polynomial of degree below $\rho|D|$ [2]. FRI proceeds in $\log|D|$ rounds, each folding the function to half its length using a verifier challenge, and terminates when the folded function is short enough to send in the clear. Its prover cost is linear in $|D|$ and its verifier cost and proof length are logarithmic [2].

Two features of FRI drive the entire survey. First, FRI places no large-characteristic requirement on $\mathbb{F}_q$: the folding map and the low-degree test work over any field with a suitable evaluation domain, which for the classical protocol means a multiplicative subgroup of smooth (highly two-adic) order [2,3]. This is precisely the freedom the small-field turn exploits. Second, FRI's soundness, the guarantee that a function *far* from the code is rejected, is governed by the *distance* to the code and by the *size of the field from which challenges are drawn*, not by the base field in which f takes its values. Sections 2.3–2.5 make this split explicit, because it is the technical heart of why a small base field can be paired with a large challenge field.

A single folding round makes the mechanism concrete. Take the evaluation domain D to be a coset of a multiplicative subgroup of order 2^k, and write the tested function's underlying polynomial by separating even and odd powers, $f(X) = f_0(X^2) + X\, f_1(X^2)$, so that f_0 and f_1 each have half the degree of f [2]. The verifier sends a challenge α, drawn from the large field of Section 2.5, and the prover folds to $f'(Y) = f_0(Y) + \alpha\, f_1(Y)$, a polynomial of half the degree defined on the squared domain $D^2 = \{x^2 : x \in D\}$, which has half the size [2]. Iterating this fold $\log_2|D|$ times drives the degree to a constant, at which point the prover sends the final codeword in the clear [2]. The verifier's job in the query phase is to test that each layer is the correct fold of the previous one: it samples a point, opens f at the two square roots that map to it, and checks that their α-combination matches the opened value of f', a local consistency check repeated across enough independent queries that a function far from the code survives with only small probability [2].

Two soundness-relevant quantities fall out of this picture. The number of queries ℓ controls the query-phase soundness error, which for a code of relative distance δ behaves like $(1-\delta)^\ell$ up to the proximity-gap correction of Section 4.3 [2, 16]; more queries mean a longer proof but a smaller error. The challenge α, and the out-of-domain point of Section 4.2, must come from a field large enough that a cheating prover cannot anticipate them. In the non-interactive setting these challenges are produced by the Fiat–Shamir transform: each is the hash of the transcript up to that point, so that the prover is bound to its earlier commitments before it learns the challenge, and the oracles themselves are Merkle-committed so that an opening is an authenticated leaf [3]. The security of this compilation rests only on the collision resistance of the hash, which is what makes the resulting argument transparent and plausibly post-quantum [3].

2.3 Polynomial commitment schemes

A polynomial commitment scheme (PCS) lets a prover commit to a polynomial and later open its evaluation at a queried point, succinctly and bindingly. The small-field systems divide by the PCS they use. FRI-based systems (ethSTARK, Plonky2/3, Circle STARKs) commit by Merkle-hashing a Reed–Solomon encoding and prove evaluations through the FRI low-degree test [3,4,7]. Code-based systems in the Brakedown lineage commit through a linear code with a tensor query structure and are field-agnostic by design [17]; Binius's original construction adapts Brakedown, giving proofs of size $O(\sqrt{n})$ for an n-coefficient polynomial [8,17]. The distinction matters for field choice because the two families impose different demands: FRI wants a smooth evaluation domain, which constrains the multiplicative structure of the field, whereas the Brakedown-style scheme is agnostic to that structure and is what first made the two-element field usable as a commitment field [8,17]. FRI-Binius later reconciled the two, adapting the field-agnostic foldable-code commitment of BaseFold [18] to binary towers and reinterpreting the additive-FFT polynomial basis of Lin, Chung, and Han [19] so that a FRI-style fold applies over $\mathbb{F}_2$, reducing Binius proofs to polylogarithmic size [9].

The commitment is only half of a proof system; the other half is the constraint system it

commits to, and here too the small-field designs reuse a common toolkit. Binius arithmetizes over binary fields by adapting the product- and permutation-check machinery of HyperPlonk [20] and the lookup argument of Lasso [21] to characteristic two, which is what lets it express bit-oriented computations such as Keccak natively [8]. The lookup-argument line is also where much of the range-check cost of Section 5.2 is paid, and it is shared with the prime-field zkVMs [1, 21]. Two axes of PCS design recur below: whether the committed polynomial is *univariate* (STARK/FRI tradition) or *multilinear* (the Binius and lookup-argument tradition [1,8]), and whether soundness rests on unique decoding or on the more aggressive list-decoding regime (Section 4.3).

2.4 Arithmetization and why the field is load-bearing

Arithmetization is the encoding of a computation as polynomial constraints over $\mathbb{F}_q$. In the AIR (Algebraic Intermediate Representation) and PLONKish styles used by the systems we survey, the trace is a table of $\mathbb{F}_q$ elements and the constraints are low-degree relations among adjacent rows [3, 4, 22]. The field enters the cost model in three places. Native field operations are the cheap primitive, so a computation whose data type matches the field is cheap to arithmetize. Operations that do *not* match (a 32-bit integer addition in a 31-bit field, a bitwise XOR in a prime field, a comparison) must be decomposed into field-native constraints, typically through range checks and lookup arguments whose cost is charged back to the trace. And the field's algebraic structure determines the cost of the hash used inside the prover: an S-box of the form $x \mapsto x^3$ is a permutation only when $3 \nmid |\mathbb{F}_q^\times|$, a constraint that, as we see in Section 3.2, shaped the design of at least one field [5]. Field choice therefore fixes what is cheap and what is expensive before a single constraint is written, upstream of any downstream tuning knob.

2.5 Soundness primitives and the role of field size

The Fiat–Shamir transform makes soundness quantitative. Each challenge is a random field element, and a cheating prover who must guess a challenge succeeds with probability at most $1/|K|$, where K is the field the challenge is drawn from. When many challenges and many FRI rounds compose, the soundness error is a sum of such terms together with a list-decoding contribution (Section 4.3), and the achievable security level is capped by $\log_2 |K|$ [10]. This single inequality, security $\leq \log_2 |K| - O(1)$, is the reason a small base field cannot supply its own challenges, and it is the quantitative seed of the extension-degree analysis in Section 4.4. It also explains a subtlety that Section 4.5 develops: the FRI soundness bounds used at deployment are conjectural, so $\log_2 |K|$ is a ceiling on *conjectured* security, and provable security can sit well below it [11].

It is worth writing the composition out, because it shows where each field enters. A deployed FRI-based argument accumulates soundness error from three sources: a batching term, of order $O(n/|K|)$, from combining many polynomials into one before the low-degree test (Section 4.3); a low-degree-test term, of order $(1-\delta)^\ell$, from the ℓ query repetitions at relative distance δ (Section 2.2); and a family of challenge-guessing terms, each $1/|K|$, from the out-of-domain point and the folding challenges (Section 4.2) [2, 10, 16]. Writing ε for the total, the security level is $\lambda = -\log_2 \varepsilon$, and every term that carries $|K|$ in a denominator ties the achievable λ to the size of the challenge field. Deployments buy back the shortfall in two ways that recur in later sections: they raise the query count ℓ, trading proof length for a smaller $(1-\delta)^\ell$, and they add proof-of-work "grinding," which forces the prover to solve a small hash puzzle per proof and adds its difficulty directly to λ [10]. Grinding is the reason a deployment can target a security level tens of bits above what its challenge-derived terms alone supply (Section 4.6).

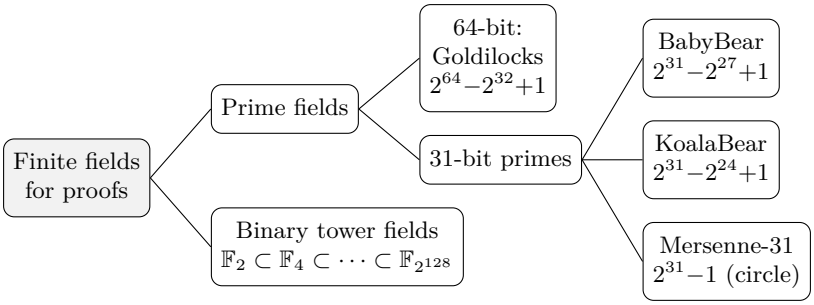

Figure 2: A taxonomy of the finite fields in production use for succinct proofs (Section 3). Prime fields divide by machine-word fit and two-adicity; the binary tower branch abandons prime fields to store a bit as a bit.

2.6 The sumcheck route and its indifference to field structure

A second family of arguments runs alongside the FRI/Reed–Solomon line and is central to the binary-field branch, so we recall it here. The sumcheck protocol reduces a claim about the sum of a multivariate polynomial over the Boolean hypercube $\{0,1\}^\nu$ to a claim about its evaluation at a single random point, through ν rounds in which the prover sends a univariate polynomial and the verifier responds with a random challenge [23]. Its relevance to field choice is that it needs *no* smooth evaluation domain: the challenges are arbitrary field elements, the prover's messages are low-degree univariate polynomials, and nothing in the protocol requires a multiplicative subgroup of highly two-adic order [20]. Where FRI is constrained by the field's multiplicative structure, sumcheck is indifferent to it.

This indifference is why the sumcheck route dominates the systems that operate over the most structure-poor fields. HyperPlonk ports PLONK [22] from a univariate, roots-of-unity setting to the hypercube, replacing quotient-polynomial checks with sumcheck-based product and permutation arguments, and is FFT-free as a result [20]. Lasso builds its lookup argument in the same idiom, proving that every entry of a committed vector lies in a table while committing to only about $m+n$ field elements for m lookups into a size-n table, and exploiting decomposable table structure so that huge tables are never materialized [21]. BaseFold generalizes the FRI folding itself, running a sumcheck in lockstep with a fold over any "foldable" linear code, not only Reed–Solomon, to obtain a field-agnostic multilinear commitment [18]. Binius stands on all three: it commits over binary towers, checks its constraints with HyperPlonk-style sumchecks, discharges its lookups with a characteristic-two Lasso, and, in the FRI-Binius refinement, commits through a binary-tower adaptation of BaseFold [8,9,18]. The multilinear/sumcheck route and the univariate/FRI route are thus two responses to the same fact: the less multiplicative structure a field has, the more a proof system must lean on sumcheck instead of the FFT.

---

3. A taxonomy of field families

The small-field turn did not converge on one field. It produced a small menu, and the entries differ along axes that this section makes explicit: bit-width relative to the machine word, two-adicity (the largest power of two dividing the order of the group used for FFTs), reduction cost, and the automorphism structure relevant to algebraic hashing. Table 1 collects the parameters; the subsections give the rationale.

Table 1. Field families in production use. *(Primes and two-adicities verified against primary sources where marked; see Section 3.5 and the hygiene note.)*

Family	Field	Prime p	bits	2-adic.	Reduction / structure	Src.
64-bit prime	Goldilocks	$2^{64} - 2^{32} + 1$	64	32	constant-free; $2^{64} \equiv 2^{32} - 1$	[4]
31-bit prime	BabyBear	$2^{31} - 2^{27} + 1$	31	27	max two-adicity at 31 bits	[5]
31-bit prime	KoalaBear	$2^{31} - 2^{24} + 1$	31	24	$x \mapsto x^3$ automorphism of $\mathbb{F}_p^\times$	[5]
Mersenne prime	Mersenne-31	$2^{31} - 1$	31	1	cheapest reduction; needs circle group (p+1=2^{31})	[6, 7]
Binary tower	$\mathbb{F}_{2^{2^k}}$	char. 2	1–128	n/a	XOR addition; tower inversion	[8, 24, 25]

3.1 The 64-bit prime: Goldilocks

Goldilocks, $p = 2^{64} - 2^{32} + 1$, is the field of Plonky2, and the Plonky2 technical report justifies it on two grounds that recur throughout the survey [4]. First, elements fit in a single 64-bit machine word, so a field element is a register rather than an array. Second, the prime's shape yields a reduction with no multiplications. The two congruences $2^{64} \equiv 2^{32} - 1 \pmod p$ and $2^{96} \equiv -1 \pmod p$ follow directly from $p = 2^{64} - 2^{32} + 1$, and they let a 128-bit product be folded back into range by arithmetic on its 32-bit limbs alone: writing the product as $c_0 + c_1 2^{32} + c_2 2^{64} + c_3 2^{96}$, the high limbs are reabsorbed using $2^{64} \equiv 2^{32} - 1$ and $2^{96} \equiv -1$, leaving a value within a couple of additions of the canonical residue [4]. The final conditional subtraction of p is itself cheap, because subtracting p is the same as adding $2^{32} - 1$ in 64-bit arithmetic, and setting a register to 0 or $2^{32} - 1$ on the carry flag is a single instruction on x86 (SBB) and ARM64 (CSETM) [4]. Goldilocks also carries a two-adicity of 32 (the multiplicative group order $p - 1 = 2^{32}(2^{32} - 1)$ is divisible by 2^{32}), which is ample for the FFTs that FRI requires [4]. It is the largest field that still fits a word, chosen so that the arithmetic is nearly free; the Miden virtual machine uses the same field for the same reasons [26].

3.2 The 31-bit primes: BabyBear and KoalaBear

The next generation halved the width again, to 31-bit primes that fit comfortably inside a 32-bit lane and expose more of a SIMD register's throughput. Two such primes are in use, and the difference between them is instructive because it is a difference of *algebraic* rather than arithmetic convenience.

BabyBear, $p = 2^{31} - 2^{27} + 1 = 2{,}013{,}265{,}921$, is chosen as the 31-bit prime of highest two-adicity: its multiplicative group order is divisible by 2^{27}, the largest such power available at this width, which maximizes the size of computation that a radix-2 FFT can address [5]. KoalaBear, $p = 2^{31} - 2^{24} + 1 = 2{,}130{,}706{,}433$, trades two-adicity down to 24 in exchange for an algebraic property: it is chosen so that the cube map $x \mapsto x^3$ is an automorphism of $\mathbb{F}_p^\times$ [5]. That property matters because arithmetization-friendly hash functions such as Poseidon and Poseidon2 use a low-degree power map $x \mapsto x^d$ as their non-linear S-box [27, 28], and x^d is a permutation of $\mathbb{F}_p$ exactly when $\gcd(d, p - 1) = 1$. The smallest useful exponent is $d = 3$, so the design question is whether $3 \mid (p - 1)$. The two 31-bit primes answer it differently, and the difference is not incidental. For BabyBear, $p - 1 = 2^{27}(2^4 - 1) = 2^{27} \cdot 3 \cdot 5$, so $3 \mid (p - 1)$ and the cube map is *not* a permutation; a Poseidon-style hash over BabyBear must climb to the next coprime exponent, $d = 7$, raising the algebraic degree of every S-box. For KoalaBear, $p - 1 = 2^{24}(2^7 - 1) = 2^{24} \cdot 127$ with 127 prime, so $\gcd(3, p - 1) = 1$ and x^3 is an automorphism, exactly the property recorded in its design rationale [5]. KoalaBear thus buys a cheaper hash S-box at the cost of three bits of two-adicity. The BabyBear/KoalaBear pair is a small natural experiment in the trade between FFT reach and hash-friendliness, both settled inside the Plonky3 toolkit [5].

3.3 The Mersenne prime and the circle construction

The cheapest prime-field arithmetic available at 31 bits belongs to the Mersenne prime $p = 2^{31} - 1$ (M31), for which reduction modulo p is essentially a single add-with-carry of the high and low

halves [6]. The obstacle is structural: the multiplicative group of $\mathbb{F}_p$ has order $2^{31} - 2$, which is divisible by two only once, so it contains no large smooth subgroup and cannot host the radix-2 FFT that classical FRI needs [7]. Mersenne-31 is, in the language of the field, FFT-unfriendly.

The general form of this obstacle had already been addressed. The elliptic-curve FFT (ECFFT) of Ben-Sasson, Carmon, Kopparty, and Levit gives $O(n \log n)$ polynomial algorithms (multiplication, interpolation, low-degree extension) over *all* finite fields, including those with no smooth multiplicative subgroup, by replacing the group of roots of unity with a sequence of sets carrying the two-to-one rational maps that the FFT butterfly needs, drawn from a well-chosen elliptic curve [29]. ECFFT decouples the FFT machinery on which FRI depends from the field-smoothness requirement, and it is the conceptual precursor to the circle construction: Circle STARKs frame their approach explicitly as being "in the lines of ECFFT," realizing the genus-zero special case in which the elliptic curve is replaced by the circle [7, 29].

The circle construction resolves this for the Mersenne prime specifically, and more simply than the general ECFFT. Its coding-theoretic foundation, the Reed–Solomon codes over the circle group, places the evaluation domain not in the multiplicative group of $\mathbb{F}_p$ but on the unit circle $x^2 + y^2 = 1$ of the quadratic ("complex") extension of the Mersenne field [6]. That circle group has order $p + 1 = 2^{31}$, which is fully two-smooth, and it supports a two-to-one folding map (informally, $x \mapsto 2x^2 - 1$) that plays the role of FRI's folding [7]. Circle STARKs build the full low-degree test and STARK protocol on this base. The construction paper reports a preliminary single-threaded benchmark in which the circle FFT runs about 1.4× faster than a classical FFT over BabyBear; it projects, but does not measure, the same advantage for a fully optimized whole-STARK implementation, and its own figures show the margin narrowing toward parity at large sizes once the implementation becomes memory-bound and multi-threaded [7]. The circle-group codes were motivated precisely as an "almost native" alternative expected to match the performance of native Reed–Solomon codes over more FFT-friendly primes [6]. The production realization is the Stwo prover, which operates natively over Mersenne-31 [30].

The Circle STARKs paper states its soundness as an explicit error bound (Theorem 7) over a generic extension field F, not as a fixed bit-security figure [7]. The concrete instantiation, in particular the choice of F as the quartic extension QM31 of M31, is fixed by the deployed prover rather than by the construction paper (Sections 4.4 and 6.4).

3.4 Binary tower fields

The most radical entry abandons prime fields. Binius builds its arguments over a tower of binary fields $\mathbb{F}_2 \subset \mathbb{F}_{2^2} \subset \mathbb{F}_{2^4} \subset \cdots \subset \mathbb{F}_{2^{128}}$, constructed by iterated quadratic extension in the manner of Wiedemann: starting from $T_0 = \mathbb{F}_2$, each level adjoins a root of $X^2 + X_{k-1}X + 1$, so that $|T_k| = 2^{2^k}$ [8, 24]. The levels grow doubly-exponentially in width ($T_0 = \mathbb{F}_2$, $T_1 = \mathbb{F}_4$, $T_2 = \mathbb{F}_{16}$, $T_3 = \mathbb{F}_{256}$, and on to $T_7 = \mathbb{F}_{2^{128}}$), and each level is a two-dimensional vector space over the one below it, so that a value can be viewed at whatever width its data demands and lifted into a larger level only when the protocol needs a bigger challenge space [8]. This nesting is exactly what the "no embedding overhead" property (Section 5.3) exploits: a bit lives in T_0, a byte in T_3, and neither is padded to a machine word. Efficient recursive algorithms for inversion and constant multiplication on such towers were given by Fan and Paar, who reduce an inversion at level k to operations at level $k - 1$, giving the tower practical arithmetic to match its structural elegance [25]. The payoff is twofold. Addition in a binary field is a bitwise XOR, which is free in hardware, in contrast to the wide integer multipliers with carry propagation that prime fields require for their arithmetic [8]. And a value that is genuinely one bit (a Boolean wire, a bit of a Keccak state) is stored in one bit of $\mathbb{F}_2$, with no wasted headroom. This second property, developed in Section 5, is the one no prime field can match: Binius commits to small-field polynomials "with no embedding overhead" [8].

3.5 Provenance: two ways a field arrives

The families in Table 1 differ in their parameters, and they also differ in how they entered the field. Mersenne-31 and the binary towers each arrived through a *construction paper* whose mathematical content was what made the field usable for proving: the circle-group codes for the former [6, 7], and the tower construction together with its succinct-argument scheme for the latter [8, 24]. Goldilocks and BabyBear arrived through *engineering*. Goldilocks is defined and justified in a field-selection section of a prover's technical report [4]; BabyBear is defined by its implementation in a toolkit, its rationale recorded in source comments and documentation rather than a theorem [5]. Neither has a construction paper because neither needed one: they are fast primes that were measured, adopted, and standardized. The claim is about how the field arrived, not about whether mathematics is present in it: Goldilocks is a Solinas-shaped prime chosen for its reduction, and KoalaBear's x^3-automorphism condition (Section 3.2) is an algebraic design decision. The distinction is that no enabling theorem was *needed* to make these fields usable, whereas one was for M31 and the binary towers. We take this asymmetry up again in Section 7, because it predicts something about where the next field will come from.

3.6 Field and hash co-design

The taxonomy so far treats the field as the object, but the field's largest single consumer of prover cycles is the hash, and hash design has become tightly coupled to field choice. Section 3.2 gave one instance: the S-box exponent of a Poseidon-style permutation is a function of the field, since x^3 is a permutation only when $3 \nmid p - 1$, forcing $d = 7$ over BabyBear and $d = 3$ over KoalaBear [27, 28]. The coupling is tighter in the newer designs. Monolith is an arithmetization-oriented hash co-designed with Goldilocks and Mersenne-31, whose lookup-based non-linear layer is defined over the specific bit-decomposition those moduli admit, making it cheap in-circuit while reaching a native speed comparable to SHA3 [31]. At the other end of the field spectrum, Skyscraper targets the large 256-bit primes used in the SNARK-wrapping step, trading small-field lookup tricks for low-degree non-invertible maps and minimized modular reductions [32]. The two mark the poles of a field-partitioned hash landscape: the same goal, an arithmetization-oriented hash at native speed, yields a different construction over a 31- or 64-bit field than over a 256-bit one, because the operations that are cheap change with the field. For a survey whose unit of analysis is the field, this is where field choice propagates into the most expensive component of the prover, and it is a reason the families of Table 1 are not freely interchangeable even where their soundness and overhead profiles align.

---

4. Extension-field security

A small base field is unsound on its own. This section explains why, recalls the two results that let a small field be paired with a large challenge field without loss of soundness, assembles across systems the relationship between base field and required extension degree, and closes with the caveat that the security targets in question are conjectural.

4.1 Why a small base field cannot supply its own challenges

The Fiat–Shamir soundness bound of Section 2.5 is unforgiving at small widths. A challenge drawn uniformly from a 31-bit field is guessed with probability about 2^{-31}; a proof that draws its randomness there admits a cheating strategy that succeeds with non-negligible probability, far above any cryptographic target [10]. The remedy, universal across the systems we survey, is to draw challenges from an extension field $K = \mathbb{F}_{p^e}$ large enough that $1/|K|$ is negligible, while

keeping the committed data (the trace, the witness) in the small base field where arithmetic is cheap. ethSTARK states the principle directly: for the protocol to be secure "one must use a larger field in some places," and it draws its constraint-combination coefficients, its out-of-domain evaluation point, and its FRI folding randomness from a quadratic extension of the base field [10]. In this split the base field carries the computation, and the soundness comes from the larger extension field.

4.2 The DEEP method: sampling outside the domain

Restoring a large challenge space is necessary but not sufficient; the soundness of FRI itself must also be strengthened, and the mechanism is the DEEP method, "Domain Extending for Eliminating Pretenders" [33]. The idea is that the verifier asks the prover for the value of the committed polynomial at a point z drawn from the *extension* field, outside the evaluation domain of the code, and uses the answer to reduce a claim about multiple polynomials to a claim about one [33]. Because z lives in a large field, a cheating prover cannot have prepared for it; sampling "outside the box" in this way improves the soundness of the low-degree test while preserving FRI's linear proving complexity [33]. DEEP is the standard soundness-boosting step in every FRI-based small-field prover, and it is what ties the extension field of Section 4.1 to the low-degree test of Section 2.2: the out-of-domain point is exactly one of the challenges that the extension must be large enough to hide.

4.3 Proximity gaps: soundness of batched FRI over small fields

The second ingredient concerns batching. FRI is almost always applied to a random linear combination of many committed polynomials rather than to each separately, and the soundness of that batching rests on a *proximity gap* [16]. Informally, the proximity-gap theorem for Reed–Solomon codes states that for an affine space of functions (such as the line spanned by two functions, or the coset formed during folding), one of two things holds: either almost every member of the space is δ-close to the code, or only a small fraction is [16]. There is no middle ground, no affine space in which a constant but non-trivial fraction of members are close. Quantitatively, below the unique-decoding radius (half the minimum distance) of a code of block length n over $\mathbb{F}_q$, every affine space is either entirely δ-close to the code or has at most an n/q fraction of members that are, so that the soundness error of the batching step scales as $O(n/q)$ [16]. Two consequences bear on field choice. The error scales *inversely* with the field size q from which the batching randomness is drawn, which is a second reason the challenge field must be large. And the gap persists even over small fields, with size polynomial in the code dimension, provided the distance stays below the Johnson/Guruswami–Sudan list-decoding bound [16]. Pushing δ up toward the list-decoding radius tightens query complexity but requires care with the field size; this is the regime in which deployed systems operate, and it is where the conjectural element of Section 4.5 enters.

The classical proximity-gap theorem exposes non-closeness using a single uniformly random element of the affine space, and that randomness must come from the large challenge field. For the binary-field branch this is refined. Diamond and Posen prove a proximity result that needs only a *logarithmic* number of random field elements, drawn with tensor-product structure, at the cost of only a logarithmic loss in the false-witness bound [34]. This tensor proximity gap is the antecedent of the soundness argument for the binary-tower commitment: it lets a prover be tested with tensor-structured challenges rather than a full random linear combination. FRI-Binius's own security proof leverages a further proximity gap in this line, established by Diamond and Gruen for interleaved codes, to make its batched evaluation and folding checks sound over characteristic two [9, 35]. The result matters for field choice because it shows the proximity-gap machinery of Section 4.3 is not tied to large prime fields; it extends, with the right sampling structure, to the smallest field there is.

The query complexity of the low-degree test is not fixed, and a line of FRI successors has reduced it, with consequences for the soundness budget of Section 2.5. STIR lowers the number of queries by shrinking the evaluation domain across rounds rather than merely folding, achieving a target security level with fewer authenticated openings and therefore shorter proofs [36]. WHIR extends the same idea to a broader class of codes and adds a fast verifier [37]. These protocols do not change the base-field-versus-challenge-field split that organizes this section (challenges still come from the extension, and soundness still scales with $|K|$), but they reduce the $(1-\delta)^{\ell}$ query term for a fixed ℓ, which in turn eases the grinding a small field must add to reach its target. They are the reason the query counts and proof sizes reported for the newest small-field provers are lower than the original FRI analysis would suggest, and any current account of the soundness budget must treat the low-degree test as a moving component rather than a fixed cost [36, 37].

4.4 The extension-degree-versus-security relationship

The systems we survey each state, individually, how large an extension they use and why. Assembled, these statements make a cross-system pattern visible: the extension degree required for a fixed security target tracks the base-field width, because achievable security is capped at $\log_2 |K| = e \log_2 p$ (Section 2.5). The relationship itself is elementary; what the assembly adds is the record of where deployments sit against that cap. Table 2 collects the deployed choices.

Table 2. Extension degree versus security target, by base field. *(All rows are drawn from the cited primary sources; the binary-tower row is a structural bound rather than a deployed configuration.)*

Base field	bits	Deployed extension	Secure-field size	Notes	Src.
ethSTARK $2^{61}+20\cdot 2^{32}+1$	61	e=2 (80/100-bit); e=3 (128-bit)	$\approx 2^{122}$–2^{183}	security $\leq \log_2 \lvert K\rvert - 1$	[10]
Goldilocks	64	e=2, $\mathbb{F}_p[X]/(X^2-7)$	$\approx 2^{128}$	quadratic ext. for soundness	[4]
BabyBear	31	e=4 (deployed)	$\approx 2^{124}$	degree-4; 128 via list-decoding + grinding	[5, 12]
Mersenne-31	31	e=4 (QM31 tower)	$\approx 2^{124}$	list-decoding + grinding; not by $\lvert F\rvert$ alone	[7, 30]
Binary tower	1	up to $\mathbb{F}_{2^{128}}$	$\approx 2^{128}$	challenge field via ring-switching (Sec. 7.1)	[8, 9]

Three points structure the table. For the wide fields, a quadratic extension suffices at the usual targets: Goldilocks pairs with $\mathbb{F}_p[X]/(X^2-7)$ for a secure field of about 2^{128} [4], and ethSTARK's 61-bit prime reaches 80- and 100-bit targets with $e=2$, needing $e=3$ only for 128-bit security, exactly because achievable security is capped by $\log_2 |K|$ [10]. For the 31-bit fields the degree doubles: a degree-four extension is required. The zkVM systematization states this as small fields "requir[ing] degree-4 extensions for 128-bit security," but it groups Goldilocks with BabyBear under that heading [12]; the grouping is imprecise, since a 64-bit base already reaches 2^{128} with a degree-two extension. The degree-four requirement is a property of the 31-bit width specifically, and we read the zkVM summary as corroborating the 31-bit rows only, consistent with that work's own more precise statement elsewhere, which ties the degree-four requirement to a 31-bit field [12]. For Mersenne-31 the extension is realized as a tower: a quadratic "complex" extension of M31 followed by a further quadratic step to the quartic field QM31, of size about 2^{124}, confirmed as the deployed choice by the Stwo whitepaper [7, 30]. Here the distinction

between field size and security level must be drawn carefully. The Circle STARKs construction attaches no bit-security number to QM31; it bounds the soundness error of its AIR-satisfiability protocol by an expression of the form $\ell^+\left[(C-1)/|F| + d(N+2)/(|F|+1-2^{n+B})\right] + \varepsilon_{\mathrm{PROX}}$ (Theorem 7), in which ℓ^+ is a list-size bound, $\varepsilon_{\mathrm{PROX}}$ is the proximity-test error, and the achievable security depends on the code rate, the list-decoding radius, and the query count, not on $|F|$ alone [7]. The deployed prover reaches its target in the list-decoding regime with proof-of-work grinding, 26 bits in Stwo, rather than by field size alone [30]. The "small field plus extension" pattern is not a single recipe. The narrower the base field, the higher the extension degree needed to reach the same target, and the arithmetic saved in the base field is partly repaid in extension-field operations, the overhead the zkVM systematization quantifies at 9–16× per extension operation [12]. We stress that Table 2 is an empirical pattern across the deployed prime fields rather than a derived law; the binary-tower row is included for completeness, but its challenge-extension is not analyzed here, and a single account that also covers it remains open (Section 7.1).

4.5 Conjectured versus provable soundness

A survey of field choice that reported the numbers of Section 4.4 as settled would mislead, because the soundness of non-interactive FRI at deployed parameters is conjectural. The FRI soundness analysis used in practice operates in the list-decoding regime and assumes that the best known attack is optimal; deployed parameters are then set against that conjecture [11]. A recent concrete-security analysis quantifies the resulting exposure [11]. Under the conjecture, deployed Fiat–Shamir FRI parameters nearly meet their targets, coming within about one bit [11]. But their *provable* security (what can be shown without the conjecture) sits substantially lower, between 21 and 63 bits below the conjectured level for all but one of the parameter sets examined [11]. The practical implication is twofold: the security levels attached to small-field systems are conjectural upper bounds, and, as the analysis notes, a better attack than the ones currently known would compromise the security of deployed systems rather than merely tightening an analysis [11]. A second, independent result points the same way. Fenzi and Sanso show that hash-based SNARGs over small base fields are information-theoretically less sound than conjectured, because the extension codes built from a small field admit large decoding lists that force a non-negligible soundness error, cutting concrete security by roughly ten bits for the 31-bit fields (8 to 11 bits across rates in their Table 1) and, more mildly, by a few bits (3 to 6) even for the 64-bit Goldilocks field [13]. That the loss reaches even the 64-bit field, though it shrinks as the field widens, is a further caution against reading the field sizes of Table 2 as security levels. Any "128-bit" claim in Table 2 should be read with these qualifiers attached.

4.6 Deployed parameters

The abstract account of the preceding subsections becomes concrete in the parameters that systems actually ship, collected in Table 3. Only the ethSTARK row is drawn from a peer-reviewed source; the rest come from vendor whitepapers, documentation, or configuration source, and are labeled accordingly.

Table 3. Deployed FRI parameters. *(Prov.: PAPER = peer-reviewed; DOCS = vendor whitepaper/docs/source. Gaps and inferences flagged in notes.)*

System	Base field	Ext.	Rate ρ (blowup)	Queries	Grind	λ	Prov.
ethSTARK	∼61-bit prime	2→3	1/4 (4)	31/41/55	20	80/100/128	PAPER
Plonky2 (rec.)	Goldilocks	2	1/8 (8)	28	16	100	DOCS
RISC Zero	BabyBear	4	1/4 (4)*	50	n/f†	96–99‡	DOCS
SP1 Turbo	BabyBear	n/a§	variable	100/logbl.	16	100	DOCS
Stwo	M31 (QM31)	4	1/2 (2)	90	26	100	DOCS‖

*rate inferred, not a boxed constant. †shipped grinding bit-count not found. ‡three figures for three layers (on-chain / base STARK / recursion). §SP1 Turbo extension degree not confirmed in docs. ‖Stwo's whitepaper example parameters (100-bit

conjectured, rate 1/2, 90 queries, 26-bit grinding). The naive "β bits per query" relation is optimistic: correlated agreement holds only to the Elias radius ($\approx 0.83\beta$–0.91β; Crites–Stewart).

ethSTARK, as the anchor row, states its soundness budget as a formula: the conjectured security is

$$\lambda \;\geq\; \min\{\,\zeta + R \cdot s,\; \log_2 |K|\,\} - 1,$$

where ζ is the grinding bit-count, s the number of FRI queries, $R = \log_2(1/\rho)$ the bits gained per query at rate ρ, and K the challenge extension field [10]. Instantiating it exposes the mechanism behind the extension-degree jump of Section 4.4. With $\zeta = 20$, rate $\rho = 1/4$ (so $R = 2$), and the quadratic extension ($\log_2 |K| \approx 122$), the query term $\zeta + R \cdot s$ reaches the 80- and 100-bit targets at $s = 31$ and $s = 41$ while staying under the field-size cap. But the 128-bit target needs $\zeta + R \cdot s \approx 130$, which *exceeds* $\log_2 |K| \approx 122$: the extension-field size, not the query count, becomes the binding constraint, and the only remedy is to enlarge K by moving to the cubic extension [10]. The degree jump from two to three is thus forced by the min in the formula, not chosen for convenience. Two patterns hold across the rest of the table. First, the rate–query trade is visible directly: Stwo's aggressive rate 1/2 costs it 90 queries, ethSTARK's 1/4 needs 31, and Plonky2's conservative 1/8 needs only 28. Second, grinding of 16 to 26 bits is universal, confirming that the "buy back the shortfall" mechanism of Section 2.5 is not a theoretical device but standard practice. Every target in the table is *conjectured* security, which returns us to the caveat of Section 4.5: none of these deployed levels is provable at the stated value [11].

The rate–query trade has a cost structure worth making explicit, because it is where field choice meets prover economics. Reading the ethSTARK formula backwards, reaching a target λ from the query term alone needs $s \approx (\lambda - \zeta)/R$ queries, with $R = \log_2(1/\rho)$ bits per query; halving the rate adds one bit per query and removes queries, while doubling it removes a bit and demands more. But the rate also fixes the blow-up factor ρ^{-1}, and the blow-up drives the two dominant prover costs. Committing the Reed–Solomon encoding is an FFT over a domain of size ρ^{-1} times the trace length, so prover time and the memory bandwidth of the commitment both scale roughly with ρ^{-1}; and each of the s queries opens a Merkle path of length $\log(\rho^{-1}T)$, so proof size scales with $s \log(\rho^{-1}T)$. The two effects pull against each other. A conservative rate (Plonky2's $\rho = 1/8$) buys $R = 3$ bits per query and a small proof at 28 queries, but pays an eight-fold blow-up in FFT work and hashing bandwidth; an aggressive rate (Stwo's $\rho = 1/2$) halves the blow-up, cutting prover time and bandwidth, but earns only $R = 1$ bit per query and so needs about 90 of them, enlarging the proof. The field enters this trade twice: the base field sets the per-element arithmetic cost of the FFT (Section 3), and the challenge extension sets the cap $\log_2 |K|$ that, once the query term would exceed it, forces the extension-degree jump rather than more queries. A prover targeting on-chain verification, where proof size is the scarce resource, is pushed toward low rates and high blow-up; one targeting throughput on constrained hardware is pushed the other way. The whole trade is priced against the list-decoding soundness of FRI, so, as Section 4.5 cautions, a better attack would reprice every row of Table 3 [11, 36].

Three caveats qualify the table. The per-query yield $R = \log_2(1/\rho)$ used above is the conjectured value; correlated agreement is now known to hold only up to the Elias radius, so the true yield is nearer 0.83 to 0.91 of it, which tightens rather than relaxes the query counts a target demands. The grinding term is quantum-fragile: a prover running Grover's search halves the effective difficulty of a proof-of-work, so ζ bits of grinding are worth about $\zeta/2$ against a quantum adversary. The surveyed systems are otherwise plausibly post-quantum, resting only on the collision-resistance of a hash, but a post-quantum reading of Table 3 should halve the grinding column before summing, which erodes precisely the "buy back the shortfall" mechanism they lean on. And the deployed targets themselves vary: several systems ship 96- to 100-bit conjectured security rather than 128, so the 128-bit figure we use as a reference point is an upper anchor, not a universal deployment target.

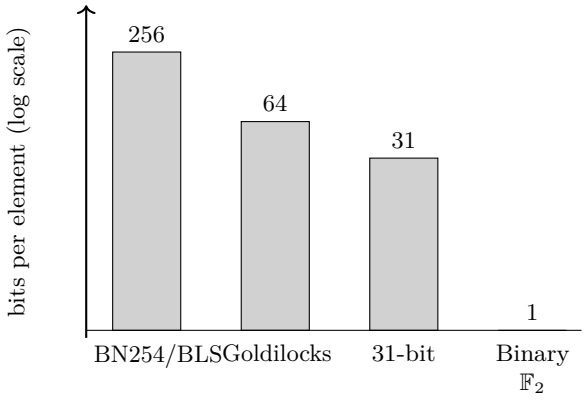

Figure 3: Field-element width across the surveyed families (Section 5). Encoding a single bit of computation consumes a full field element, so a wider field over-provisions: a 256-bit field spends 256 bits per bit of payload, Goldilocks 64, a 31-bit prime 31, and a binary field exactly one. This is a measure of embedding headroom, not of end-to-end prover cost; no controlled cross-field performance comparison exists (Section 5.5, Table 4).

5. Embedding and arithmetization overhead

The first motivation for small fields, given in the introduction, was that a 256-bit field wastes the hardware. This section makes the argument precise, separates the peer-reviewed content from the vendor benchmarks that dominate this part of the literature, and states plainly the place where no rigorous comparison yet exists.

5.1 The embedding problem

Embedding overhead is the cost of representing a datum in a field wider than the datum requires. Its clearest quantification is due to the Jolt analysis, which observes that a prover over a 31-bit field commits, per RISC-V step, to at least 275 field elements, and that this is roughly equivalent to committing to 34 full 256-bit field elements per step once the widths are reconciled [1]. The equivalence is a data-volume proxy rather than a commitment-cost identity: it compares total committed bits, whereas the cost of a Merkle commitment scales with the number of leaves, not their width. As a first-order measure of how much a field over-provisions, though, it is the best-anchored figure the peer-reviewed literature currently offers, for want of a controlled measurement. The comparison cuts both ways: it shows the small field doing the same work at a fraction of the committed-data volume, and it shows that even a 31-bit field is not free of waste, since 275 elements is still more than the underlying computation strictly needs. The overhead is largest for large fields and shrinks, without vanishing, as the field narrows [1].

5.2 Range checks and non-native arithmetic

Embedding overhead has a second face that the raw-width accounting misses: even when a field is narrow, computation whose data type does not match the field must be simulated. A 32-bit integer addition in a 31-bit prime field overflows the field and must be decomposed with a carry and a range check; a 64-bit multiplication becomes several field multiplications with range-checked limbs; a bitwise operation in a prime field has no native counterpart at all and is realized through lookup arguments [1,3]. These costs are charged back to the trace as additional columns and constraints, and they partly offset the arithmetic savings of the narrow field. The narrowing of the field, in other words, trades one overhead (wasted width) for another (simulation of non-native operations), and the balance depends on the workload:

hash-heavy computations that can be expressed in field-native S-boxes favor the prime fields, whereas bit-heavy computations pay range-check tolls that the binary fields avoid.

A concrete case fixes the magnitude. Consider a 32-bit unsigned addition $c = a + b \bmod 2^{32}$ arithmetized over Mersenne-31, whose modulus is $2^{31} - 1$. The mathematical sum $a + b$ can reach $2^{33} - 2$, which exceeds both 2^{32} and the field modulus, so the operation cannot be a single field addition. A standard arithmetization splits each 32-bit operand into limbs, computes the sum with an explicit carry bit, and constrains the result with range checks that force each limb into its intended interval: several trace cells and constraints for one machine-level instruction. The range checks themselves are not free: modern arithmetizations discharge them with lookup arguments, proving that each limb appears in a precomputed table of admissible values, which is exactly the role Lasso plays and the reason lookups are the dominant cost center in small-field zkVMs [1, 21]. Far from a rounding error, the width mismatch between a 32-bit datum and a 31-bit field is a structural tax paid on every non-native operation. It is the strongest single argument for a field whose native width already matches the data, whether a 64-bit field that swallows 32-bit values whole or a binary field that represents them bit-for-bit.

5.3 The no-embedding-overhead property of binary fields

The binary-tower branch is defined by closing the embedding gap entirely. Because the tower includes $\mathbb{F}_2$ itself, a one-bit value is committed as one bit, and Binius's commitment scheme is constructed so that, unlike prior work, it "treats small-field polynomials with no embedding overhead" [8]. This is a structural rather than a quantitative claim: it is not that the overhead is small but that, by committing directly over the tower with a packing that respects the sub-field structure, there is no embedding step to pay for [8]. FRI-Binius preserves this property while replacing the square-root-sized Brakedown commitment with a polylogarithmic one, so that the no-overhead encoding is retained alongside small proofs [9]. Among the families of Section 3, this property is unique to the binary towers; every prime field, however narrow, pays some embedding cost for values below its width.

5.4 The hardware argument, and its provenance

A further argument for binary fields is made at the level of hardware: addition is XOR and comes for free, whereas prime-field arithmetic requires wide integer multipliers with carry propagation, so a binary-field design maps more cleanly onto silicon [38]. The most-cited figures for this argument come from the vendor rather than the peer-reviewed literature. The Binius developers report that a 256-bit prime field wastes throughput even relative to a small prime field (that a 64-bit field still "leaves a 64× savings on the table" per bit) and that a 32-bit binary-field operation can be several times cheaper than a Mersenne-31 operation [38, 39]. Widely-read technical expositions make the same case [40]. We report these figures because they are the ones the field cites, but we label them as vendor and blog sources, not peer-reviewed measurements, and we note that they are illustrative rather than end-to-end: binary-field *multiplication* still requires a carryless multiply followed by reduction, so the "free XOR" applies to addition, not to the full arithmetic [39].

5.5 An honest gap

The overhead literature is asymmetric, and Table 4 makes the asymmetry visible by collecting the figures the field actually cites, each tagged by the strength of its source.

Table 4. Reported overhead and arithmetic-cost figures. *(PR = peer-reviewed; V = vendor/blog. The provenance column is the point: the sharp cross-field ratios are all vendor-sourced, and no two rows share a workload, cost model, and commitment scheme.)*

Figure	Reported value	Fields compared	Prov.	Src.
Jolt prover commitment	≈6 256-bit elts/step	256-bit (Jolt)	PR	[1]
RISC Zero commitment	≥275 31-bit ≈ 34 256-bit/step	BabyBear vs 256-bit	PR	[1]
Cairo-VM commitment	≈50 elts ≈ 13 256-bit/step	Cairo-VM vs 256-bit	PR	[1]
Circle FFT (1-thread)	≈1.4× vs BabyBear FFT	M31 vs BabyBear	PR	[7]
No embedding overhead	structural (not a ratio)	binary tower	PR	[8]
M31 arithmetic	≈1.3× vs BabyBear	M31 vs BabyBear	V	[41]
Bit-commitment	≈50× vs Plonky2	binary vs Goldilocks	V	[39]
"Savings on the table"	≈64× per bit	Goldilocks vs binary	V	[39]
Hardware efficiency	≈5× vendor; 2.9× throughput (1428/500 Mmult/s)	binary-32 vs M31	V	[38]
FRI-Binius proof size	3.5 vs 11.5 MiB (2^{32}-coeff $\mathbb{F}_2$)	Binius PCS variants	V	[42]

The embedding argument has a peer-reviewed anchor [1], and the no-embedding-overhead property of binary fields is a published structural result [8]. The peer-reviewed rows, however, are not mutually comparable: the three commitment-cost figures are normalized to a single unit only by the Jolt authors' own stated 8-to-1 hashing assumption, a normalization the paper itself calls "crude," and they hold data width and commitment scheme constant only informally [1]. The sharpest cross-field claims, the 5×, 50×, and 64× ratios that most directly argue the binary-field case, are all vendor-sourced, and each uses a different cost model (per-operation compute, bit-commitment throughput, silicon gate depth). The *controlled* comparison a survey most wants (a head-to-head measurement of arithmetization and embedding overhead for one common workload across Goldilocks, BabyBear, Mersenne-31, and the binary towers, accounting for range checks and non-native arithmetic under matched security parameters) does not exist in the peer-reviewed literature. We flag this as the sharpest empirical gap in the area, and return to it in Section 7. The hardware-advantage premise motivating the binary-field branch, and part of the motivation for the small-field turn as a whole, rests at present only on vendor-strength evidence: the peer-reviewed literature quantifies embedding waste (Section 5.1) but not the end-to-end hardware win. We note that this premise is not established at peer-reviewed strength. A bounded scalar micro-measurement (Appendix A) illustrates the point from the other side: raw field-multiply latency for the three prime fields differs by only about a tenth on one core and does not reproduce the vendor ratios, because those live in the SIMD and hardware regimes a scalar core cannot exhibit.

---

6. Provers mapped to fields

The field families of Section 3 are legible through the provers that adopted them, and each system's documentation records a rationale that corroborates the taxonomy. Table 5 gives the map; the subsections add the reasoning and correct one common misattribution.

Table 5. Production provers and their fields.

System	Field	Prime	Challenge ext.	Rationale	Src.
Plonky2	Goldilocks	$2^{64}-2^{32}+1$	e=2, X^2-7	word fit + constant-free reduction	[4]
Miden VM	Goldilocks	$2^{64}-2^{32}+1$	e=2, 3	same field as Plonky2; FFT-friendly	[26]
Plonky3	multi	BB/KB/M31/Gold.	per field	multi-field toolkit; SIMD backends	[5]
SP1	BabyBear	$2^{31}-2^{27}+1$	via Plonky3	RISC-V zkVM on Plonky3	[5, 43]
RISC Zero	BabyBear	$2^{31}-2^{27}+1$	e=4	RV32IM; own STARK stack	[1, 44]
Stwo	Mersenne-31	$2^{31}-1$	QM31 (e=4)	cheapest prime arith.; circle group	[7, 30]
Binius	binary tower	char. 2	binary ext.	no embedding overhead; hardware fit	[8, 9]
ethSTARK	61-bit prime	$2^{61}+20\cdot 2^{32}+1$	e=2→3	Rescue-native; reference STARK	[10]

6.1 The Goldilocks provers: Plonky2 and Miden

Plonky2 is the system that established the 64-bit prime as a practical proving field, and its report's field-selection section is the primary source for the Goldilocks rationale of Section 3.1 [4]. Its full title ("Fast Recursive Arguments with PLONK and FRI") records a second motivation that the field serves: recursion, in which a proof verifies another proof, so that many proofs can be aggregated into one [4]. Recursion makes the prover run the verifier's own hashing and field arithmetic inside the circuit, which puts a premium on a field whose operations are cheap to arithmetize and whose two-adicity supports the FFTs of the inner verification; Goldilocks was chosen with this recursive workload in view, not only the base proving cost [4]. The Miden virtual machine adopts the same field for the same reasons, citing its efficient reduction and high two-adicity, and drawing its challenges from small extensions for security [26]. That two independent systems settled on the identical prime is itself evidence that at the 64-bit width the design space is narrow. Goldilocks is close to the unique good choice.

6.2 The multi-field toolkit: Plonky3 and its downstream

Plonky3 is not a single prover but a toolkit that supports several fields (BabyBear, KoalaBear, Mersenne-31, and Goldilocks) behind a common arithmetization and commitment interface, each with hand-tuned SIMD backends for AVX2, AVX-512, and NEON [5]. This design turns field choice into a configuration decision rather than a rewrite, and it is the clearest institutional expression of the small-field turn: the toolkit treats the field as a swappable component and lets the workload pick. Its downstream systems exercise that flexibility. SP1, a RISC-V zkVM from Succinct, is built on Plonky3 over BabyBear and is deployed as a proving backend for rollups [5, 43]; other zkVMs in the ecosystem, such as Valida, use the same AIR-plus-FRI machinery [5]. Because the fields share an interface, a system can also switch as the evidence changes: the migration of parts of the ecosystem toward KoalaBear and Mersenne-31 as their backends matured is a live example of field choice remaining contested rather than settled [5]. Plonky3 is thus both an artifact of the turn and an instrument of it, lowering the cost of adopting whichever field a workload favors.

6.3 A correction: RISC Zero runs its own stack

A common secondary-source claim groups RISC Zero with the Plonky3 provers. The grouping is inaccurate. RISC Zero is a zkVM for the RV32IM instruction set that operates over BabyBear, but it runs its own STARK proving stack rather than Plonky3 [1, 44]. The accurate statement is narrower and still supports the taxonomy: RISC Zero is a BabyBear system, and the Jolt analysis of its per-step commitment cost (Section 5.1) treats it as such [1]. We record the correction because the misattribution is widespread and because a survey of field choice should get the field-to-system map right even where it does not get the framework right.

6.4 Mersenne-31 in production: Stwo

The circle construction of Section 3.3 reaches production in Stwo, StarkWare's prover over Mersenne-31, which realizes the QM31 quartic secure field for its challenges, applies 26 bits of proof-of-work grinding, and proves Cairo programs through the Circle STARK protocol [7, 30]. Stwo is the existence proof that the FFT-unfriendliness of the Mersenne prime, long a reason to avoid it, is fully surmountable, and that the field's arithmetic advantage can be cashed out end-to-end.

6.5 Binary towers: Binius and FRI-Binius

Binius is the prover of the binary-tower branch. It pairs a HyperPlonk-style constraint system [20] and a Lasso-style lookup argument [21], both adapted to characteristic two, with a Brakedown-based multilinear commitment over the tower [8, 17]. The combination is what lets it capture bit-oriented computations such as Keccak-256 and Grøstl natively, with the no-embedding-overhead property of Section 5.3 [8]. The original construction paid for this with square-root-sized proofs, the characteristic cost of a Brakedown commitment [8, 17]. FRI-Binius removes that cost. Its central device is *ring-switching*: a sumcheck-based compiler that turns a multilinear commitment over a large extension field into one over that field's ground field (here $\mathbb{F}_2$) with no embedding overhead, so that the per-input commitment cost equals the large-field scheme's cost for an input of the same bit-size [9]. Instantiated on a characteristic-two adaptation of BaseFold's foldable-code commitment [18] and the reinterpreted Lin–Chung–Han basis [19], ring-switching brings a multilinear FRI-folding to $\mathbb{F}_2$ and reduces proof size to polylogarithmic while retaining the packed encoding, with soundness resting on the tensor proximity gap of Section 4.3 [9, 34]. FRI-Binius reports an order-of-magnitude reduction in proof size over the original Brakedown-based construction [9]; the developers' worked example shows 3.5 versus 11.5 MiB for a 2^{32}-coefficient $\mathbb{F}_2$ multilinear, about a threefold reduction at that size, which we treat as vendor-reported (Section 5.4) [42]. Together, Binius and FRI-Binius occupy the corner of the design space that the prime-field systems cannot reach: computation whose natural width is a single bit.

6.6 The reference point: ethSTARK

ethSTARK predates the 31-bit generation but belongs in the map as the reference instantiation of the small-field-plus-extension pattern, operating over a 61-bit prime and drawing all verifier randomness from a quadratic (or, for 128-bit security, cubic) extension [10]. Its native hash, Rescue-Prime, is an arithmetization-oriented design built on a power-map S-box, so the field-versus-hash-friendliness considerations of Section 3.2 apply to it as directly as to the Poseidon-based 31-bit systems [45, 46]. It is the system whose documentation states the design principle most explicitly, and it anchors the wide end of Table 2.

---

7. Open problems and future directions

The systematization surfaces four problems that no current work resolves. We state them as open problems rather than dress them as results; the survey's contribution is the taxonomy and the cross-system assembly, and these four are the gaps that remain.

7.1 A unified extension-degree-versus-security law

Table 2 assembles the deployed extension degrees across base fields, but it is a table of *practice*, not a theorem. Each system fixes its extension degree against its own security analysis and its

own base field, and the resulting pattern (quadratic for the 61- and 64-bit fields, quartic for the 31-bit fields) is empirical. A single account that derives the required extension degree as a function of base-field width and security target, uniform across the prime fields and reconciled with the binary-tower case, would replace the table with a law. The ingredients exist in the proximity-gap and DEEP analyses [16, 33]; what is missing is the synthesis, particularly for the 31-bit fields that define the modern turn.

The binary-tower case admits a partial answer at the elementary level, though the substantive question stays open. In the ring-switching commitment of FRI-Binius, verifier challenges are drawn from an extension L of the small field K of power-of-two degree 2^κ, and the field-size contribution to the soundness error is $1/|L|$ [9]. For a binary tower the natural choice of L is a tower level $T_k = \mathbb{F}_{2^{2^k}}$, so $|L| = 2^{2^k}$ and the field-size term alone reaches a target λ once $2^k \geq \lambda$, that is, once the level satisfies $k \geq \lceil \log_2 \lambda \rceil$; for the standard 128-bit target this is $k = 7$, the level $\mathbb{F}_{2^{128}}$ to which the tower is conventionally built (Section 3.4). This is only the challenge-guessing term of the soundness budget, the exact binary restatement of the elementary prime-field bound $\log_2 |K| = e \log_2 p \geq \lambda$; it says nothing about the proximity-gap, query, and grinding terms (Section 4.6) that dominate the real analysis. Its one structural lesson is granularity: a prime field tunes the extension degree e in unit steps, whereas a binary tower tunes the level k in powers of two, so its challenge field jumps from $\mathbb{F}_{2^{64}}$ to $\mathbb{F}_{2^{128}}$ with nothing available in between. As in the prime-field case, working in the list-decoding regime with query repetition and grinding lets the deployed challenge field sit below this naive field-size bound [9]. The genuinely open problem is therefore sharper than "analyze the binary case": it is to state one law that returns the minimal challenge field, expressed as a prime-extension degree or a tower level, for a target λ under a given rate and query budget, uniformly across the two families and their different granularities.

7.2 A controlled cross-field overhead comparison

Section 5.5 identified the empirical gap: there is no peer-reviewed, controlled comparison of arithmetization and embedding overhead across the field families for a common workload. Such a study, fixing a benchmark computation and measuring committed-data volume, range-check burden, and prover time across Goldilocks, BabyBear, Mersenne-31, and the binary towers under matched security parameters, would convert the current patchwork of per-system and vendor benchmarks into evidence. It is the single measurement that would most advance the area, and its absence is conspicuous given the maturity of the implementations.

7.3 Parameters in light of the provable-versus-conjectured gap

The concrete-security analysis of Section 4.5 shows a 21–63-bit gap between provable and conjectured soundness for deployed FRI parameters [11]. Whether the deployed small-field systems have responded (by raising query counts, adding grinding, or widening extensions) or continue to rely on the conjecture for their headline security levels is not documented in one place. A survey of the parameter choices actually shipped, read against the concrete-security bounds, would tell the community how much of its security is proved and how much is assumed.

7.4 Where the next field comes from

Section 3.5 drew a distinction between fields that arrived through a construction paper (Mersenne-31, the binary towers) and fields that arrived through engineering (Goldilocks, BabyBear). The distinction is predictive. Further gains from prime fields appear to be exhausted at the 31-bit width (the arithmetic is nearly free and the remaining costs are structural), which suggests that the next advance, if it is a field advance at all, will come from a construction that unlocks a field previously thought unusable, as the circle group did for Mersenne-31, rather than from

measuring a faster prime. The binary-tower line, still early, is the most likely site of such a construction. Whether the field will continue to shrink, or whether the frontier will move from the field to the commitment scheme, is the open strategic question the taxonomy poses.

---

8. Conclusion

The small-field turn is best understood not as a sequence of optimizations but as the exploration of a design space defined by a single trade: a narrower base field buys cheaper arithmetic and less embedding waste, at the price of soundness that must be repurchased from an extension field and of non-native operations that must be simulated. The systems that carried out this exploration (Plonky2 and Miden over Goldilocks, Plonky3 and its downstream over the 31-bit primes, Stwo over the Mersenne prime through the circle construction, and Binius over binary towers) occupy distinct corners of that space, and each corner records the same three tensions in a different balance. We have organized the fields by their structural properties, assembled across systems the relationship between base-field width and the extension degree soundness requires, separated the peer-reviewed overhead results from the vendor benchmarks, and mapped each production prover to its field. Two threads stand out in closing. The security levels that headline these systems are conjectural, resting on the assumption that known attacks against FRI are optimal, with a measured gap to what is provable [11]. And the fields themselves arrived by two different routes, one mathematical and one empirical: some were unlocked by a new construction, others adopted because measurement showed them fast. That difference, more than any single benchmark, indicates where the frontier will move next. The turn toward small fields has largely run its course within the prime-field families; whether it continues will depend on constructions not yet written.

Limitations and verification status

Apart from the bounded micro-measurement of Appendix A, this survey runs no experiments; every other quantitative figure is reported from a cited source and labeled by the strength of that source, with vendor and blog measurements distinguished from peer-reviewed results (Table 4). The primary sources for every construction, including the Circle STARKs, FRI-Binius, and Stwo papers and the zkVM systematization, were read directly, and the finite-field parameters, congruences, and the S-box automorphism analysis of Section 3 were independently verified. Three limitations remain. First, the extension-degree pattern of Table 2 and the deployed parameters of Table 3 reflect conjectured soundness and shipped configurations at the time of writing; both are moving targets, and several parameter values for the vendor systems, in particular query counts and blow-up factors, are drawn from documentation and source rather than peer-reviewed analysis. Second, the micro-measurement of Appendix A is a single-machine, single-core, scalar result offered only to illustrate the regime-dependence of cross-field ratios; it is not the controlled study whose absence Section 5.5 identifies. Third, the proximity gap on which the FRI-Binius soundness proof rests is a further result of Diamond and Gruen [35] that we cite but do not re-derive here.

Declarations

Data availability. This is a systematization of published literature; it introduces no new datasets. All sources are cited and publicly available.

Ethics. The work involves no human subjects, personal data, or sensitive material.

Funding and competing interests. The author received no funding for this work and declares no competing interests.

Appendix A: An illustrative micro-measurement

This survey runs no experiments beyond the following bounded measurement, which is included to make the regime-dependence of the cross-field ratios in Section 5 concrete and is emphatically *not* the controlled study Section 5.5 calls for. We implemented scalar multiplication in the three prime fields in C, each with its idiomatic reduction: the Mersenne fold for Mersenne-31, Montgomery multiplication for BabyBear (the reduction production implementations use), and the specialized $2^{64} \equiv 2^{32} - 1$ reduction for Goldilocks. Each fast reduction was validated against a straightforward reference implementation, and a dependency-bound multiply loop was timed on a single Apple M1 core with no SIMD. Multiplication ran at about 131 million operations per second for Mersenne-31, about 133 for Goldilocks, and about 122 for BabyBear, with a few percent of run-to-run variation on this passively-cooled core: a spread of under ten percent. Goldilocks is the fastest of the three, and Mersenne-31 is only about 1.07× ahead of BabyBear.

Two features stand out. First, in scalar, latency-bound code the three fields are within about a tenth of one another, and the 64-bit Goldilocks field is not slower than the 31-bit ones; the measurement does not reproduce the vendor claim that Mersenne-31 arithmetic is about 1.3× faster than BabyBear. Section 5.4 gives the reason for treating those ratios cautiously: the 31-bit advantage lives in SIMD lane packing, two elements per 64-bit lane against Goldilocks and roughly eight against a 256-bit field, and in hardware gate counts, none of which a single scalar core exhibits. Second, this times raw field multiplication, not the arithmetization and embedding of a common workload with its range checks and non-native operations, so it is a regime-dependence illustration, not a prover-cost comparison. The C source is included in the paper's source package for reproducibility (`fieldbench.c`).

References

[1] A. Arun, S. Setty, and J. Thaler, "Jolt: SNARKs for virtual machines via lookups," in *Advances in Cryptology – EUROCRYPT 2024*, ser. Lecture Notes in Computer Science, vol. 14656. Springer, 2024, pp. 3–33.

[2] E. Ben-Sasson, I. Bentov, Y. Horesh, and M. Riabzev, "Fast Reed-Solomon interactive oracle proofs of proximity," in *45th International Colloquium on Automata, Languages, and Programming (ICALP)*, 2018.

[3] ——, "Scalable, transparent, and post-quantum secure computational integrity," Cryptology ePrint Archive, Paper 2018/046, 2018. [Online]. Available: https://eprint.iacr.org/2018/046

[4] Polygon Zero Team, "Plonky2: Fast recursive arguments with PLONK and FRI," Polygon Zero, Technical Report (Draft), Sep. 2022. [Online]. Available: https://github.com/0xPolygonZero/plonky2/blob/main/plonky2/plonky2.pdf

[5] ——, "Plonky3: A toolkit for implementing polynomial IOPs," https://github.com/Plonky3/Plonky3, 2024.

[6] U. Haböck, D. Lubarov, and J. Nabaglo, "Reed-Solomon codes over the circle group," Cryptology ePrint Archive, Paper 2023/824, 2023. [Online]. Available: https://eprint.iacr.org/2023/824

[7] U. Haböck, D. Levit, and S. Papini, "Circle STARKs," Cryptology ePrint Archive, Paper 2024/278, 2024. [Online]. Available: https://eprint.iacr.org/2024/278

[8] B. E. Diamond and J. Posen, "Succinct arguments over towers of binary fields," in *Advances in Cryptology – EUROCRYPT 2025, Part IV*, ser. Lecture Notes in Computer Science. Cham: Springer, 2025.

[9] ——, "Polylogarithmic proofs for multilinears over binary towers," in *Advances in Cryptology – EUROCRYPT 2026, Part VII*, ser. Lecture Notes in Computer Science, vol. 16547. Springer, 2026, pp. 3–32.

[10] StarkWare, "ethSTARK documentation," Cryptology ePrint Archive, Paper 2021/582, 2021. [Online]. Available: https://eprint.iacr.org/2021/582

[11] A. R. Block and P. R. Tiwari, "On the concrete security of non-interactive FRI," in *14th Conference on Security and Cryptography for Networks (SCN)*, 2024. [Online]. Available: https://eprint.iacr.org/2024/1161

[12] Y. Yang, Y. Cheng, H. Tang, G. Yang, B. Zhang, and K. Ren, "SoK: Understanding zkVM: From research to practice," in *Proceedings of the 2026 ACM Asia Conference on Computer and Communications Security (ASIA CCS '26)*, 2026. [Online]. Available: https://eprint.iacr.org/2026/525

[13] G. Fenzi and A. Sanso, "Small-field hash-based SNARGs are less sound than conjectured," Cryptology ePrint Archive, Paper 2025/2197, 2025. [Online]. Available: https://eprint.iacr.org/2025/2197

[14] J. Liang, D. Hu, P. Wu, Y. Yang, Q. Shen, and Z. Wu, "SoK: Understanding zk-SNARKs: The gap between research and practice," arXiv preprint arXiv:2502.02387, 2025. [Online]. Available: https://arxiv.org/abs/2502.02387

[15] S. Chaliasos, J. Ernstberger, D. Theodore, D. Wong, M. Jahanara, and B. Livshits, "SoK: What don't we know? Understanding security vulnerabilities in SNARKs," arXiv preprint arXiv:2402.15293, 2024. [Online]. Available: https://arxiv.org/abs/2402.15293

[16] E. Ben-Sasson, D. Carmon, Y. Ishai, S. Kopparty, and S. Saraf, "Proximity gaps for Reed-Solomon codes," *Journal of the ACM*, 2023.

[17] A. Golovnev, J. Lee, S. Setty, J. Thaler, and R. S. Wahby, "Brakedown: Linear-time and field-agnostic SNARKs for R1CS," in *Advances in Cryptology – CRYPTO 2023*, ser. Lecture Notes in Computer Science, vol. 14082. Springer, 2023, pp. 193–226.

[18] H. Zeilberger, B. Chen, and B. Fisch, "BaseFold: Efficient field-agnostic polynomial commitment schemes from foldable codes," in *Advances in Cryptology – CRYPTO 2024*, ser. Lecture Notes in Computer Science, vol. 14929. Springer, 2024.

[19] S.-J. Lin, W.-H. Chung, and Y. S. Han, "Novel polynomial basis and its application to Reed-Solomon erasure codes," in *IEEE 55th Annual Symposium on Foundations of Computer Science (FOCS)*, 2014, pp. 316–325.

[20] B. Chen, B. Bünz, D. Boneh, and Z. Zhang, "HyperPlonk: Plonk with linear-time prover and high-degree custom gates," in *Advances in Cryptology – EUROCRYPT 2023*, ser. Lecture Notes in Computer Science, vol. 14005. Springer, 2023, pp. 499–530.

[21] S. Setty, J. Thaler, and R. S. Wahby, "Unlocking the lookup singularity with Lasso," in *Advances in Cryptology – EUROCRYPT 2024*, ser. Lecture Notes in Computer Science, vol. 14656. Springer, 2024, pp. 180–209.

[22] A. Gabizon, Z. J. Williamson, and O. Ciobotaru, "PLONK: Permutations over Lagrange-bases for oecumenical noninteractive arguments of knowledge," Cryptology ePrint Archive, Paper 2019/953, 2019. [Online]. Available: https://eprint.iacr.org/2019/953

[23] C. Lund, L. Fortnow, H. J. Karloff, and N. Nisan, "Algebraic methods for interactive proof systems," *Journal of the ACM*, vol. 39, no. 4, pp. 859–868, 1992.

[24] D. Wiedemann, "An iterated quadratic extension of GF(2)," *The Fibonacci Quarterly*, vol. 26, no. 4, pp. 290–294, 1988.

[25] J. L. Fan and C. Paar, "On efficient inversion in tower fields of characteristic two," in *Proc. IEEE International Symposium on Information Theory (ISIT)*. IEEE, 1997, p. 20.

[26] Polygon Miden / 0xMiden, "Miden VM documentation," https://github.com/0xMiden/miden-vm, 2024.

[27] L. Grassi, D. Khovratovich, C. Rechberger, A. Roy, and M. Schofnegger, "Poseidon: A new hash function for zero-knowledge proof systems," in *30th USENIX Security Symposium (USENIX Security 21)*. USENIX Association, 2021, pp. 519–535. [Online]. Available: https://www.usenix.org/conference/usenixsecurity21/presentation/grassi

[28] L. Grassi, D. Khovratovich, and M. Schofnegger, "Poseidon2: A faster version of the Poseidon hash function," in *Progress in Cryptology – AFRICACRYPT 2023*, ser. Lecture Notes in Computer Science, vol. 14064. Springer, 2023, pp. 177–203.

[29] E. Ben-Sasson, D. Carmon, S. Kopparty, and D. Levit, "Elliptic curve fast Fourier transform (ECFFT) part I: Low-degree extension in time $O(n \log n)$ over all finite fields," in *Proc. 2023 ACM-SIAM Symposium on Discrete Algorithms (SODA)*. SIAM, 2023, pp. 700–737.

[30] D. Carmon, L. Goldberg, U. Haböck, L. Lerer, I. Lesokhin, S. Papini, and S. Samocha, "S-two whitepaper," Cryptology ePrint Archive, Paper 2026/532, 2026. [Online]. Available: https://eprint.iacr.org/2026/532

[31] L. Grassi, D. Khovratovich, R. Lüftenegger, C. Rechberger, M. Schofnegger, and R. Walch, "Monolith: Circuit-friendly hash functions with new nonlinear layers for fast and constant-time implementations," *IACR Transactions on Symmetric Cryptology*, vol. 2024, no. 3, pp. 44–83, 2024.

[32] C. Bouvier, L. Grassi, D. Khovratovich, K. Koschatko, C. Rechberger, F. Schmid, and M. Schofnegger, "Skyscraper: Fast hashing on big primes," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2025, no. 2, pp. 743–780, 2025.

[33] E. Ben-Sasson, L. Goldberg, S. Kopparty, and S. Saraf, "DEEP-FRI: Sampling outside the box improves soundness," in *11th Innovations in Theoretical Computer Science Conference (ITCS)*, 2020. [Online]. Available: https://arxiv.org/abs/1903.12243

[34] B. E. Diamond and J. Posen, "Proximity testing with logarithmic randomness," *IACR Communications in Cryptology*, vol. 1, no. 1, 2024.

[35] B. E. Diamond and A. Gruen, "Proximity gaps in interleaved codes," *IACR Communications in Cryptology*, vol. 1, no. 4, 2025.

[36] G. Arnon, A. Chiesa, G. Fenzi, and E. Yogev, "STIR: Reed-Solomon proximity testing with fewer queries," in *Advances in Cryptology – CRYPTO 2024*, ser. Lecture Notes in Computer Science, vol. 14929. Springer, 2024, pp. 380–413.

[37] ——, "WHIR: Reed-Solomon proximity testing with super-fast verification," in *Advances in Cryptology – EUROCRYPT 2025*, ser. Lecture Notes in Computer Science. Springer, 2025.

[38] Irreducible, "Binary tower fields are the future of verifiable computing," Irreducible blog, 2024. [Online]. Available: https://www.irreducible.com/posts/binary-tower-fields-are-the-future-of-verifiable-computing

[39] Irreducible (formerly Ulvetanna), "Binius: A hardware-optimized SNARK," Irreducible blog, 2023. [Online]. Available: https://www.irreducible.com/posts/binius-hardware-optimized-snark

[40] V. Buterin, "Binius: Highly efficient proofs over binary fields," vitalik.eth.limo, 2024. [Online]. Available: https://vitalik.eth.limo/general/2024/04/29/binius.html

[41] ——, "Exploring circle STARKs," vitalik.eth.limo, 2024. [Online]. Available: https://vitalik.eth.limo/general/2024/07/23/circlestarks.html

[42] Irreducible, "FRI-Binius: Improved polynomial commitments for binary towers," Irreducible blog, 2024. [Online]. Available: https://www.irreducible.com/posts/fri-binius

[43] Succinct Labs, "SP1: A performant, open-source zkVM," https://github.com/succinctlabs/sp1, 2024.

[44] RISC Zero, "RISC Zero: General-purpose zero-knowledge zkVM (RV32IM)," https://github.com/risc0/risc0, 2023.

[45] A. Szepieniec, T. Ashur, and S. Dhooghe, "Rescue-Prime: a standard specification (SoK)," Cryptology ePrint Archive, Paper 2020/1143, 2020. [Online]. Available: https://eprint.iacr.org/2020/1143

[46] A. Aly, T. Ashur, E. Ben-Sasson, S. Dhooghe, and A. Szepieniec, "Design of symmetric-key primitives for advanced cryptographic protocols," *IACR Transactions on Symmetric Cryptology*, vol. 2020, no. 3, pp. 1–45, 2020.