Proximity Testing with Logarithmic Randomness

Benjamin E. Diamond
Ulvetanna
bdiamond@ulvetanna.io

Jim Posen
Ulvetanna
jposen@ulvetanna.io

Abstract

A fundamental result dating to *Ligero* (CCS '17) establishes that each fixed linear block code exhibits *proximity gaps* with respect to the collection of affine subspaces, in the sense that each given subspace either resides entirely close to the code, or else contains only a small portion which resides close to the code. In particular, any given subspace's *failure* to reside entirely close to the code is necessarily witnessed, with high probability, by a uniformly randomly sampled element of that subspace. We investigate a variant of this phenomenon in which the witness is not sampled uniformly from the subspace, but rather from a much smaller subset of it. We show that a *logarithmic* number of random field elements (in the dimension of the subspace) suffice to effect an analogous proximity test, with moreover only a *logarithmic* (multiplicative) loss in the possible prevalence of false witnesses. We discuss applications to recent noninteractive proofs based on linear codes, including *Brakedown* and *Orion* (CRYPTO '22).

1 Introduction

Proximity testing of linear block codes is an important target of many reductions, for example throughout the literature on succinct noninteractive proofs. In the basic version of this problem, a *claimed* codeword is tested for proximity to some given fixed linear block code, by means of an interactive protocol (or more generally, an *interactive oracle proof*). The resulting protocol should *accept* genuine codewords with probability one; conversely, it should *reject* non-codewords with a probability closely related to the initial vector's distance from the code. It should also feature efficiency—say, measured in the number of oracle queries, or rounds of interaction—which grows favorably as a function of the code's block length (say, logarithmically).

In many applications, it is necessary to test whether a *list* of vectors consists entirely of words which are close to the code. This latter task is made precise as a proximity test for the code's *interleaved code*, defined as the set of matrices whose rows are all codewords, where the *distance* between two matrices is defined to be the number of columns at which the two matrices don't *entirely* agree. Indeed, proximity tests for interleaved linear codes reside at the heart of many recent zero-knowledge proof protocols, including *Ligero* [AHIV17], *Brakedown* [GLS+21], and *Orion* [XZS22].

Interleaved proximity tests are typically effected by random linear combinations. In this paradigm, the verifier uniformly randomly samples combination coefficients for the vectors in the list, requests the corresponding combination, and finally subjects the *combination* to a standard proximity test. In order for this reduction to be sound, it should hold that the subspace generated by the list feature related *maximal* and *average* distances from the code. More precisely, it should hold that the failure of the subspace's *farthest* element to be close to the code implies in turn that of the *vast majority* of the code's elements. This property is established for general linear codes by Ames, Hazay, Ishai and Venkitasubramaniam's *Ligero* [AHIV17]; in this setting, the notion of "closeness" is formalized by means of a certain *proximity parameter* (required to remain below a third of the code's distance).

A drawback of this approach stems from its communication and randomness complexity. Indeed, it requires that the verifier sample and send as many coefficients as there are elements in the list. Even in the random oracle model, this requirement can induce practical consequences, as it does, for example, in the setting of *proof composition*, in which the verifier's check must necessarily be encoded into a circuit. In fact, below, we explain how this issue impacts the zero-knowledge proof protocol *Orion* of Xie, Zhang and Song [XZS22, Fig. 4], and invalidates that protocol's stated polylogarithmic verifier complexity.

1.1 Our Contribution

We introduce a batching process for proximity tests of general linear codes—that is, a reduction from the interleaved code's proximity testing problem to the standard proximity testing problem—which consumes only logarithmically many random coefficients in the size m of the initial list of vectors. Moreover, at least for proximity parameters within a certain range (explained below), our procedure's error parameter—which, by definition, upper-bounds the probability with which the verifier selects a *proximal* element despite testing a *non-proximal* subspace—exceeds by only a logarithmic multiplicative factor the "base" error parameter applicable to the standard affine parameter test (this latter parameter is given by the proximity-gap result of [AHIV17, Lem. A.1]).

In the particular setting of Reed–Solomon codes, a result of Ben-Sasson, Carmon, Ishai, Kopparty and Saraf [BSCI+23, Thm. 1.5] achieves a proximity test with sublinear randomness; indeed, that result uses just a *single* random parameter to test an m-generator subspace for proximity. Its error parameter, however, exceeds by a multiplicative factor of $m-1$ that of the affine case. In the setting of *general* linear codes, Ben-Sasson, Kopparty, and Saraf [BSKS18, Thm. 12] describe a single-parameter proximity test, which, on the other hand, incurs exponential soundness loss in the list size m. In view of these previous results, our protocol achieves a favorable randomness–soundness tradeoff; it requires only logarithmically many parameters, and incurs only logarithmic multiplicative soundness loss. Our test is the first that we know of which achieves a practical soundness error bound, and consumes sublinear randomness, in the setting of general linear codes.

An interleaved test's *proximity parameter*, by definition, captures the *degree* of proximity the test detects. Our result works only for proximity parameters smaller than a third of the code's distance. We inherit this range restriction from the state-of-the-art for *standard* (i.e., linear-complexity) proximity testing. This state-of-the-art—whose proof, attributed to Roth and Zémor, appears in a recent update to Ligero [AHIV17, § A] (see also Theorem 2.1 below)—establishes proximity gaps for *affine lines* for proximity parameters smaller than a third of the code's distance. (Various strengthenings of this result in the Reed–Solomon setting have been attained by [BSCI+23].) We note that the analogue of this latter result for more general proximity parameters—say, smaller than half of the code's distance (i.e., up to its unique decoding radius)—remains an important open problem; our work would immediately profit from its hypothetical future resolution.

Our construction entails, roughly, that the verifier, given the initial list $u_0, \ldots, u_{m-1}$ of vectors, sample logarithmically many random scalars $r_0, \ldots, r_{\log m-1}$, send these to the prover, and finally request the combination of the vectors $u_0, \ldots, u_{m-1}$ whose coefficient vector is given by the *tensor product* (or *Kronecker product*) expansion $(1-r_0, r_0) \otimes \cdots \otimes (1-r_{\log m-1}, r_{\log m-1})$. The verifier then subjects this latter combination to a standard proximity test. As above, in order for this maneuver to be sound, it should hold that, for each initial list for which the subspace $\langle u_0, \ldots, u_{m-1} \rangle$ does *not* consist entirely of elements which are close to the code, *most* tuples $(r_0, \ldots, r_{\log m-1}) \in \mathbb{F}_q^{\log m}$ yield tensor-products whose corresponding combinations are themselves far from the code. This is exactly what we prove in our main result, given in Section 3.

Besides its attractive asymptotic profile and its simplicity, our construction is moreover strongly motivated by certain applications to polynomial commitment schemes, as we now sketch. Indeed, a certain approach to the problem of *multilinear polynomial commitment*—which appears to date to Ligero [AHIV17], and is explicitly isolated in the subsequent work *Brakedown* [GLS+21]—makes use of a suitable error-correcting code. This scheme, which, following Brakedown, we call the *Ligero multilinear polynomial commitment scheme*, proceeds by collating the coefficients of a given multilinear polynomial into the rows of a matrix, and then encoding this matrix row-wise (under the particular linear block code chosen for use). Crucially, if the resulting matrix is close to an *interleaved* codeword, then the committed polynomial is well-defined, and may be extracted. The Ligero scheme thus subjects the encoded matrix to an interleaved proximity test (the "testing" phase), before finally requesting its underlying polynomial's evaluation (the "evaluation" phase). The observation underlying our work is that if *our* batching procedure is used for the interleaved proximity test—and if the verifier's evaluation point is *random* (a minor condition which holds in all applications we're aware of)—then the testing and evaluation phases of the Ligero scheme become identical, and can be consolidated. The resulting gains in simplicity and efficiency are substantial. For example, we reduce the proof size of the Ligero scheme—*regardless* of the code used—by a factor of $\sqrt{2}$. In the special case that a linear-time-encodable code is used (as it is in Brakedown [GLS+21] and Orion [XZS22]), we moreover improve both the prover's and verifier's respective runtimes by a factor of 2, up to lower-order terms. We provide further details in Section 4 and thorough concrete benchmarks in Section 5.

We briefly sketch our proof (see also Theorem 3.1 below). Our proof makes blackbox use of the proximity gaps result for *affine lines* due to Roth and Zémor (see Theorem 2.1 below, in which we present a thorough, and somewhat simpler, proof of this result). Essentially, we observe that the tensor product exhibits a recursive substructure, whereby, when a l-variable tensor product is used as a combination vector, the resulting combination is *itself* an interpolation, over an affine line, of two $l-1$-variable tensor combinations. Under the hypothesis whereby many among the initial l-tensor combinations are close to the code, we manage to deduce that many $l-1$-tensor combinations yield *lines* which contain large close-to-the-code subsets. Applying Theorem 2.1 to *these* lines, we conclude that both $l-1$-combinations themselves frequently reside close to the code, thereby "pushing down" the initial hypothesis to two half-sized instances of the problem. Inducting, we reach the base case, which is once again simply Theorem 2.1. Finally, we show that two half-dimensional subspaces which individually exhibit correlated agreement may be "reconciled", so as to yield correlated agreement on their sum. We isolate a condition under which this reconciliation can be performed, whereby *both* $l-1$-tensors are *simultaneously* as *far* as is possible from the code (in that they disagree with the code everywhere outside of the correlated agreement set of the subspace in which they reside). The difficult part is to produce an appropriate such $l-1$-tensor (i.e., for which both combinations are far from the code). To achieve this, we bound the sizes of the "bad" sets within which the relevant $l-1$-tensors become spuriously close to the code; this in turn entails a union bound over the vanishing loci of $l-1$-variate polynomials, each bounded in size by the Schwartz–Zippel lemma. This latter technique can be viewed as a multivariate generalization of an idea which, in univariate form, appears throughout several prior works (see e.g. Roth and Zémor [AHIV17, § A] and Ben-Sasson, Kopparty, and Saraf [BSKS18, Lem. 8]). The idea whereby a *maximally far* element of a subspace can, in a sense, "force agreement" between words appears, implicitly, in a proof of Ben-Sasson, Carmon, Ishai, Kopparty and Saraf [BSCI$^+$23, § 6.3].

Section 4—in which, applying our new proximity test, we describe a certain improved scheme for multilinear polynomial commitment—also presents technical difficulties. The difficult part is to show that our polynomial commitment scheme features *witness-extended emulation.* We note that emulation is trivial for codes which admit efficient *decoders*, like the Reed–Solomon codes used by Ligero [AHIV17, § A]; we, however, treat general codes. When efficient decoding is not assumed, emulation becomes much more difficult (and requires rewinding). Indeed, our scheme imposes somewhat sophisticated demands on the emulator, which must collect a sequence of passing proximity tests with *linearly independent* combination vectors. We observe that a proof of a related result—in Brakedown [GLS$^+$21, § 4]—appears to suffer from a certain technical flaw, which invalidates that proof (see Remark 4.13 below, in which we moreover discuss a possible remediation for the flaw). We introduce a new emulation strategy, departing significantly from Brakedown's. Our emulator is actually quite simple, and is inspired by that of Bootle, Cerulli, Chaidos, Groth and Petit's classic *forking lemma* [BCC$^+$16, Lem. 1]. Its analysis, however, is challenging, and introduces a handful of new ideas. The main technical issue is that a malicious prover could, in principle, act in such a way as to thwart the emulator, by, say, outputting successful proofs with vastly higher probability when the verifier's challenge vector yields a tensor belonging to some proper subspace. In particular, its *conditional* distribution of proofs—that is, these proofs' distribution, conditioned on success—may depart radically from uniform, and may tend towards certain events which cause the emulator to fail. Our idea is to show that if the prover's success probability is sufficiently high—specifically, higher than the *square root* of that of the failure events, a quantity which, though likewise negligible, decays much more slowly—then this conditional distribution necessarily concentrates away from the failure events. The idea is to "split the difference" in the exponent (the square root operation has precisely the effect of *halving* the superlogarithmic decay function implicit in the failure probability's exponent). This square root is overwhelmingly higher than the failure probability itself; on the other hand, it's still negligible. This idea is key to our proof, and seems in some sense to be essential; in Remark 4.14, we discuss how to use it to fix a separate issue in Brakedown's proof.

In Section 5 below, we describe how our technique *concretely* improves the performance of certain polynomial commitment schemes. Focusing on the Ligero-style scheme discussed above, we exhibit a $\sqrt{2}$-factor improvement to that protocol's proof size, up to lower-order terms, for each input polynomial size; we also improve the protocol's prover and verifier time by at least twofold, up to lower-order terms, for each input polynomial size.

1.2 Prior Work

Ideas related to ours appear throughout several prior works. Brakedown [GLS$^+$21] identifies *Ligero* [AHIV17] as the progenitor of many of its ideas, though the latter work treats only *arithmetic circuits*, and doesn't present a polynomial commitment scheme *per se*.

On the other hand, Ligero appears to have initiated the study of proximity gaps; we use extensively that work's proximity gap result for affine spaces [AHIV17, § A]. In fact, the proof of that result resides in partial form across the two stated results [AHIV17, Lem. 4.3] and [AHIV17, Lem. A.1]. The former is due to Ames, Hazay, Ishai and Venkitasubramaniam, and appears in earlier versions of that work; the latter, on the other hand, is attributed by the authors to Roth and Zémor, and was added in a subsequent update. We observe that the former result—that is, [AHIV17, Lem. 4.3]—in fact already contains most of the techniques required to make the proof go through. Roth and Zémor's [AHIV17, Lem. A.1], on the other hand, introduces the idea of replacing both generators of the line with elements which are close to the code (though in an unnecessarily complicated form, in which the elements are assumed moreover to reside close to the origin). We synthesize and simplify these various ideas in our treatment below, given in Theorem 2.1.

A further conceptual predecessor to Brakedown [GLS$^+$21] appears in the form of Bootle, Cerulli, Chaidos, Groth and Petit [BCC$^+$16, § 3]; that work presents a *univariate* polynomial commitment scheme, which, nonetheless, arranges the polynomial's coefficients into a square matrix, and commits to their rows. That work doesn't use an error-correcting code *or* Merkle hashing, and admits square-root-sized—as opposed to constant-sized—commitments. Moreover, it doesn't invoke a proximity test at all, so that our topic is inapplicable to it.

The work Bootle, Chiesa and Groth [BCG20] bears some resemblance to ours, though differs fundamentally. That work presents a protocol for R1CS in the *tensor IOP* model, as well as a compiler from tensor IOPs to standard IOPs. The latter compiler invokes a proximity test for so-called *tensor codes*. In that protocol, over the course of multiple rounds, the prover repeatedly "folds" an initial tensor, using verifier-supplied randomness, and, in each round, sends the resulting intermediate tensor to the verifier. While the security proof of that protocol invokes the proximity-gaps result [AHIV17, § A], that result is applied "fold-wise" to the prover's successive intermediate tensors. That proof's structure thus differs importantly from ours; our prover performs $\log m$ folds "in one shot", sending only the final result, and our protocol is *constant-round*. In a sense, our proof of soundness must thus "do more work" than that protocol's, since it lacks access to the prover's intermediate folds. (The verifier of [BCG20, p. 30], by contrast, has access to these intermediate folds, and can check them "incrementally".)

Our polynomial commitment scheme, again, exploits the setting in which the verifier's point query is *random*. The insight whereby a polynomial commitment scheme suitable only for *random* points can be made more efficient than one suitable for *arbitrary* points appears to date to the work *Marlin* of Chiesa, Hu, Maller, Mishra, Vesely, and Ward [CHM$^+$20, § 6], though that work treats univariate polynomials. The work *Vortex* of Belling and Soleimanian [BS22] also makes this observation, in, moreover, the setting of a commitment scheme involving a proximity test, though that work's polynomials are again univariate. Indeed, Vortex observes that—in the random setting—its scheme's *testing* and *evaluation* phases can be merged. Puzzlingly, Vortex [BS22, § 6.2] cites *Brakedown* for this observation; beyond the fact that Brakedown's polynomials are multilinear, we have moreover failed to find a remark to this effect in Brakedown. Separately, Vortex claims that the soundness of their merged test is proven by [BCG20]. As noted above, the result proved by [BCG20, p. 30] is incomparable to that required to merge Brakedown's—or Vortex's—testing and evaluation phases. Rather, in the *univariate* setting, the soundness of this merge is, at least in the special case of Reed–Solomon codes (which Vortex uses), established in fact by [BSCI$^+$23, Thm. 1.5]. In the multilinear setting, the soundness of the merged procedure is precisely what we prove in this paper.

Brakedown [GLS$^+$21] serves as the most direct inspiration for this work. That work isolates the "Ligero-style" multilinear polynomial commitment scheme, citing both [AHIV17] and [BCG20] (though we emphasize that this technique appears at best implicitly in the latter works). We also take a degree of inspiration from the proof strategy of [GLS$^+$21, § 4], though we correct various issues in that proof, as noted above.

Acknowledgements. We would like to thank Benedikt Bünz for suggesting to us that the Ligero scheme's testing and evaluation phases could be combined, given a soundness proof for tensor product-based batching. We would like to thank Pratyush Mishra for explaining certain aspects of Marlin.

2 Background and Notation

We generally adopt the notation of [AHIV17] and [BSCI+23] in what follows. A *code* of length n over an alphabet Σ is a subset of Σ^n. We write q for a prime power, $\mathbb{F}_q$ for the finite field of order q, and $V \subset \mathbb{F}_q^n$ for a *linear* $[n, k, d]$-code over $\mathbb{F}_q$. We write $w(u)$ for the Hamming weight of a vector, $d(u, w)$ for the Hamming distance between two points, and $d(u, V) := \min_{v \in V} d(u, v)$ for the distance from a point to a code. We write $B(u, e) := \left\{ y \in \mathbb{F}_q^n \;\middle|\; d(u, y) \leq e \right\}$ for the *Hamming ball* of radius $e \geq 0$ centered at $u \in \mathbb{F}_q^n$. The unique decoding radius of an $[n, k, d]$-code V is $\left\lfloor \frac{d-1}{2} \right\rfloor$; in particular, for each $u \in \mathbb{F}_q^n$, we have that $\left| B\left(u, \left\lfloor \frac{d-1}{2} \right\rfloor\right) \cap V \right| \leq 1$. We finally write $\Delta(u, v) \subset \{0, \ldots, n-1\}$ for the *disagreement set* between u and a codeword v.

Given a linear code $V \subset \mathbb{F}_q^n$ and an integer $m \geq 1$, we have its corresponding *m-fold interleaved code*, defined as the subset $V^m \subset \left(\mathbb{F}_q^n\right)^m \cong \left(\mathbb{F}_q^m\right)^n$. We understand this latter set as a length-n block code over the alphabet $\mathbb{F}_q^m$. In particular, its elements are naturally identified with matrices in $\mathbb{F}_q^{m \times n}$, where two such matrices *differ* at a column if they differ at *any* of that column's components. We write matrices $(u_i)_{i=0}^{m-1} \in \mathbb{F}_q^{m \times n}$ row-wise. That a matrix $(u_i)_{i=0}^{m-1} \in \mathbb{F}_q^{m \times n}$ is within distance e to the code V^m—in which event we write $d^m\left((u_i)_{i=0}^{m-1}, V^m\right) \leq e$—entails precisely that there exists a subset $D := \Delta^m\left((u_i)_{i=0}^{m-1}, V^m\right)$, say, of $\{0, \ldots, n-1\}$, of size at most e, for which, for each $i \in \{0, \ldots, m-1\}$, the row u_i admits a codeword $v_i \in V$ for which $u_i|_{\{0,\ldots,n-1\} \setminus D} = v_i|_{\{0,\ldots,n-1\} \setminus D}$. We emphasize that the subset $D \subset \{0, \ldots, n-1\}$ is *fixed*, and does not vary as the row-index $i \in \{0, \ldots, m-1\}$ varies. In this circumstance, following the terminology of [BSCI+23], we say that the vectors $(u_i)_{i=0}^{m-1}$ feature *correlated agreement* outside of the set D, or that they feature *e-correlated agreement*. We note that the condition whereby the vectors $(u_i)_{i=0}^{m-1}$ feature e-correlated agreement with V^m implies *a fortiori* that every element in $(u_i)_{i=0}^{m-1}$'s row-span is itself within distance at most e from V.

We define the *tensor product* of vectors inductively on length-two vectors of the form $(1-r, r)$, where $r \in \mathbb{F}_q$; that is, we stipulate that $\left(s_0, \ldots, s_{m/2-1}\right) \otimes (1-r, r) := (1-r) \cdot \left(s_0, \ldots, s_{m/2-1}\right) \parallel r \cdot \left(s_0, \ldots, s_{m/2-1}\right)$. In particular, we thereby give meaning to iterated expressions of the form $(1-r_0, r_0) \otimes \cdots \otimes (1-r_{l-1}, r_{l-1})$ by left-association (the natural extension of this definition to operands of arbitrary length *is* associative). We note that the tensor product operation is *not* commutative. For notational purposes, we use the abbreviation $\bigotimes_{i=0}^{l-1}(1-r_i, r_i)$ to refer to the above expression (where the left-to-right order is again understood), which is a length-2^l vector. We note that this vector in fact consists precisely of the evaluations at the fixed point $(r_0, \ldots, r_{l-1}) \in \mathbb{F}_q^l$ of the 2^l *Lagrange basis polynomials* in $\mathbb{F}_q[X_0, \ldots, X_{l-1}]$, taken with respect to the evaluation set $\{0, 1\}^l \subset \mathbb{F}_q^l$. Indeed, this latter basis is precisely the list of polynomials $\bigotimes_{i=0}^{l-1}(1-X_i, X_i)$. We note that these polynomials are $\mathbb{F}_q$-linearly independent elements of $\mathbb{F}_q[X_0, \ldots, X_{l-1}]$, where we view the latter ring as an $\mathbb{F}_q$-vector space.

The probability distributions we consider are exclusively uniform over sets of the form $\mathbb{F}_q^l$. We write $\mu(R)$ for the probability mass of some given subset $R \subset \mathbb{F}_q^l$; clearly, $\mu(R) = \frac{|R|}{q^l}$.

Two distribution ensembles $\{\mathcal{Y}_0(a, \lambda)\}_{a \in \{0,1\}^*, \lambda \in \mathbb{N}}$ and $\{\mathcal{Y}_1(a, \lambda)\}_{a \in \{0,1\}^*, \lambda \in \mathbb{N}}$, with values in $\{0, 1\}$, say, are *statistically close* if there exists a negligible function $\mathsf{negl}(\lambda)$ for which, for each $a \in \{0, 1\}^*$ and each $\lambda \in \mathbb{N}$,

$$|\Pr[\mathcal{Y}_0(a, \lambda) = 1] - \Pr[\mathcal{Y}_1(a, \lambda) = 1]| \leq \mathsf{negl}(\lambda).$$

We note that the negligible function negl must suffice simultaneously for *each possible* $a \in \{0, 1\}^*$. For details on indistinguishability and further context, we refer to Lindell [Lin17].

We now recapitulate a key result due to Ames, Hazay, Ishai and Venkitasubramaniam [AHIV17, Lem. 4.3] and to Roth and Zémor [AHIV17, § A]. For completeness, we include a thorough proof. We closely follow [AHIV17, Lem. 4.3] and [AHIV17, § A], though we manage to significantly simplify those proofs.

Theorem 2.1 (Roth–Zémor [AHIV17, § A]). *Fix an arbitrary $[n, k, d]$-code $V \subset \mathbb{F}_q^n$, and a proximity parameter $e \in \left\{0, \ldots, \left\lfloor \frac{d-1}{3} \right\rfloor\right\}$. If given elements u_0 and u_1 of $\mathbb{F}_q^n$ satisfy*

$$\Pr_{r \in \mathbb{F}_q}\left[d((1-r) \cdot u_0 + r \cdot u_1, V) \leq e\right] > \frac{e+1}{q},$$

then $d^2\left((u_i)_{i=0}^{1}, V^2\right) \leq e$.

Proof. We write $R^* := \{r \in \mathbb{F}_q \mid d((1-r)\cdot u_0 + r \cdot u_1, V) \le e\}$. The theorem's hypothesis clearly implies that $|R^*| > e+1 \ge 1$. We thus write r_0^* and r_1^* for two distinct elements of R^*. Clearly, the elements $(1-r_0^*)\cdot u_0 + r_0^* \cdot u_1$ and $(1-r_1^*)\cdot u_0 + r_1^* \cdot u_1$ span the same affine line u_0 and u_1 do. It thus suffices to prove the theorem after performing the replacements $u_0 := (1-r_0^*)\cdot u_0 + r_0^* \cdot u_1$ and $u_1 := (1-r_1^*)\cdot u_0 + r_1^* \cdot u_1$. In particular, we may safely assume without loss of generality that $d(u_0, V) \le e$ and $d(u_1, V) \le e$ hold. We write v_0 and v_1 for codewords for which $d(u_0, v_0) \le e$ and $d(u_1, v_1) \le e$, respectively, hold.

We suppose for contradiction that the conclusion of the theorem is false, and that $d^2\left((u_i)_{i=0}^1, V^2\right) > e$. We note that $(u_i)_{i=0}^1$ clearly has correlated agreement outside of $\Delta(u_0, v_0) \cup \Delta(u_1, v_1)$; our assumption thus implies in particular that $|\Delta(u_0, v_0) \cup \Delta(u_1, v_1)| > e$. For each $j \in \Delta(u_0, v_0) \cup \Delta(u_1, v_1)$, we write

$$C_j := \left\{ r \in \mathbb{F}_q \;\middle|\; ((1-r)\cdot u_0 + r \cdot u_1)|_{\{j\}} = ((1-r)\cdot v_0 + r \cdot v_1)|_{\{j\}} \right\}$$

for the set of parameters $r \in \mathbb{F}_q$ at which $(1-r)\cdot u_0 + r \cdot u_1$ and $(1-r)\cdot v_0 + r \cdot v_1$ "spuriously agree" at the index j. For each $j \in \Delta(u_0, v_0) \cup \Delta(u_1, v_1)$, $|C_j| \le 1$, as $C_j \subset \mathbb{F}_q$ is the zero locus of a nonzero affine-linear function. By the guarantees $|\Delta(u_0, v_0)| \le e$ and $|\Delta(u_1, v_1)| \le e$, and because $|\Delta(u_0, v_0) \cup \Delta(u_1, v_1)| > e$ (by the above argument), applying the identity $|\Delta(u_0, v_0) \cup \Delta(u_1, v_1)| = |\Delta(u_1, v_1)| + |\Delta(u_1, v_1)| - |\Delta(u_0, v_0) \cap \Delta(u_1, v_1)|$, we conclude that $|\Delta(u_0, v_0) \cap \Delta(u_1, v_1)| < e$. We finally observe that for each j outside of this intersection, the set C_j is *either* $\{0\}$ or $\{1\}$. Specifically, for $j \in \Delta(u_0, v_0) \setminus \Delta(u_1, v_1)$, $C_j = \{1\}$, while for $j \in \Delta(u_1, v_1) \setminus \Delta(u_0, v_0)$, $C_j = \{0\}$. It thus follows that $\left|\bigcup_{j \in \Delta(u_0, v_0) \cup \Delta(u_1, v_1)} C_j\right| \le e+1$.

We fix an element $r^* \in R^*$, and write $v^* \in V$, say, for the (uniquely determined) codeword for which $d((1-r^*)\cdot u_0 + r^* \cdot u_1, v^*) \le e$. We note that $d((1-r^*)\cdot u_0 + r^* \cdot u_1, (1-r^*)\cdot v_0 + r^* \cdot v_1) \le 2 \cdot e$, since these two vectors agree outside of $\Delta(u_0, v_0) \cup \Delta(u_1, v_1)$. By the triangle inequality, we thus have that:

$$\begin{aligned} d((1-r^*)\cdot v_0 + r^* \cdot v_1, v^*) &\le d((1-r^*)\cdot u_0 + r^* \cdot u_1, v^*) + d((1-r^*)\cdot u_0 + r^* \cdot u_1, (1-r^*)\cdot v_0 + r^* \cdot v_1) \\ &\le 3 \cdot e \\ &< d, \end{aligned}$$

so that $v^* = (1-r^*)\cdot v_0 + r^* \cdot v_1$. We conclude that $d((1-r^*)\cdot u_0 + r^* \cdot u_1, (1-r^*)\cdot v_0 + r^* \cdot v_1) \le e$ in fact holds, and hence that $r^* \in \bigcup_{j \in \Delta(u_0, v_0) \cup \Delta(u_1, v_1)} C_j$. It follows in turn that $R^* \subset \bigcup_{j \in \Delta(u_0, v_0) \cup \Delta(u_1, v_1)} C_j$, so that $|R^*| \le e+1$ and $\Pr_{r \in \mathbb{F}_q}[d((1-r)\cdot u_0 + r \cdot u_1, V) \le e] \le \frac{e+1}{q}$, and the theorem's hypothesis is false. This completes the proof. □

Remark 2.2. A result exactly analogous to Theorem 2.1—with identical parameters—holds for *arbitrary-dimensional* affine subspaces, and can moreover be proven using Theorem 2.1. In fact, precisely this reduction is carried out in [BSCI+23, § 6.3] (in the list-decoding setting no less, though that work's approach is straightforwardly specialized). Since we don't need this more general result below, we omit its treatment.

3 Main Result

We now describe our new interleaved-to-standard reduction for proximity testing. We assume in what follows that the list length m is a power of 2. In our test, we use the tensor product expression $(1-r_0, r_0) \otimes \cdots \otimes (1-r_{\log m-1}, r_{\log m-1})$, where the elements $r_0, \ldots, r_{\log m-1}$ of $\mathbb{F}_q$ are independently random, as a combination vector over the input list; our error parameter increases over that of the affine case by a multiplicative factor of less than $2 \cdot \log m$. We make blackbox use of Theorem 2.1 throughout. In order to state a slightly simpler bound, we deliberately exclude the case $e = 0$. Specifically, we have the following result:

Theorem 3.1. *Fix an arbitrary $[n, k, d]$-code $V \subset \mathbb{F}_q^n$, and a proximity parameter $e \in \left\{1, \ldots, \left\lfloor \frac{d-1}{3} \right\rfloor\right\}$. If given elements $u_0, \ldots, u_{m-1}$ of $\mathbb{F}_q^n$ satisfy*

$$\Pr_{(r_0, \ldots, r_{\log m-1}) \in \mathbb{F}_q}\left[d\left(\begin{bmatrix} \bigotimes_{i=0}^{\log m-1}(1-r_i, r_i) \end{bmatrix} \cdot \begin{bmatrix} — & u_0 & — \\ & \cdots & \\ — & u_{m-1} & — \end{bmatrix}, V \right) \le e \right] > 2 \cdot \log m \cdot \frac{e}{q},$$

then $d^m\left((u_i)_{i=0}^{m-1}, V^m\right) \le e$.

Proof. We write $l := \log m$ once and for all. In the base case $l = 1$, the result follows immediately from Theorem 2.1, since $2 \cdot \frac{e}{q} \geq \frac{e+1}{q}$ so long as $e > 0$.

We now let $l > 1$ be arbitrary, and assume the hypothesis of the theorem. By way of induction, we construct two smaller instances of the problem, each of size $l - 1$, and establish the theorem's hypothesis on these instances. We prepare the process by introducing notation. For each tuple $(r_0, \ldots, r_{l-2}) \in \mathbb{F}_q^{l-1}$, we record the following abbreviations, each involving an appropriate half-list of the initial list $(u_i)_{i=0}^{2^l-1}$:

$$M_0 := \begin{bmatrix} & \bigotimes_{i=0}^{l-2}(1 - r_i, r_i) & \end{bmatrix} \cdot \begin{bmatrix} — & u_0 & — \\ & \cdots & \\ — & u_{2^{l-1}-1} & — \end{bmatrix}, \quad M_1 := \begin{bmatrix} & \bigotimes_{i=0}^{l-2}(1 - r_i, r_i) & \end{bmatrix} \cdot \begin{bmatrix} — & u_{2^{l-1}} & — \\ & \cdots & \\ — & u_{2^l-1} & — \end{bmatrix}.$$

We emphasize that each pair $M_0 := M_0(r_0, \ldots, r_{l-2})$ and $M_1 := M_1(r_0, \ldots, r_{l-2})$ actually depends on a fixed choice of tuple $(r_0, \ldots, r_{l-2}) \in \mathbb{F}_q^{l-1}$; we slightly abuse notation by omitting this tuple. Our recursive substructure relies on the following identity, valid for each $(r_0, \ldots, r_{l-1}) \in \mathbb{F}_q^l$:

$$\begin{bmatrix} & \bigotimes_{i=0}^{l-1}(1 - r_i, r_i) & \end{bmatrix} \cdot \begin{bmatrix} — & u_0 & — \\ & \cdots & \\ — & u_{2^l-1} & — \end{bmatrix} = \begin{bmatrix} 1 - r_{l-1} & r_{l-1} \end{bmatrix} \cdot \begin{bmatrix} — & M_0 & — \\ — & M_1 & — \end{bmatrix}.$$

This identity follows directly from the definition of the tensor product, and is easily verified by means of an explicit calculation. We finally define various loci in $\mathbb{F}_q^{l-1}$. We write $R_0 := \left\{(r_0, \ldots, r_{l-2}) \in \mathbb{F}_q^{l-1} \mid d(M_0, V) \leq e\right\}$ and $R_1 := \left\{(r_0, \ldots, r_{l-2}) \in \mathbb{F}_q^{l-1} \mid d(M_1, V) \leq e\right\}$ for the loci consisting of those $l - 1$-tuples for which M_0 and M_1 (respectively) are at most e-far from the code. We finally introduce the expression

$$p(r_0, \ldots, r_{l-2}) := \Pr_{r_{l-1} \in \mathbb{F}_q}\left[d\left(\begin{bmatrix} 1 - r_{l-1} & r_{l-1} \end{bmatrix} \cdot \begin{bmatrix} — & M_0 & — \\ — & M_1 & — \end{bmatrix}, V\right) \leq e\right],$$

defined for each $(r_0, \ldots, r_{l-2}) \in \mathbb{F}_q^{l-1}$, and set $R^* := \left\{(r_0, \ldots, r_{l-2}) \in \mathbb{F}_q^{l-1} \,\middle|\, p(r_0, \ldots, r_{l-2}) > \frac{e+1}{q}\right\}$. In words, $R^* \subset \mathbb{F}_q^{l-1}$ is that locus consisting of those $l - 1$-tuples whose resulting combinations M_0 and M_1 (with the initial matrix's lower and upper halves, respectively) interpolate to form a *line* a substantial proportion of whose elements reside close to the code.

We begin with the following lemma:

Lemma 3.2. $R^* \subset R_0 \cap R_1$.

Proof. Indeed, that $(r_0, \ldots, r_{l-2}) \in R^*$ holds entails, by definition, that the hypothesis of Theorem 2.1 holds with respect to the affine line spanned by the relevant combinations M_0 and M_1. That theorem implies that $d(M_0, V) \leq e$ (so that $(r_0, \ldots, r_{l-2}) \in R_0$) and $d(M_1, V) \leq e$ (so that $(r_0, \ldots, r_{l-2}) \in R_1$). □

The following lemma shows that the subset $R^* \subset \mathbb{F}_q^{l-1}$ is necessarily large:

Lemma 3.3. $\mu(R^*) > 2 \cdot (l - 1) \cdot \frac{e}{q}$.

Proof. The result follows from a probability decomposition argument, as we explain below:

$$\begin{aligned} 2 \cdot l \cdot \frac{e}{q} &< \Pr_{(r_0, \ldots, r_{l-1}) \in \mathbb{F}_q^l}\left[d\left(\begin{bmatrix} & \bigotimes_{i=0}^{l-1}(1 - r_i, r_i) & \end{bmatrix} \cdot \begin{bmatrix} — & u_0 & — \\ & \cdots & \\ — & u_{2^l-1} & — \end{bmatrix}, V\right) \leq e\right] && \text{(by hypothesis.)} \\ &= \Pr_{(r_0, \ldots, r_{l-1}) \in \mathbb{F}_q^l}\left[d\left(\begin{bmatrix} 1 - r_{l-1} & r_{l-1} \end{bmatrix} \cdot \begin{bmatrix} — & M_0 & — \\ — & M_1 & — \end{bmatrix}, V\right) \leq e\right] && \text{(recursive substructure.)} \\ &\leq \frac{e+1}{q} + \Pr_{(r_0, \ldots, r_{l-2}) \in \mathbb{F}_q^{l-1}}[(r_0, \ldots, r_{l-2}) \in R^*]. \end{aligned}$$

Assuming the validity of the final step, we see that $\mu(R^*) \geq 2 \cdot l \cdot \frac{e}{q} - \frac{e+1}{q} = (2 \cdot l - 1) \cdot \frac{e}{q} - \frac{1}{q} \geq 2 \cdot (l-1) \cdot \frac{e}{q}$, as desired. We presently justify the final step. Upper-bounding the second-to-last expression slice-wise—either by $\frac{e+1}{q}$ or by 1, depending on whether the slice $(r_0, \ldots, r_{l-2}) \overset{?}{\in} R^*$—we establish the bound $\frac{e+1}{q} \cdot \Pr_{(r_0,\ldots,r_{l-2}) \in \mathbb{F}_q^{l-1}}\left[p(r_0, \ldots, r_{l-2}) \leq \frac{e+1}{q}\right] + \Pr_{(r_0,\ldots,r_{l-2}) \in \mathbb{F}_q^{l-1}}\left[p(r_0, \ldots, r_{l-2}) > \frac{e+1}{q}\right] \leq \frac{e+1}{q} + \Pr_{(r_0,\ldots,r_{l-2}) \in \mathbb{F}_q^{l-1}}[(r_0, \ldots, r_{l-2}) \in R^*]$. This completes the proof of the lemma. □

Upon combining Lemmas 3.3 and 3.2, we immediately conclude that the probabilities $\mu(R_0)$ and $\mu(R_1)$ are both themselves greater than $2 \cdot (l-1) \cdot \frac{e}{q}$. In other words, the hypothesis of the theorem is fulfilled with respect to the parameter $l-1$ and to both of the half-sublists $(u_i)_{i=0}^{2^{l-1}-1}$ and $(u_i)_{i=2^{l-1}}^{2^l-1}$. This justifies our inductive use of the theorem with respect to these half-sublists.

We thus conclude the consequence of the theorem with respect to the submatrices $(u_i)_{i=0}^{2^{l-1}-1}$ and $(u_i)_{i=2^{l-1}}^{2^l-1}$. We write $e_0 := d^{2^{l-1}}\left((u_i)_{i=0}^{2^{l-1}-1}, V^{2^{l-1}}\right)$ and $e_1 := d^{2^{l-1}}\left((u_i)_{i=2^{l-1}}^{2^l-1}, V^{2^{l-1}}\right)$ for these submatrices' interleaved distances, as well as D_0 and D_1 for their corresponding (correlated) disagreement subsets of $\{0, \ldots, n-1\}$. We finally write $(v_i)_{i=0}^{2^{l-1}-1}$ and $(v_i)_{i=2^{l-1}}^{2^l-1}$ for the corresponding lists of close codewords. By analogy with M_0 and M_1, we now record, for *each* $(r_0, \ldots, r_{l-2}) \in \mathbb{F}_q^{l-1}$, the following abbreviations:

$$N_0 := \begin{bmatrix} \bigotimes_{i=0}^{l-2}(1-r_i, r_i) \end{bmatrix} \cdot \begin{bmatrix} — & v_0 & — \\ & \cdots & \\ — & v_{2^{l-1}-1} & — \end{bmatrix}, \quad N_1 := \begin{bmatrix} \bigotimes_{i=0}^{l-2}(1-r_i, r_i) \end{bmatrix} \cdot \begin{bmatrix} — & v_{2^{l-1}} & — \\ & \cdots & \\ — & v_{2^l-1} & — \end{bmatrix}.$$

We define further loci in $\mathbb{F}_q^{l-1}$:

$$B_0 := \left\{(r_0, \ldots, r_{l-2}) \in \mathbb{F}_q^{l-1} \mid d(M_0, N_0) < e_0\right\}, \quad B_1 := \left\{(r_0, \ldots, r_{l-2}) \in \mathbb{F}_q^{l-1} \mid d(M_1, N_1) < e_1\right\}.$$

We understand these loci as the subsets of the parameter space at which M_0 and M_1 (respectively) become *closer* to N_0 and N_1 than correlated agreement demands.

The following lemma shows that the loci B_0 and B_1 are not too large:

Lemma 3.4. *$\mu(B_0) \leq (l-1) \cdot \frac{e}{q}$ and $\mu(B_1) \leq (l-1) \cdot \frac{e}{q}$.*

Proof. We let $b \in \{0, 1\}$ be arbitrary, and prove the result for B_b. For each index $j \in D_b$, we write

$$C_{b,j} := \left\{(r_0, \ldots, r_{l-2}) \in \mathbb{F}_q^{l-1} \;\middle|\; M_b|_{\{j\}} = N_b|_{\{j\}}\right\}$$

for the locus in $\mathbb{F}_q^{l-1}$ on which M_b and N_b "spuriously agree" at the index j. We note that each $C_{b,j} \subset \mathbb{F}_q^{l-1}$ is precisely the vanishing locus of a certain combination of the $l-1$-variate multilinear Lagrange basis polynomials, where the combination vector—because $j \in D_b$—is *not* identically zero. We conclude that the combination is itself nonzero; the Schwartz–Zippel lemma thus implies that $\mu(C_{b,j}) \leq \frac{l-1}{q}$. These sets' union thus has mass at most $m\left(\bigcup_{j \in D_b} C_{b,j}\right) \leq |D_b| \cdot \frac{l-1}{q} \leq (l-1) \cdot \frac{e}{q}$, where, in the last step, we exploit the inductive hypothesis $|D_b| = e_b \leq e$. On the other hand, $B_b = \bigcup_{j \in D_b} C_{b,j}$. This completes the proof. □

For the following lemma, we recall the set $R^* \subset \mathbb{F}_q^{l-1}$ introduced above.

Lemma 3.5. *$R^* \not\subset B_0 \cup B_1$.*

Proof. Indeed, by Lemma 3.3, $\mu(R^*) > 2 \cdot (l-1) \cdot \frac{e}{q}$; on the other hand, Lemma 3.4 gives that the masses $\mu(B_0)$ and $\mu(B_1)$ are each at most $(l-1) \cdot \frac{e}{q}$. □

By Lemma 3.5, there necessarily exists some element $(r_0^*, \ldots, r_{l-2}^*) \in R^* \setminus (B_0 \cup B_1)$. We write M_0^* and M_1^* for the corresponding values of M_0 and M_1, and moreover define N_0^* and N_1^* analogously. Because $(r_0^*, \ldots, r_{l-2}^*) \in R^*$, an application of Theorem 2.1 to the line spanned by M_0^* and M_1^* yields a subset $D^* \subset \{0, \ldots, n-1\}$, satisfying $|D^*| = e^*$, say, where $e^* \leq e$, together with codewords O_0 and O_1 which respectively agree with M_0^* and M_1^* *outside of* D^*.

For each $b \in \{0,1\}$, because $(r_0^*, \ldots, r_{l-2}^*) \notin B_b$ moreover holds, we have in fact the disagreement set *equality* $\Delta(M_b^*, N_b^*) = D_b$ (as opposed to a proper inclusion). We write $\Delta(M_b^*, O_b)$ for the disagreement set of M_b^* and O_b. By definition of D^*, $\Delta(M_b^*, O_b) \subset D^*$ clearly holds; on the other hand, because $d(M_b^*, N_b^*) \leq e_b \leq e$ and $d(M_b^*, O_b) \leq e^* \leq e$ simultaneously hold, unique decoding implies that $N_b^* = O_b$, and that in fact $\Delta(M_b^*, O_b) = D_b$. We conclude that $D_b \subset D^*$.

It follows that $D_0 \cup D_1 \subset D^*$. We conclude that $(u_i)_{i=0}^{2^{l-1}-1}$ and $(u_i)_{i=2^{l-1}}^{2^l-1}$ have mutual correlated agreement outside of the set $D_0 \cup D_1$ of size at most $e^* \leq e$. This completes the proof of the theorem. □

Remark 3.6. For simplicity, Theorem 3.1 refrains from treating the case $e = 0$. An analogous result—albeit with the slightly weakened error bound $2 \cdot \log m \cdot \frac{e+1}{q}$—holds over the full range $e \in \left\{0, \ldots, \lfloor \frac{d-1}{3} \rfloor\right\}$.

Remark 3.7. In the Reed–Solomon setting, Ben-Sasson, Carmon, Ishai, Kopparty and Saraf [BSCI+23, Thm. 1.4] achieve proximity gaps for e up to the unique decoding radius, albeit with a bound $\frac{n}{q}$ on the false-witness probability which is somewhat worse than that of $\frac{e+1}{q}$ attained by our Theorem 2.1. (They also present results beyond the unique decoding radius, but for much more complicated bounds.) Their [BSCI+23, Cor. 1.3], on the other hand, treats a more restricted setting, in which the subspace is *assumed* to reside entirely close to the space; it states that all except a proportion consisting of at most $\frac{n}{q}$ among any such subspace's elements necessarily attain the subspace's *maximal* distance e^*. We note that the technique of our Lemma 3.4 above can in fact be used to establish a strengthening of [BSCI+23, Cor. 1.3], whereby this proportion is improved to $\frac{e^*}{q}$. This improvement does *not* strengthen or contradict the main result [BSCI+23, Thm. 1.2]. Indeed, the proof of [BSCI+23, Cor. 1.3] is essentially a naïve contraposition of [BSCI+23, Thm. I.2]; it does *not* leverage the correlated agreement which [BSCI+23, Thm. 1.4] stands to guarantee, or even the hypothesis whereby the entirety of the subspace is close to the code. These deficiencies appear to account for its relative weakness, and appear to represent an oversight on the part of [BSCI+23].

For completeness, we extract and state separately the strengthening of [BSCI+23, Cor. 1.3] discussed in Remark 3.7, which we moreover adapt to our setting. We omit its proof, as its key ideas appear above.

Theorem 3.8. *Fix an arbitrary $[n,k,d]$-code $V \subset \mathbb{F}_q^n$, and an affine-linear subspace $U \subset \mathbb{F}_q^n$. If the maximal distance $e^* := \max_{u \in U} d(u, V)$ satisfies $e^* \in \{0, \ldots, \lfloor \frac{d-1}{3} \rfloor\}$, then*

$$\Pr_{u \in U}[\Delta(u, V) < e^*] \leq \frac{e^*}{q}.$$

Theorem 3.8 admits a short proof, which, *first*, applies (an arbitrary-dimensional analogue of) Theorem 2.1 to U, to establish correlated agreement, and *then* proceeds with an argument akin to that of Lemma 3.4.

4 Polynomial Commitment

In this section, we note that our logarithmic batching procedure allows a certain well-known *polynomial commitment scheme* to be simplified and improved. Several recent constructions of succinct proofs—such as Brakedown [GLS+21], Orion [XZS22], and Vortex [BS22]—make use of a particular subprotocol for *multilinear polynomial commitment*, which we call the *Ligero multilinear polynomial commitment scheme.* The Ligero scheme—like polynomial commitment schemes in general—allows the prover to commit to a polynomial, and later, given an evaluation point supplied by the verifier, to evaluate the polynomial at the given point, and finally to produce a proof attesting to its evaluation's correctness.

We observe that—in all of the above protocols—the verifier evaluates each committed polynomial only a *random* point, as opposed to at an *arbitrary* point. This latter fact is itself explained by the *sum-check* reduction, as we briefly explain. That protocol reduces the problem of obtaining the *sum* of a multivariate polynomial's respective evaluations over the unit cube to the problem of evaluating the polynomial *once* at a single point in its domain, which, crucially, is *random* (sampled throughout the course of the sum-check protocol). We refer to [Set20, § 3] for further details. In other words, these succinct proof protocols employ a tool which is more powerful than necessary. Capitalizing on this observation, we isolate a special sort of multilinear polynomial commitment scheme, suitable only for *random* queries (see Definition 4.3). Our restricted notion serves as a drop-in replacement for the standard scheme in all of the above applications.

We moreover introduce a new commitment scheme—suitable only for random samplers—which significantly simplifies the Ligero protocol, as we now explain. We observe that, in the particular variant of the Ligero protocol which uses *our* batching procedure in lieu of the standard, linear-complexity proximity test (and where, once again, we assume that the verifier's evaluation point is random), the resulting "testing" and "evaluation" phases become identical, and can be consolidated. This measure yields gains in both simplicity and efficiency. Indeed, our approach reduces the Ligero scheme's proof size by a $\sqrt{2}$ factor, and also significantly improves the prover's and verifier's concrete computational costs in the random-evaluation setting. We explain our strategy in detail below.

This observation—i.e., whereby a polynomial commitment scheme suitable only for *random* evaluation points may be made more efficient than one suitable for *arbitrary* evaluation points—dates back to Chiesa, Hu, Maller, Mishra, Vesely, and Ward's *Marlin* [CHM$^+$20, § 6], in which the polynomial commitment schemes at hand are proven secure only for so-called "admissible query samplers" (we note, separately, that that work treats commitments to arbitrary-degree, and *univariate*, polynomials). In that setting, a *query sampler* is, by definition, an efficient algorithm which determines where the polynomials at hand are to be evaluated; a query sampler is said to be *admissible* if (roughly) it necessarily requests that each polynomial at hand be evaluated at least once on some point drawn uniformly from a superpolynomially-sized set (with additional queries also permissible). In Marlin's setting, a *general* query sampler may always be bootstrapped into an *admissible* one, by means of the concatenation of additional random queries; this transformation, however, imposes obvious efficiency costs. If the *desired* query sampler, on the other hand, is admissible to begin with—at is in their applications—then this extra work can be saved. In the language of Marlin, we prove, essentially, that our scheme is secure *provided* the query sampler is admissible (we treat, however, a restricted notion of "admissibility", for notational convenience, as we discuss below Definition 4.4). Put differently, we apply an insight already independently attained by Marlin to the setting of multilinear commitments.

The case of *Orion*. Beyond its efficiency advantages, our approach moreover resolves a more serious obstacle in the setting of *proof composition.* Indeed, in typical applications—which assume the random oracle model—the verifier need not send its combination coefficients *explicitly* to the prover, as both parties may generate them locally by the means of queries to the random oracle. The generation and transmission of these coefficients, in this setting, thus do *not* impact the protocol's verifier or communication complexity. In contrast, in the setting of proof composition—in which the verifier's check is necessarily encoded into a circuit—the random oracle introduces problems. For one, it must be instantiated concretely, so that it ceases to be a (true) random oracle. This fact may affect the security analysis of the inner protocol. Separately, hash function evaluations are expensive to encode in circuits. To evade these issues, many protocols extract the inner verifier's generation of the relevant random coefficients from the relevant circuit, and stipulate that the outer verifier instead populate them directly, as public inputs to the *outer* proof.

This latter strategy may impact the computational complexity of the outer verifier, particularly when, say, the inner verifier uses a proximity test in the style of [AHIV17, § A] or [BSCI$^+$23, Thm. 1.6] (with *linear* randomness complexity in the list size). For example, the *Orion* zero-knowledge proof protocol of Xie, Zhang and Song [XZS22, Prot. 4] proves satisfiability of a size-N arithmetic circuit by recursively invoking a *linear-randomness* batched proximity test on a list of $\Theta(\sqrt{N})$ vectors, and moreover delegates the randomness-generation required by this latter task to the outermost verifier. This strategy makes the computational complexity of its outermost verifier $\Omega(\sqrt{N})$, and invalidates its stated $O(\log^2 N)$ complexity.

Indeed, we explain this issue in detail, using the notation of Orion. In line 7 of [XZS22, Prot. 4], the *outer* prover "receives a random vector $\gamma_0 \in \mathbb{F}^k$ from the verifier"; here, $k = \Theta(\sqrt{N})$. While the outer prover may certainly use Fiat–Shamir to generate γ_0 locally (i.e., from the statement and transcript), this is beside the point, since the outer *verifier* must, in this setting, do the same, and supply γ_0 as a public input to the outer proof in line 18 of [XZS22, Prot. 4]. Alternatively, the outer verifier could demand that the outer prover *prove* that it correctly generated $\gamma_0 \in \mathbb{F}^k$ from the transcript during the course of its proof (i.e., that it applied Fiat–Shamir correctly). This, however, would prohibitively increase the outer prover's cost. Orion does not take this latter measure; its verifier's complexity is thus actually $\Omega(\sqrt{N})$, contrary to its claims.

The approach whereby *single-parameter* batching—i.e., using powers of a single random parameter—is instead used does not resolve the issue. Indeed, that approach would prohibitively increase the protocol's soundness error, by a factor *linear* in the list length $\Theta(\sqrt{N})$ in the Reed–Solomon case [BSCI$^+$23, Thm. 1.5], and—what is much worse—by an *exponential* factor in the case of general codes [BSKS18, Thm. 12].

Our technique stands to dissolve this issue on behalf of protocols, like Orion, which use proximity tests recursively, in that our test's logarithmic randomness complexity can easily be borne by an outer verifier. (The small, *logarithmic* soundness loss introduced by our test can be addressed by means of an increase in the number of trials, or—in case the field is very small—using the technique discussed in [GLS+21, p. 49].)

Our protocol. We now sketch slightly more thoroughly how our batching procedure allows the Ligero multilinear polynomial commitment scheme to be simplified (in the random-evaluation setting). We recall that the Ligero scheme begins by collating the m^2 coefficients of a given multilinear polynomial in $2 \cdot \log m$ variables (say), expressed moreover with respect to the Lagrange basis over the unit cube $\{0,1\}^{2 \cdot \log m}$, into the rows of an $m \times m$ matrix. This matrix is then encoded row-wise, using some fixed linear block code; the resulting matrix is finally committed to. (In Ligero [AHIV17], as well as in *Shockwave* [GLS+21], the Reed–Solomon code is used; in *Brakedown* [GLS+21, § 4.2], a newly introduced *linear-time-encodable* code is used instead.) Crucially, if the committed matrix is close to an *interleaved codeword*, then the committed polynomial is well-defined, and may be extracted.

The Ligero scheme thus proceeds in two phases. In the *testing* phase, the verifier applies an interleaved proximity test to the committed matrix. Specifically, the verifier reduces the interleaved proximity problem given by the initial matrix to a standard proximity testing problem, by means of a random combination of its rows. It then solves the latter by directly requesting the message underlying the combination (which is well-defined if the prover is honest), encoding the supplied message, and finally probabilistically testing it for equality with the combination by means of queries at random columns.

Having established the matrix's proximity to the interleaved code, the verifier initiates the *evaluation* phase, in which the committed polynomial is evaluated at the verifier's chosen point. In light of the of committed polynomial's assumed structure, this latter phase may be effected by means of a *further* combination of the committed rows (and a further proximity test), where—this time—the coefficient vector is a tensor. Indeed, it is straightforward to check that for any multilinear polynomial $t(X_0, \cdots, X_{2 \cdot \log m - 1}) \in \mathbb{F}_q[X_0, \cdots, X_{2 \cdot \log m - 1}]$ encoded in the above manner, and any point $(r_0, \ldots, r_{2 \cdot \log m - 1}) \in \mathbb{F}_q^{2 \cdot \log m}$, we have the equality:

$$t(r_0, \ldots, r_{2 \cdot \log m - 1}) = \begin{bmatrix} \bigotimes_{i=\log m}^{2 \cdot \log m - 1} (1 - r_i, r_i) \end{bmatrix} \cdot \begin{bmatrix} — & t_0 & — \\ & \cdots & \\ — & t_{m-1} & — \end{bmatrix} \cdot \begin{bmatrix} \bigotimes_{i=0}^{\log m - 1} (1 - r_i, r_i) \end{bmatrix}^T,$$

where the length-m rows $t_0, \ldots, t_{m-1}$ contain t's collated coefficients. It follows that if the verifier requests the message $t' := \bigotimes_{i=\log m}^{2 \cdot \log m - 1} (1 - r_i, r_i) \cdot (t_i)_{i=0}^{m-1}$ during the evaluation phase, then it may additionally calculate t's value at $(r_0, \ldots, r_{2 \cdot \log m - 1})$ by means of the further local calculation $t' \cdot \bigotimes_{i=0}^{\log m - 1} (1 - r_i, r_i) = t(r_0, \ldots, r_{2 \cdot \log m - 1})$. If the point $(r_0, \ldots, r_{2 \cdot \log m - 1}) \in \mathbb{F}_q^{2 \cdot \log m}$ is random, then this latter evaluation procedure becomes identical to our batched proximity test, and can supplant the testing phase altogether.

In the remainder of this section, we make our construction precise, and analyze the resulting gains. We begin by defining multilinear polynomial commitment schemes, closely following Setty [Set20, § 2.4].

Definition 4.1. A *multilinear polynomial commitment scheme* is a tuple of algorithms $\Pi = (\mathsf{Setup}, \mathsf{Commit}, \mathsf{Open}, \mathsf{Prove}, \mathsf{Verify})$, with the following syntax:

- $\mathsf{params} \leftarrow \Pi.\mathsf{Setup}(1^\lambda, l)$. On input the security parameter λ and a size parameter l, where $l = O(\log \lambda)$, $\Pi.\mathsf{Setup}$ samples params, which includes (possibly among other things) a finite field order $q = 2^{O(\lambda)}$.

- $(c, u) \leftarrow \Pi.\mathsf{Commit}(\mathsf{params}, t)$. On input a multilinear polynomial $t(X_0, \ldots, X_{2 \cdot l - 1}) \in \mathbb{F}_q[X_0, \ldots, X_{2 \cdot l - 1}]$, $\Pi.\mathsf{Commit}$ returns a commitment c to t, together with an *opening hint* u.

- $b \leftarrow \Pi.\mathsf{Open}(\mathsf{params}, c; t, u)$. On input a commitment c, a multilinear polynomial $t(X_0, \ldots, X_{2 \cdot l - 1}) \in \mathbb{F}_q[X_0, \ldots, X_{2 \cdot l - 1}]$, and an opening hint u, $\Pi.\mathsf{Open}$ verifies the claimed decommitment t of c, using u.

- $\pi \leftarrow \Pi.\mathsf{Prove}(\mathsf{params}, c, s, (r_0, \ldots, r_{2 \cdot l - 1}); t, u)$. On input a commitment c, a purported evaluation $s \in \mathbb{F}_q$, an evaluation point $(r_0, \ldots, r_{2 \cdot l - 1}) \in \mathbb{F}_q^{2 \cdot l}$, a multilinear polynomial $t(X_0, \ldots, X_{2 \cdot l - 1}) \in \mathbb{F}_q[X_0, \ldots, X_{2 \cdot l - 1}]$, and an opening hint u, $\Pi.\mathsf{Prove}$ generates an evaluation proof π.

- $b \leftarrow \Pi.\mathsf{Verify}(\mathsf{params}, c, s, (r_0, \ldots, r_{2 \cdot l - 1}), \pi)$. On input a commitment c, a purported evaluation s, an evaluation point $(r_0, \ldots, r_{2 \cdot l - 1}) \in \mathbb{F}_q^{2 \cdot l}$, and a proof π, $\Pi.\mathsf{Verify}$ outputs a success bit $b \in \{0, 1\}$.

The demand $l = O(\log \lambda)$ is necessary, lest the number of coefficients $m^2 = 2^{2 \cdot l}$ of each multilinear $t(X_0, \ldots, X_{2 \cdot l-1})$ be superpolynomial in λ. Similarly, the requirement $q = 2^{O(\lambda)}$ ensures that $\mathbb{F}_q$-elements are efficiently representable.

The scheme Π is *complete* if the obvious correctness properties hold. That is, for honestly generated $\mathsf{params} \leftarrow \Pi.\mathsf{Setup}(1^\lambda, l)$, each honestly generated commitment $(c, u) \leftarrow \Pi.\mathsf{Commit}(\mathsf{params}, t)$ to some multilinear polynomial $t(X_0, \ldots, X_{2 \cdot l-1}) \in \mathbb{F}_q[X_0, \ldots, X_{2 \cdot l-1}]$ should satisfy $\Pi.\mathsf{Open}(\mathsf{params}, c; t, u) = 1$; moreover, each honestly generated proof $\pi \leftarrow \Pi.\mathsf{Prove}(\mathsf{params}, c, s, (r_0, \ldots, r_{2 \cdot l-1}); t, u)$—for $(r_0, \ldots, r_{2 \cdot l-1}) \in \mathbb{F}_q^{2 \cdot l}$ given arbitrarily—should satisfy $\Pi.\mathsf{Verify}(\mathsf{params}, c, s, (r_0, \ldots, r_{2 \cdot l-1}), \pi) = 1$ with probability 1.

We say that Π is moreover *efficient* if the size of each commitment satisfies $|c| = O(\lambda)$, the size of each proof satisfies $|\pi| = o(\lambda \cdot m^2)$, the routines $\Pi.\mathsf{Commit}$, $\Pi.\mathsf{Open}$, and $\Pi.\mathsf{Prove}$ all run in time $\widetilde{O}(\lambda \cdot m^2)$, and $\Pi.\mathsf{Verify}$ runs in time $o(\lambda \cdot m^2)$.

We now give security definitions for multilinear polynomial commitment schemes. We first record the following definition of binding, which is essentially identical to that given in [Set20, Def. 2.11]:

Definition 4.2. For each multilinear polynomial commitment scheme Π, size parameter l, and PPT adversary $\mathcal{A}$, we define the *binding experiment* $\mathsf{Binding}_{\mathcal{A}}^{\Pi,l}(\lambda)$ as follows:

1. The experimenter samples $\mathsf{params} \leftarrow \Pi.\mathsf{Setup}(1^\lambda, l)$, and gives params to $\mathcal{A}$.
2. The adversary outputs $(c, t_0, t_1, u_0, u_1) \leftarrow \mathcal{A}(\mathsf{params})$, where c is a commitment, $t_0(X_0, \ldots, X_{2 \cdot l-1})$ and $t_1(X_0, \ldots, X_{2 \cdot l-1})$ are multilinear polynomials in $\mathbb{F}_q[X_0, \ldots, X_{2 \cdot l-1}]$, and u_0 and u_1 are opening hints.
3. The output of the experiment is defined to be 1 if $t_0 \neq t_1$, $\Pi.\mathsf{Open}(\mathsf{params}, c; t_0, s_0)$, and $\Pi.\mathsf{Open}(\mathsf{params}, c; t_1, s_1)$ all hold; otherwise, it is defined to be 0.

The multilinear polynomial commitment scheme Π is said to be *binding* if, for each PPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\lambda)$ for which, for each $\lambda \in \mathbb{N}$ and $l = O(\log \lambda)$, $\Pr\left[\mathsf{Binding}_{\mathcal{A}}^{\Pi,l}(\lambda)\right] \leq \mathsf{negl}(\lambda)$.

Finally, we record the following notion of *extractability* for multilinear polynomial commitment schemes. In what follows, we closely follow both Marlin [CHM$^+$20, Def. 6.2] and Setty [Set20, Def. 2.11].

Definition 4.3. For each multilinear polynomial commitment scheme Π, security parameter λ, size parameter l, PPT query sampler $\mathcal{Q}$, stateful PPT adversary $\mathcal{A}$, expected PPT emulator $\mathcal{E}$, and PPT distinguisher $\mathcal{D}$, we define two random variables $\mathsf{Real}_{\mathcal{Q},\mathcal{A},\mathcal{E},\mathcal{D}}^{\Pi,l}(\lambda)$ and $\mathsf{Emul}_{\mathcal{Q},\mathcal{A},\mathcal{E},\mathcal{D}}^{\Pi,l}(\lambda)$, each valued in $\{0, 1\}$, as follows:

1. The experimenter samples $\mathsf{params} \leftarrow \Pi.\mathsf{Setup}(1^\lambda, l)$, and gives params to $\mathcal{A}$, $\mathcal{Q}$ and $\mathcal{E}$.
2. The adversary outputs a commitment $c \leftarrow \mathcal{A}(\mathsf{params})$.
3. The query sampler outputs $(r_0, \ldots, r_{2 \cdot l-1}) \leftarrow \mathcal{Q}(\mathsf{params})$.
4. The experimenter proceeds in one of two separate ways:
   - $\mathsf{Real}_{\mathcal{Q},\mathcal{A},\mathcal{E},\mathcal{D}}^{\Pi,l}(\lambda)$: Run $(s, \pi) \leftarrow \mathcal{A}(r_0, \ldots, r_{2 \cdot l-1})$. Output the single bit $\mathcal{D}(c, s, \pi)$.
   - $\mathsf{Emul}_{\mathcal{Q},\mathcal{A},\mathcal{E},\mathcal{D}}^{\Pi,l}(\lambda)$: Run $(s, \pi; t, u) \leftarrow \mathcal{E}^{\mathcal{A}}(r_0, \ldots, r_{2 \cdot l-1})$. Output the single bit $\mathcal{D}(c, s, \pi) \wedge (\Pi.\mathsf{Verify}(\mathsf{params}, c, s, (r_0, \ldots, r_{2 \cdot l-1}), \pi) \implies (\Pi.\mathsf{Open}(\mathsf{params}, c; t, u) \wedge t(r_0, \ldots, r_{2 \cdot l-1}) = s))$.

The multilinear polynomial commitment scheme Π is said to be *extractable* with respect to the query sampler $\mathcal{Q}$ if, for each PPT adversary $\mathcal{A}$, there exists an expected PPT emulator $\mathcal{E}$ for which, for each PPT distinguisher $\mathcal{D}$, the distributions $\left\{\mathsf{Real}_{\mathcal{Q},\mathcal{A},\mathcal{E},\mathcal{D}}^{\Pi,l}(\lambda)\right\}_{l,\lambda \in \mathbb{N}}$ and $\left\{\mathsf{Emul}_{\mathcal{Q},\mathcal{A},\mathcal{E},\mathcal{D}}^{\Pi,l}(\lambda)\right\}_{l,\lambda \in \mathbb{N}}$ are statistically close.

In step 4 of Definition 4.3, we give $\mathcal{E}$ full rewinding access to $\mathcal{A}$, including its random tape; we suppress this fact for notational convenience. We emphasize that the implicit negligible function negl in Definition 4.3, which depends in general on $\mathcal{Q}$, $\mathcal{A}$, $\mathcal{E}$, and $\mathcal{D}$, is *not* allowed to depend on l; rather, it must work simultaneously for *all* $\lambda \in \mathbb{N}$ and $l \in \mathbb{N}$.

The following definition is a simplification of [CHM$^+$20, Def. 6.5], which requires that $\mathcal{Q}$ sample uniformly randomly ([CHM$^+$20, Def. 6.5] permits instead that $\mathcal{Q}$ sample uniformly from a superpolynomially large set).

Definition 4.4. The query sampler $\mathcal{Q}$ is *admissible* if, for each λ and l, and each parameter set $\mathsf{params} \leftarrow \Pi.\mathsf{Setup}(1^\lambda, l)$, containing the field size q, say, it holds that $(r_0, \ldots, r_{2\cdot l-1}) \leftarrow \mathcal{Q}(\mathsf{params})$ is uniform over $\mathbb{F}_q^{2\cdot l}$.

While our results go through for certain more general $\mathcal{Q}$, our applications require only that $(r_0, \ldots, r_{2\cdot l-1})$ be uniform; we thus restrict to this setting for simplicity. For example, our results hold even when we assume merely that $(r_0, \ldots, r_{2\cdot l-1}) \leftarrow \mathcal{Q}(\mathsf{params})$ is distributed in such a way that its projection $(r_l, \ldots, r_{2\cdot l-1})$ onto its second half is uniform over a set $S \subset \mathbb{F}_q^l$ of mass $\Omega(1)$ (where S may depend on λ, l, and params, and even on c, but its mass is nonetheless bounded from below by a constant independent of these quantities). As this definition's dependence on a projection is somewhat ad-hoc, we refrain from treating it explicitly.

Remark 4.5. We compare our definitional framework to those of Setty [Set20, Def. 2.11] and Bünz, Fisch and Szepieniec [BFS20, Def. 4] (which are identical) and to that of Marlin [CHM+20, Def. 6.2]. Our treatment can be viewed as the "meet" of these two approaches, as we presently explain. We prove our scheme's security only for a certain class of query samplers (as Marlin does); [Set20, Def. 2.11] and [BFS20, Def. 4] on the other hand require the scheme at hand to be simultaneously secure against *all* efficient query samplers. On the other hand, we allow our extractor full rewinding access to $\mathcal{A}$ (as [Set20, Def. 2.11] and [BFS20, Def. 4] do); Marlin instead requires that $\mathcal{E}$ extract t *immediately* after seeing c, before seeing $(r_0, \ldots, r_{2\cdot l-1})$ or π. Our definition thus selectively incorporates these definitions' respective slight weakenings with respect to each other. We note that Marlin's "early extraction" requirement meets the demands imposed by non-constant-round protocols, where, in fact, $\mathcal{E}$ must moreover be *non-rewinding*. As this latter setting doesn't apply to us, we accept the relaxation adopted by [Set20, Def. 2.11] and [BFS20, Def. 4] (again, however, with a restricted query sampler).

We now instantiate our concrete scheme Π in the random oracle model, using Merkle tree commitments, in a manner which evokes Ben-Sasson, Chiesa and Spooner [BSCS16]'s transformation from *interactive oracle proofs* to *non-interactive random oracle proofs*.

Construction 4.6 (Main polynomial commitment scheme)**.**
We define $\Pi = (\mathsf{Setup}, \mathsf{Commit}, \mathsf{Open}, \mathsf{Prove}, \mathsf{Verify})$ as follows.

- $\mathsf{params} \leftarrow \Pi.\mathsf{Setup}(1^\lambda, l)$. On input 1^λ and l, set $m := 2^l$, and return a prime power $q \geq 2^{\omega(\log \lambda)}$, an $[n, m, d]$-code $V \subset \mathbb{F}_q^n$ for which $n = 2^{O(l)}$ and $d = \Omega(n)$, and a repetition parameter $\kappa = \Theta(\lambda)$.
- $(c, u) \leftarrow \Pi.\mathsf{Commit}(\mathsf{params}, t)$. On input $t(X_0, \ldots, X_{2\cdot l-1}) \in \mathbb{F}_q[X_0, \ldots, X_{2\cdot l-1}]$, express $t = (t_0, \ldots, t_{m^2-1})$ in coordinates with respect to the Lagrange basis on $\{0,1\}^{2\cdot l}$, collate the resulting vector into an $m \times m$ matrix $(t_i)_{i=0}^{m-1}$, and encode $(t_i)_{i=0}^{m-1}$ row-wise, so obtaining a further matrix $(u_i)_{i=0}^{m-1}$. Output a Merkle commitment c to $(u_i)_{i=0}^{m-1}$ and the opening hint $u := (u_i)_{i=0}^{m-1}$.
- $b \leftarrow \Pi.\mathsf{Open}(\mathsf{params}, c; t, u)$. On input $t(X_0, \ldots, X_{2\cdot l-1}) \in \mathbb{F}_q[X_0, \ldots, X_{2\cdot l-1}]$, opening hint $(u_i)_{i=0}^{m-1}$, and commitment c, verify that $(u_i)_{i=0}^{m-1}$ Merkle-hashes to c. Collate t into a matrix $(t_i)_{i=0}^{m-1}$, encode the resulting matrix row-wise, and verify that $d^m\left((\mathsf{Enc}(t_i))_{i=0}^{m-1}, (u_i)_{i=0}^{m-1}\right) \overset{?}{<} \frac{d}{3}$.

We define $\Pi.\mathsf{Prove}$ and $\Pi.\mathsf{Verify}$ by applying the Fiat–Shamir heuristic to the following interactive protocol, where $\mathcal{P}$ has $t(X_0, \ldots, X_{2\cdot l-1})$ and $(u_i)_{i=0}^{m-1}$, and $\mathcal{P}$ and $\mathcal{V}$ have c, s, and $(r_0, \ldots, r_{2\cdot l-1}) \in \mathbb{F}_q^{2\cdot l}$.

- $\mathcal{P}$ sends $\mathcal{V}$ $t' := \bigotimes_{i=l}^{2\cdot l-1}(1 - r_i, r_i) \cdot (t_i)_{i=0}^{m-1}$ in the clear.
- For each $i \in \{0, \ldots, \kappa - 1\}$, $\mathcal{V}$ samples $j_i \leftarrow \{0, \ldots, n-1\}$. $\mathcal{V}$ sends $\mathcal{P}$ the set $J := \{j_0, \ldots, j_{\kappa-1}\}$.
- $\mathcal{P}$ sends $\mathcal{V}$ the columns $\left\{(u_{i,j})_{i=0}^{m-1}\right\}_{j \in J}$, each featuring an accompanying Merkle path against c.
- $\mathcal{V}$ computes $\mathsf{Enc}(t')$. For each $j \in J$, $\mathcal{V}$ verifies the Merkle path attesting to $(u_{i,j})_{i=0}^{m-1}$, and moreover requires that $\bigotimes_{i=l}^{2\cdot l-1}(1 - r_i, r_i) \cdot (u_{i,j})_{i=0}^{m-1} \overset{?}{=} \mathsf{Enc}(t')_j$. Finally, $\mathcal{V}$ requires $s \overset{?}{=} t' \cdot \bigotimes_{i=0}^{l-1}(1 - r_i, r_i)$.

Our scheme is clearly complete. We note that the requirement $n = 2^{O(l)}$ is necessary *merely* for V to be efficiently encodable. The requirement $d = \Omega(n)$ entails that V has constant relative distance.

We moreover have the following security guarantees:

Theorem 4.7. *The scheme of Construction 4.6 is binding.*

Proof. We fix a tuple (c, t_0, t_1, u_0, u_1) for which $\Pi.\mathsf{Open}(\mathsf{params}, c; t_0, s_0)$ and $\Pi.\mathsf{Open}(\mathsf{params}, c; t_1, s_1)$ both hold. Barring a collision in the random oracle, we have that $u_0 = u_1$; we write $(u_i)_{i=0}^{m-1}$ for the common hint. By hypothesis, the respective encodings $(\mathsf{Enc}(t_{0,i}))_{i=0}^{m-1}$ and $(\mathsf{Enc}(t_{1,i}))_{i=0}^{m-1}$ of t_0 and t_1 satisfy $d^m\left((\mathsf{Enc}(t_{0,i}))_{i=0}^{m-1}, (u_i)_{i=0}^{m-1}\right) < \frac{d}{3}$ and $d^m\left((\mathsf{Enc}(t_{1,i}))_{i=0}^{m-1}, (u_i)_{i=0}^{m-1}\right) < \frac{d}{3}$. By unique decoding, we conclude that $(\mathsf{Enc}(t_{0,i}))_{i=0}^{m-1} = (\mathsf{Enc}(t_{1,i}))_{i=0}^{m-1}$. Because Enc is injective, we finally conclude that $t_0 = t_1$. □

Theorem 4.8. *If the query sampler $\mathcal{Q}$ is admissible, then the scheme of Construction 4.6 is extractable.*

Proof. We define an extractor $\mathcal{E}$ in the following way. Given access to $\mathcal{A}$, and on inputs params, c and $(r_0, \ldots, r_{2\cdot l-1})$, $\mathcal{E}$ operates as follows:

1. $\mathcal{E}$ internally runs $\mathcal{A}$ on the further input $(r_0, \ldots, r_{2\cdot l-1})$ in a straight-line manner, until $\mathcal{A}$ outputs s and π. If $\Pi.\mathsf{Verify}(\mathsf{params}, c, s, (r_0, \ldots, r_{2\cdot l-1}), \pi) = 0$, then $\mathcal{E}$ outputs $(s, \pi; \perp, \perp)$ and terminates.

2. Having already obtained $\mathcal{A}$'s commitment c and observed $\mathcal{A}$'s random oracle queries, $\mathcal{E}$ runs the straight-line extractor of [BSCS16, § A.1], and so obtains $u := (u_i)_{i=0}^{m-1}$, or else outputs $(s, \pi; \perp, \perp)$ if it fails.

3. $\mathcal{E}$ writes $(r_{0,0}, \ldots, r_{0,2\cdot l-1})$ for the randomness it used above and t'_0 for the message sent by $\mathcal{A}$ during the course of its initial proof, and moreover proceeds as follows:

Algorithm 1

```
1: assign $i := 1$.
2: while $i < m$ do
3:     rewind $\mathcal{A}$ to its initial point (i.e., immediately after outputting $c$).
4:     freshly sample $(r_{i,0}, \ldots, r_{i,2\cdot l-1}) \leftarrow \mathcal{Q}(\mathsf{params})$.
5:     run $\mathcal{A}$ again on $(r_{i,0}, \ldots, r_{i,2\cdot l-1})$, with fresh verifier randomness, until it outputs $(s, \pi)$.
6:     if $\Pi.\mathsf{Verify}(\mathsf{params}, c, s, (r_{i,0}, \ldots, r_{i,2\cdot l-1}), \pi)$ then
7:         write $t'_i$ for the message sent by $\mathcal{A}$ during its proof.
8:         increment $i \mathrel{+}= 1$.
```

$\mathcal{E}$ checks if the $m \times m$ matrix $\left(\bigotimes_{j=l}^{2\cdot l-1}(1 - r_{i,j}, r_{i,j})\right)_{i=0}^{m-1}$ is invertible. If it's not, $\mathcal{E}$ outputs $(s, \pi; \perp, u)$.

4. Otherwise, $\mathcal{E}$ performs the matrix operation:

$$\begin{bmatrix} — & t_0 & — \\ & \cdots & \\ — & t_{m-1} & — \end{bmatrix} := \begin{bmatrix} — & \bigotimes_{j=l}^{2\cdot l-1}(1 - r_{0,j}, r_{0,j}) & — \\ & \cdots & \\ — & \bigotimes_{j=l}^{2\cdot l-1}(1 - r_{m-1,j}, r_{m-1,j}) & — \end{bmatrix}^{-1} \cdot \begin{bmatrix} — & t'_0 & — \\ & \cdots & \\ — & t'_{m-1} & — \end{bmatrix},$$

sets as $t(X_0, \cdots, X_{2\cdot l-1}) \in \mathbb{F}_q[X_0, \cdots, X_{2\cdot l-1}]$ the polynomial whose coefficients (in the Lagrange basis) are given by the concatenation of $(t_i)_{i=0}^{m-1}$'s rows, and outputs $(s, \pi; t, u)$.

We now argue that $\mathcal{E}$ meets the requirements of Definition 4.3. We first argue that $\mathcal{E}$ runs in expected polynomial time in λ. We write ε for the probability that $\mathcal{A}$ passes, *conditioned* on its state as of the point at which it outputs c (this latter probability is taken over the coins of both $\mathcal{Q}$ and $\mathcal{V}$, and over the further coins of $\mathcal{A}$). We note that, for each c, $\mathcal{E}$ enters Algorithm 1 above with probability exactly ε, and, moreover, satisfies the condition of line 6 with probability exactly ε per iteration of that algorithm (since $\mathcal{E}$ simulates the same view to $\mathcal{A}$ during each successive iteration of Algorithm 3 as it does during its initial execution of $\mathcal{A}$). We conclude that $\mathcal{E}$'s total expected runtime is at most $1 + \varepsilon \cdot \frac{m-1}{\varepsilon} = m$ times the time it takes to run Construction 4.6 once; this total time is thus polynomial in λ (and independent of c and ε).

We now analyze the distribution returned by $\mathcal{E}$. We note that the outputs (c, s, π) upon which $\mathcal{D}$ runs are identically distributed in the distributions $\mathsf{Real}^{\Pi,l}_{\mathcal{Q},\mathcal{A},\mathcal{E},\mathcal{D}}(\lambda)$ and $\mathsf{Emul}^{\Pi,l}_{\mathcal{Q},\mathcal{A},\mathcal{E},\mathcal{D}}(\lambda)$. It thus suffices to show that it holds in at most negligibly many executions of $\mathcal{A}$, $\mathcal{Q}$ and $\mathcal{E}$ that, simultaneously, $\Pi.\mathsf{Verify}(\mathsf{params}, c, s, (r_0, \ldots, r_{2\cdot l-1}), \pi) = 1$ and *either* $\Pi.\mathsf{Open}(\mathsf{params}, t, c) = 0$ or $t(r_0, \ldots, r_{2\cdot l-1}) \neq s$.

We write $Q(\lambda)$ for the number of queries $\mathcal{A}$ makes to the random oracle during one execution. By [BSCS16, Lem. 3], it holds with probability at most $\frac{Q(\lambda)^2+1}{2^\lambda}$, which is negligible, that *either* $\mathcal{E}$ fails to extract c in step 2 *or* that $\mathcal{A}$ outputs inconsistent Merkle leaves during its initial proof π. We thus freely assume that $\mathcal{E}$ successfully extracts the opening hint $u := (u_i)_{i=0}^{m-1}$ in step 2 and that, throughout its initial proof π, $\mathcal{A}$ outputs Merkle leaves $(u_{i,j})_{i=0}^{m-1}$ consistent with $(u_i)_{i=0}^{m-1}$.

The following lemma shows that we may, moreover, safely restrict our attention to the setting in which the extracted matrix $(u_i)_{i=0}^{m-1}$ is close to the code.

Lemma 4.9. *If its initial matrix satisfies $d^m\Big((u_i)_{i=0}^{m-1}, V^m\Big) \geq \frac{d}{3}$, then $\mathcal{A}$ passes with negligible probability.*

Proof. We write $e := \lfloor \frac{d-1}{3} \rfloor$, and abbreviate $u' := \bigotimes_{i=l}^{2\cdot l-1}(1-r_i, r_i) \cdot (u_i)_{i=0}^{m-1}$. Under the hypothesis of the lemma, Theorem 3.1 implies that, if the second half $(r_l, \ldots, r_{2\cdot l-1}) \in \mathbb{F}_q^l$ of the verifier's random point resides *outside* a set of mass at most $2 \cdot l \cdot \frac{e}{q}$ in $\mathbb{F}_q^l$, then we have $d(u', V) > e$. For such $(r_l, \ldots, r_{2\cdot l-1})$, we thus have in particular that $d(u', \mathsf{Enc}(t')) > e$, since $\mathsf{Enc}(t')$ is a codeword. It follows that $\Pr_{j \leftarrow \{0,\ldots,n-1\}}\left[u'_j = \mathsf{Enc}(t')_j\right] < 1-\frac{e}{n}$. Finally, by our assumption about $\mathcal{A}$'s Merkle paths, we necessarily have $\bigotimes_{i=l}^{2\cdot l-1}(1-r_i, r_i) \cdot (u_{i,j})_{i=0}^{m-1} = u'_j$ for each column $(u_{i,j})_{i=0}^{m-1}$ output by $\mathcal{A}$. We see that $\mathcal{A}$'s chance of passing (over the coins of $\mathcal{Q}$ and $\mathcal{V}$) is at most $l \cdot \frac{2\cdot d}{3\cdot q} + \left(1 - \frac{d-3}{3\cdot n}\right)^\kappa$. As $q \geq 2^{\omega(\log \lambda)}$ holds by construction, and d and l are polynomial in λ, we see that $l \cdot \frac{2\cdot d}{3\cdot q}$ is negligible. On the other hand, because $d \in \Omega(n)$ and $\kappa \in \Theta(\lambda)$, we likewise have that $\left(1 - \frac{d-3}{3\cdot n}\right)^\kappa \leq (1 - \Omega(1))^{\Theta(\lambda)}$ is negligible. This completes the proof of the lemma. □

Indeed, the lemma shows that it holds only for a negligible proportion of executions that $d^m\Big((u_i)_{i=0}^{m-1}, V^m\Big) \geq \frac{d}{3}$ *and* $\mathcal{E}$ proceeds into step 3. We may thus safely ignore these executions. We accordingly assume in what follows that $d^m\Big((u_i)_{i=0}^{m-1}, V^m\Big) < \frac{d}{3}$. In particular, it holds that the respective messages underlying $\mathcal{A}$'s initial matrix $(u_i)_{i=0}^{m-1}$ are well-defined; we write $(t_i)_{i=0}^{m-1}$ for these messages. The following lemma shows that we may *further* restrict our attention to the case in which $\mathcal{A}$ correctly outputs $t' = \bigotimes_{i=l}^{2\cdot l-1}(1-r_i, r_i) \cdot (t_i)_{i=0}^{m-1}$ during its initial proof.

Lemma 4.10. *If its message $t' \neq \bigotimes_{i=l}^{2\cdot l-1}(1-r_i, r_i) \cdot (t_i)_{i=0}^{m-1}$, then $\mathcal{A}$ passes with negligible probability.*

Proof. We again fix $e := \lfloor \frac{d-1}{3} \rfloor$ and abbreviate $u' := \bigotimes_{i=l}^{2\cdot l-1}(1-r_i, r_i) \cdot (u_i)_{i=0}^{m-1}$; we moreover write $v' := \bigotimes_{i=l}^{2\cdot l-1}(1-r_i, r_i) \cdot (\mathsf{Enc}(t_i))_{i=0}^{m-1}$. By our assumption above, $d^m\Big((u_i)_{i=0}^{m-1}, V^m\Big) \leq e$ holds; in particular, $d(u', v') \leq e$. On the other hand, our hypothesis implies that $\mathsf{Enc}(t') \neq v'$. By the reverse triangle inequality, we thus have:

$$d(u', \mathsf{Enc}(t')) \geq |d(\mathsf{Enc}(t'), v') - d(u', v')| \geq d - e.$$

As above, we conclude that $\Pr_{j \leftarrow \{0,\ldots,n-1\}}\left[u'_j = \mathsf{Enc}(t')_j\right] \leq 1-\frac{d-e}{n}$; moreover, we again have that $\bigotimes_{i=l}^{2\cdot l-1}(1-r_i, r_i) \cdot (u_{i,j})_{i=0}^{l-1} = u'_j$ for each $j \in J$ queried by the verifier. We thus upper-bound $\mathcal{A}$'s probability of passing by $\left(1 - \frac{2\cdot d}{3\cdot n}\right)^\kappa$. This again completes the proof, in light of the guarantees $d \in \Omega(n)$ and $\kappa \in \Theta(\lambda)$. □

We thus restrict our attention to the case in which $\mathcal{A}$'s initial proof π verifies, $\mathcal{E}$ successfully extracts $(u_i)_{i=0}^{m-1}$, and *both* $d^m\Big((u_i)_{i=0}^{m-1}, V^m\Big) < \frac{d}{3}$ and $t' = \bigotimes_{i=l}^{2\cdot l-1}(1-r_i, r_i) \cdot (t_i)_{i=0}^{m-1}$ hold. We denote:

$$\delta := \frac{l}{q} + \left(1 - \frac{2\cdot d}{3\cdot n}\right)^\kappa + \frac{Q(\lambda)^2+1}{2^\lambda}.$$

Since δ is negligible in λ, $\sqrt{\delta}$ also is. In this light, we may simply ignore each execution for which $\mathcal{A}$'s probability of success $\varepsilon \leq \sqrt{\delta}$, since in that case $\mathcal{E}$ proceeds into step 3 in the first place with negligible probability. We thus assume that $\varepsilon > \sqrt{\delta}$ in what follows.

We first show that, with overwhelming probability, as of the conclusion of *each* iteration $i \in \{1, \ldots, m-1\}$ of Algorithm 1, we have $t'_i = \bigotimes_{j=l}^{2\cdot l-1}(1 - r_{i,j}, r_{i,j}) \cdot (t_i)_{i=0}^{m-1}$.

Lemma 4.11. *The probability that* $t'_i \neq \bigotimes_{j=l}^{2\cdot l-1}(1-r_{i,j}, r_{i,j}) \cdot (t_i)_{i=0}^{m-1}$ *for* any $i \in \{1, \ldots, m-1\}$ *is negligible.*

Proof. We write V for the event in which $\mathcal{A}$ submits an accepting proof, and E for the event in which $\mathcal{A}$ outputs the correct message $t' = \bigotimes_{i=l}^{2\cdot l-1}(1 - r_i, r_i) \cdot (t_i)_{i=0}^{m-1}$. By the argument of Lemma 4.10, *if* $\mathcal{A}$ submits Merkle leaves consistent with $(u_i)_{i=0}^{m-1}$, then $\mathcal{V}$ accepts with probability at most $\left(1 - \frac{2\cdot d}{3\cdot n}\right)^{\kappa}$. By a union bound, we conclude that $\Pr[V \mid \neg E] \leq \frac{Q(\lambda)^2+1}{2^\lambda} + \left(1 - \frac{2\cdot d}{3\cdot n}\right)^{\kappa} \leq \delta$. Our assumption $\varepsilon > \sqrt{\delta}$ thus implies:

$$\sqrt{\delta} < \varepsilon = \Pr[V] = \Pr[V \wedge E] + \Pr[V \mid \neg E] \cdot \Pr[\neg E] \leq \Pr[V \wedge E] + \Pr[\neg E] \cdot \delta,$$

so that $\Pr[V \wedge E] > \sqrt{\delta} - \Pr[\neg E] \cdot \delta \geq \sqrt{\delta} - \delta$.

By Bayes' theorem—and using the facts noted above—we conclude that:

$$\Pr[E \mid V] = \frac{\Pr[V \wedge E]}{\Pr[V \wedge E] + \Pr[V \mid \neg E] \cdot \Pr[\neg E]} > \frac{\sqrt{\delta} - \delta}{\sqrt{\delta} - \delta + \delta} = 1 - \sqrt{\delta}.$$

We thus see that, under our assumption, the probability that *all* of $\mathcal{E}$'s iterations $i \in \{1, \ldots, m-1\}$ satisfy $t'_i = \bigotimes_{j=l}^{2\cdot l-1}(1 - r_{i,j}, r_{i,j}) \cdot (t_i)_{i=0}^{m-1}$ is greater than $\left(1 - \sqrt{\delta}\right)^{m-1}$, which is overwhelming. Indeed, by a standard binomial approximation, we have that $1 - \left(1 - \sqrt{\delta}\right)^{m-1} \leq (m-1) \cdot \sqrt{\delta}$, which is negligible. □

The following lemma is the last remaining step.

Lemma 4.12. *The probability that the rows* $\left(\bigotimes_{j=l}^{2\cdot l-1}(1 - r_{i,j}, r_{i,j})\right)_{i=0}^{m-1}$ *are linearly dependent is negligible.*

Proof. We fix an arbitrary *proper* linear subspace $A \subset \mathbb{F}_q^m$, and moreover define its preimage $S := \left\{(r_l, \ldots, r_{2\cdot l-1}) \in \mathbb{F}_q^l \;\middle|\; \bigotimes_{i=l}^{2\cdot l-1}(1 - r_i, r_i) \in A\right\}$ under the tensor map. We first argue that $\mu(S) \leq \frac{l}{q}$. It suffices to prove the result only in case A is a hyperplane. We write $a = (a_0, \ldots, a_{m-1})$ for a vector of coefficients, *not* all zero, for which $A = \left\{u \in \mathbb{F}_q^m \;\middle|\; u \cdot a = 0\right\}$ holds. By construction, $(r_l, \ldots, r_{2\cdot l-1}) \in S$ if and only if $\bigotimes_{i=l}^{2\cdot l-1}(1 - r_i, r_i) \cdot a = 0$.

We note that $S \subset \mathbb{F}_q^l$ is nothing other than the vanishing locus of that combination of the l-variate multilinear Lagrange polynomials given by the coefficient vector a. Because a is not identically zero and these polynomials are linearly independent, we conclude that the combination is itself nonzero. Applying Schwartz–Zippel, we conclude that its vanishing locus $S \subset \mathbb{F}_q^l$ is of mass at most $\mu(S) \leq \frac{l}{q}$, as desired.

As before, we write V for the event in which $\mathcal{A}$ submits an accepting proof; slightly abusing notation, we denote also by S the event in which $\mathcal{Q}$'s query satisfies $(r_l, \ldots, r_{2\cdot l-1}) \in S$. By the above argument, we have that $\mu(S) = \frac{l}{q} \leq \delta$. On the other hand, by our assumption $\varepsilon > \sqrt{\delta}$, we have:

$$\sqrt{\delta} < \varepsilon = \Pr[V] \leq \mu(S) + \Pr[V \mid \neg S] \cdot (1 - \mu(S)),$$

so that $\Pr[V \mid \neg S] \cdot (1 - \mu(S)) > \sqrt{\delta} - \mu(S) \geq \sqrt{\delta} - \delta$.

Again using Bayes' theorem, and using the above facts, we have that:

$$\Pr[\neg S \mid V] = \frac{\Pr[V \mid \neg S] \cdot (1 - \mu(S))}{\Pr[V \mid \neg S] \cdot (1 - \mu(S)) + \Pr[V \mid S] \cdot \mu(S)} > \frac{\sqrt{\delta} - \delta}{\sqrt{\delta} - \delta + \delta} = 1 - \sqrt{\delta}.$$

We now consider the successive vectors $(r_{i,l}, \ldots, r_{i,2\cdot l-1}) \in \mathbb{F}_q^l$ collected by $\mathcal{E}$ over the course of its *successful* executions of the main loop of Algorithm 3. For each $i^* \in \{1, \ldots, m-1\}$, writing $A \subset \mathbb{F}_q^m$ for the span of the vectors $\left(\bigotimes_{j=l}^{2\cdot l-1}(1 - r_{i,j}, r_{i,j})\right)_{i=0}^{i^*-1}$, we conclude that, with probability greater than $1 - \sqrt{\delta}$, the vector $(r_{i^*,l}, \ldots, r_{i^*,2\cdot l-1}) \in \mathbb{F}_q^l$ collected during the $i^{*\text{th}}$ iteration satisfies $\bigotimes_{j=l}^{2\cdot l-1}(1 - r_{i^*,j}, r_{i^*,j}) \notin A$. We see that the rows $\left(\bigotimes_{j=l}^{2\cdot l-1}(1 - r_{i,j}, r_{i,j})\right)_{i=0}^{m-1}$ are independent with probability greater than $\left(1 - \sqrt{\delta}\right)^{m-1}$, which is overwhelming, since $1 - \left(1 - \sqrt{\delta}\right)^{m-1} \leq (m-1) \cdot \sqrt{\delta}$ is negligible. This completes the proof of the lemma. □

We finally argue that the values t and $u = (u_i)_{i=0}^{m-1}$ extracted by $\mathcal{E}$ satisfy $\Pi.\mathsf{Open}(\mathsf{params}, c; t, u)$ and $t(r_0, \ldots, r_{2\cdot l-1}) = s$. Indeed, under the condition guaranteed by the successful execution of step 2 above, $\mathcal{A}$'s initial matrix $(u_i)_{i=0}^{m-1}$ is well-defined, and hashes to c, and moreover is successfully extracted by $\mathcal{E}$. Under the condition guaranteed by Lemma 4.9, a matrix $(t_i)_{i=0}^{m-1}$ for which $d^m\left((\mathsf{Enc}(t_i))_{i=0}^{m-1}, (u_i)_{i=0}^{m-1}\right) < \frac{d}{3}$ exists and is unique. Under the conditions guaranteed by Lemmas 4.11 and 4.12, $\mathcal{E}$ extracts precisely this matrix $(t_i)_{i=0}^{m-1}$ in steps 3 and 4. Finally, Lemma 4.10 guarantees that $\mathcal{A}$'s first message satisfies $t' = \bigotimes_{i=l}^{2\cdot l-1}(1-r_i, r_i) \cdot (t_i)_{i=0}^{m-1}$; on the other hand, $\Pi.\mathsf{Verify}(\mathsf{params}, c, s, (r_0, \ldots, r_{2\cdot l-1}), \pi)$ implies that $s = t' \cdot \bigotimes_{i=0}^{l-1}(1-r_i, r_i)$. We conclude that $s = \bigotimes_{i=l}^{2\cdot l-1}(1-r_i, r_i) \cdot (t_i)_{i=0}^{m-1} \cdot \bigotimes_{i=0}^{l-1}(1-r_i, r_i) = t(r_0, \ldots, r_{2\cdot l-1})$, as required. This completes the proof of the theorem. □

We record a few remarks about our proof. Theorem 4.8's difficulty arises, roughly, from the fact that the *conditional* distribution of the messages t'_i and of the random vectors $(r_{i,0}, \ldots, r_{i,2\cdot l-1}) \in \mathbb{F}_q^{2\cdot l}$ which $\mathcal{E}$ adds—that is, the distribution of these values, conditioned on $\mathcal{A}$ passing—can be highly arbitrary; $\mathcal{A}$ could, for example, output a successful proof with vastly higher probability when $\bigotimes_{i=l}^{2\cdot l-1}(1-r_i, r_i)$ resides in some fixed low-dimensional subspace $A \subset \mathbb{F}_q^m$ (let's say) than when it doesn't, thereby thwarting $\mathcal{E}$'s extraction. Our proof thus argues that if $\mathcal{A}$ succeeds with high enough probability—specifically, with probability greater than a certain cutoff which, crucially, is *still negligible*, but which decays much more slowly than that of the relevant failure events—then the *conditional* distribution of $\mathcal{A}$'s outputs necessarily concentrates away from these bad events. The key idea is that δ, by virtue of being negligible, necessarily admits an expression of the form $\delta = 2^{-f(\lambda)}$, for some $f(\lambda) \in \omega(\log(\lambda))$; we thus have in turn that $\sqrt{\delta}$ takes the form $2^{-\frac{1}{2}\cdot f(\lambda)}$. This latter quantity is *greater* than δ by a factor of $2^{\frac{1}{2}\cdot f(\lambda)}$, which is superpolynomial; on the other hand, it is *itself* nonetheless still negligible. This maneuver, whereby the exponent is halved, can be performed on any negligible function. Upon excluding from our treatment those executions for which $\varepsilon \leq \sqrt{\delta}$, we find that, in the remaining executions, $\mathcal{A}$'s success probability is sufficiently "high" that failure events necessarily figure negligibly in it, regardless of $\mathcal{A}$'s strategy. This latter step is made precise by means of Bayes' theorem.

Remark 4.13. We we compare our proof strategy to that of Brakedown [GLS+21, Lem. 2], which seeks to prove a similar result. Brakedown's proof, essentially, handles the non-uniformity of $\mathcal{A}$'s conditional output distribution by stipulating that the *emulator* $\mathcal{E}$ filter "actively", using rejection sampling to curate an *artificially uniform* distribution over some sufficiently large set of coefficient vectors. We note that that procedure requires that $\mathcal{E}$ "know" $\mathcal{A}$'s success probability ε. That knowledge is actually forbidden to $\mathcal{E}$; indeed, though $\mathcal{E}$ may depend on $\mathcal{A}$, $\mathcal{A}$'s success probability $\varepsilon := \varepsilon(\mathcal{A}, c)$ can depend arbitrarily, in general, on its initial commitment c, and $\mathcal{E}$ must work simultaneously for *all possible* commitments c. The emulator $\mathcal{E}$ similarly can't "hardcode" the probability $\varepsilon := \varepsilon(\mathcal{A}, c)$ *for each possible* c, even given polynomially many advice bits. We refer, for example, to the remark of Attema, Fehr, and Klooß [AFK22] that "We stress though that $\mathcal{E}$ cannot depend on (or 'know') certain properties of $\mathcal{A}$, such as . . . the success probability $\varepsilon(\mathcal{A}, c)$" (we have slightly adjusted their notation). This problem is separate from that of blackbox access, and affects $\mathcal{E}$ *even* given its non-blackbox access to $\mathcal{A}$. This issue invalidates Brakedown's proof as written. (We use ε in our *analysis* of $\mathcal{E}$'s effectiveness, but *not* in its very operation.) In fact, a technique by which $\mathcal{E}$ can *estimate* ε—with overwhelming confidence, over a range—appears in a work of Lindell and Hazay [LH10, Thm. 6.5.6], who in turn attribute the idea to Goldreich. It seems possible that Brakedown could salvage its proof by means of that measure (though we note that the resulting emulator would be significantly more complex than ours).

Remark 4.14. We note a further issue with Brakedown's proof, as well as a simple amendment. That proof argues that $(m-1) \cdot \frac{2}{\varepsilon \cdot q}$ "is negligible by our assumption that ε is non-negligible while $\frac{1}{q}$ is negligible". This assertion is not valid. Indeed, that ε is non-negligible entails exactly (unwinding definitions) that there exists a polynomial $p(\lambda)$ for which, for infinitely many λ, $\frac{1}{\varepsilon} \leq p(\lambda)$. We obtain no guarantees, however, about how $\frac{1}{\varepsilon}$ behaves for *the other* λ; in particular, it could be enormous (say, on the order of 2^λ). For these other λ, therefore, we cannot conclude anything about $\frac{1}{\varepsilon} \cdot \frac{2}{q}$, let alone that it decays faster than any polynomial in λ. The proof could perhaps go through—up to the issue discussed in the above remark—if, instead, it treated not "non-negligible" ε, but rather $\varepsilon > \frac{1}{\sqrt{q}}$. Executions for which $\varepsilon \leq \frac{1}{\sqrt{q}}$ succeed with negligible chance, and can be ignored; when $\varepsilon > \frac{1}{\sqrt{q}}$, on the other hand, we have that $\frac{2}{\varepsilon \cdot q} < \frac{2}{\sqrt{q}}$, which *is* negligible (provided $\frac{1}{q}$ is).

We discuss the efficiency of Construction 4.6. Implemented naïvely, Construction 4.6 admits proofs with exactly $m \cdot (\kappa + 1)$ $\mathbb{F}_q$-elements ($\mathcal{P}$ must send the single m-element message t', as well as κ m-element columns $(u_{i,j})_{i=0}^{m-1}$). We recall an optimization discussed in Brakedown [GLS+21, § 4], and attributed by that work to Ligero. Construction 4.6 works even when the input matrix $(t_i)_{i=0}^{m-1}$ is not square, but rather of size $m_0 \times m_1$, say, where $m_0 \cdot m_1 = 2^{2 \cdot l}$. Moreover, the resulting variant of the protocol has proof size exactly $m_1 + \kappa \cdot m_0$. To minimize this size, we choose m_0 and m_1 so that $m_1 = \kappa \cdot m_0$ holds; in particular, we set $m_0 := \frac{1}{\sqrt{\kappa}} \cdot m$ and $m_1 := \sqrt{\kappa} \cdot m$ (where m here denotes 2^l). The resulting proof clearly has size $m_1 + \kappa \cdot m_0 = 2 \cdot \sqrt{\kappa} \cdot m$. This measure thus improves the proof size quadratically in κ (compared to the naïve approach in which a square matrix is used). The *standard* Ligero commitment scheme's proof size, on the other hand, is $2 \cdot m_1 + \kappa \cdot m_0$, since two messages must be sent (our improvement eliminates this factor of two). Brakedown's optimization thus seeks to achieve $2 \cdot m_1 = \kappa \cdot m_0$, and accordingly sets $m_0 := \sqrt{\frac{2}{\kappa}} \cdot m$ and $m_1 := \sqrt{\frac{\kappa}{2}} \cdot m$. The resulting proof size is thus $2 \cdot \sqrt{2 \cdot \kappa} \cdot m$. We note the resulting extra factor of $\sqrt{2}$, absent from our proof's size.

We finally discuss Construction 4.6's prover and verifier runtime efficiency (*after* the above proof size optimization is incorporated). We write $\mathsf{Enc}(\lambda)$ for the runtime of V's encoding procedure. It is easy to see that $\mathcal{P}$'s runtime is $\frac{1}{\sqrt{\kappa}} \cdot m \cdot \mathsf{Enc}(\lambda)$ during the commitment phase and $\frac{1}{\sqrt{\kappa}} \cdot m \cdot \sqrt{\kappa} \cdot m = m^2$ in the evaluation phase, for a total of $\frac{1}{\sqrt{\kappa}} \cdot m \cdot \mathsf{Enc}(\lambda) + m^2$. In the special case that Enc is linear-time in m—Brakedown's code [GLS+21, § 5] satisfies this property—the *total* cost of both phases becomes $m^2 + O\left(\frac{1}{\sqrt{\lambda}} \cdot m^2\right)$; since each $t(X_0, \ldots, X_{2 \cdot l - 1})$ requires m^2 field elements to represent, this efficiency is essentially optimal. The prover cost of the *standard* Ligero scheme is $\frac{1}{\sqrt{\kappa}} \cdot m \cdot \mathsf{Enc}(\lambda) + 2 \cdot m^2$ (we note the extra factor of 2). Specializing again to the linear-time-encodable case, we obtain a total cost of $2 \cdot m^2 + O\left(\frac{1}{\sqrt{\lambda}} \cdot m^2\right)$; we see that we improve the prover runtime of Brakedown's commitment scheme by a factor of 2, up to lower-order terms. (If the implicit linear constant in the runtime of Enc is very large, however, then our improvement may remain limited until λ becomes large.)

Construction 4.6's verifier complexity is $\mathsf{Enc}(\lambda) + \kappa \cdot \frac{1}{\sqrt{\kappa}} \cdot m = \mathsf{Enc}(\lambda) + \sqrt{\kappa} \cdot m$. Assuming again that Enc is linear-time in m, this cost becomes $\sqrt{\kappa} \cdot m + O(m)$, which is of square-root complexity in both λ and the size of t. The verifier complexity of the *standard* Ligero scheme is $2 \cdot \mathsf{Enc}(\lambda) + 2 \cdot \sqrt{2 \cdot \kappa} \cdot m$. We thus improve the verifier's complexity as well by a factor of more than two.

5 Concrete Efficiency

We implemented our modified polynomial commitment scheme by modifying the repository `conroi / lcpc`. We ran all benchmarks on a `c7g.8xlarge` AWS instance, with an AWS *Graviton3* processor with 32 virtual cores. We used a prime field $\mathbb{F}_q$ of 191 bits, so attaining a security level of at least 128 bits throughout. We used the hash function *Blake3*. Benchmarks are given in Tables 1, 2 and 3.

Tables 1, 2, and 3 exhibit improvements matching those predicted by the abstract efficiency analysis in Section 4. We recall that we used matrix sizes designed to minimize *proof size* throughout. Different choice strategies could target, for example, verifier time. Finally, all benchmarks include various lower-order costs, including the costs of generating, transmitting, and verifying Merkle paths, for example.

We record a remark about the concrete security achieved by our protocol. The analyses of Lemmas 4.9 and 4.10 show that the prover's soundness is controlled by the expression $l \cdot \frac{2 \cdot d}{3 \cdot q} + \left(1 - \frac{d-3}{3 \cdot n}\right)^{\kappa}$ (the soundness error of Lemma 4.9 dominates). This expression's *first term* is larger by a factor of $2 \cdot l = 2 \cdot \log\left(\frac{m}{\sqrt{\kappa}}\right)$ than that of the analogous expression in [GLS+21]. For all sizes benchmarked above, for which $m^2 \leq 2^{28}$, we have that $\log(2 \cdot l) = \log\left(2 \cdot \log\left(\frac{m}{\sqrt{\kappa}}\right)\right) \leq 5$; in this light, our protocol technically requires a field $\mathbb{F}_q$ roughly 5 bits larger (in the worst case) in order to achieve equivalent security. On the other hand, in practice, our chosen field size is governed by the limb size of our machines; this 5-bit difference is thus immaterial in practice. This picture would be different if our *logarithmic* loss l were replaced by, say, a *linear* loss proportional to m.

Commitment Scheme		Number of Coefficients			
Variant	Code	2^{16}	2^{20}	2^{24}	2^{28}
[GLS+21]	Reed–Solomon, $\rho = \frac{1}{2}$	2.054	8.146	54.256	632.380
	Reed–Solomon, $\rho = \frac{1}{4}$	1.389	5.448	45.183	603.177
	Brakedown, $\rho = 0.65$	8.138	26.136	119.125	876.627
This work	Reed–Solomon, $\rho = \frac{1}{2}$	1.183	4.252	27.163	314.819
	Reed–Solomon, $\rho = \frac{1}{4}$	1.122	4.190	27.011	313.479
	Brakedown, $\rho = 0.65$	5.197	14.173	60.700	439.457

Table 1: Time (ms) for $\Pi.\mathsf{Prove}$, $\lceil \log q \rceil = 191$

Commitment Scheme		Number of Coefficients			
Variant	Code	2^{16}	2^{20}	2^{24}	2^{28}
[GLS+21]	Reed–Solomon, $\rho = \frac{1}{2}$	4.501	11.314	34.308	119.849
	Reed–Solomon, $\rho = \frac{1}{4}$	4.039	9.211	24.582	79.406
	Brakedown, $\rho = 0.65$	23.758	89.216	384.866	2,266.047
This work	Reed–Solomon, $\rho = \frac{1}{2}$	2.582	6.158	17.992	61.849
	Reed–Solomon, $\rho = \frac{1}{4}$	3.155	7.501	21.804	75.097
	Brakedown, $\rho = 0.65$	12.883	45.780	194.597	1,114.807

Table 2: Time (ms) for $\Pi.\mathsf{Verify}$, $\lceil \log q \rceil = 191$

Commitment Scheme		Number of Coefficients			
Variant	Code	2^{16}	2^{20}	2^{24}	2^{28}
[GLS+21]	Reed–Solomon, $\rho = \frac{1}{2}$	0.459	1.384	5.016	19.471
	Reed–Solomon, $\rho = \frac{1}{4}$	0.329	1.040	3.841	14.999
	Brakedown, $\rho = 0.65$	5.227	9.791	26.537	92.013
This work	Reed–Solomon, $\rho = \frac{1}{2}$	0.365	1.009	3.516	13.471
	Reed–Solomon, $\rho = \frac{1}{4}$	0.267	0.771	2.740	10.557
	Brakedown, $\rho = 0.65$	4.852	8.291	20.537	68.013

Table 3: Proof size (MiB), $\lceil \log q \rceil = 191$

References

[AFK22] Thomas Attema, Serge Fehr, and Michael Klooß. Fiat–Shamir transformation of multi-round interactive proofs. In Eike Kiltz and Vinod Vaikuntanathan, editors, *Theory of Cryptography*, pages 113–142, Cham, 2022. Springer Nature Switzerland.

[AHIV17] Scott Ames, Carmit Hazay, Yuval Ishai, and Muthuramakrishnan Venkitasubramaniam. Ligero: Lightweight sublinear arguments without a trusted setup. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, CCS '17, pages 2087—2104, New York, NY, USA, 2017. Association for Computing Machinery. Updated full version.

[BCC+16] Jonathan Bootle, Andrea Cerulli, Pyrros Chaidos, Jens Groth, and Christophe Petit. Efficient zero-knowledge arguments for arithmetic circuits in the discrete log setting. In Marc Fischlin and

Jean-Sébastien Coron, editors, *Advances in Cryptology – EUROCRYPT 2016*, pages 327–357, Berlin, Heidelberg, 2016. Springer Berlin Heidelberg.

[BCG20] Jonathan Bootle, Alessandro Chiesa, and Jens Groth. Linear-time arguments with sublinear verification from tensor codes. In Rafael Pass and Krzysztof Pietrzak, editors, *Theory of Cryptography*, Cham, 2020. Springer International Publishing.

[BFS20] Benedikt Bünz, Ben Fisch, and Alan Szepieniec. Transparent SNARKs from DARK compilers. In Anne Canteaut and Yuval Ishai, editors, *Advances in Cryptology – EUROCRYPT 2020*, pages 677–706, Cham, 2020. Springer International Publishing.

[BS22] Alexandre Belling and Azam Soleimanian. Vortex: Building a lattice-based SNARK scheme with transparent setup. `https://ia.cr/2022/1633`, 2022.

[BSCI$^+$23] Eli Ben-Sasson, Dan Carmon, Yuval Ishai, Swastik Kopparty, and Shubhangi Saraf. Proximity gaps for Reed–Solomon codes. *Journal of the ACM*, 70(5), October 2023.

[BSCS16] Eli Ben-Sasson, Alessandro Chiesa, and Nicholas Spooner. Interactive oracle proofs. In *Proceedings, Part II, of the 14th International Conference on Theory of Cryptography*, volume 9986, pages 31–60, Berlin, Heidelberg, 2016. Springer-Verlag.

[BSKS18] Eli Ben-Sasson, Swastik Kopparty, and Shubhangi Saraf. Worst-case to average case reductions for the distance to a code. In Rocco A. Servedio, editor, *33rd Computational Complexity Conference*, pages 24:1–24:23. Dagstuhl Publishing, 2018.

[CHM$^+$20] Alessandro Chiesa, Yuncong Hu, Mary Maller, Pratyush Mishra, Noah Vesely, and Nicholas Ward. Marlin: Preprocessing zkSNARKs with universal and updatable SRS. In Anne Canteaut and Yuval Ishai, editors, *Advances in Cryptology – EUROCRYPT 2020*, Lecture Notes in Computer Science, pages 738–768, Cham, 2020. Springer International Publishing. Full version.

[GLS$^+$21] Alexander Golovnev, Jonathan Lee, Srinath Setty, Justin Thaler, and Riad S. Wahby. Brakedown: Linear-time and post-quantum SNARKs for R1CS. `https://ia.cr/2021/1043`, 2021.

[LH10] Yehuda Lindell and Carmit Hazay. *Efficient Secure Two-Party Protocols*. Information Security and Cryptography. Springer, Berlin, Heidelberg, 2010.

[Lin17] Yehuda Lindell. How to simulate it – a tutorial on the simulation proof technique. In Yehuda Lindell, editor, *Tutorials on the Foundations of Cryptography: Dedicated to Oded Goldreich*, pages 277–346. Springer International Publishing, Cham, 2017.

[Set20] Srinath Setty. Spartan: Efficient and general-purpose zkSNARKs without trusted setup. In Daniele Micciancio and Thomas Ristenpart, editors, *Advances in Cryptology – CRYPTO 2020*, Cham, 2020. Springer International Publishing. Full version.

[XZS22] Tiancheng Xie, Yupeng Zhang, and Dawn Song. Orion: Zero knowledge proof with linear prover time. In *Advances in Cryptology – CRYPTO 2022*, pages 299–328, Berlin, Heidelberg, 2022. Springer-Verlag. Full version.