Succinct Proofs and Linear Algebra

Alex Evans
aevans@baincapital.com

Guillermo Angeris
gangeris@baincapital.com

September 2023

Abstract

Succinct proof systems play an important role in many contexts, including in blockchains and privacy, but it is often difficult to understand how these systems work without understanding some of the deep cryptographic techniques that are used in their construction. In this paper, we show that, using some simple abstractions, a number of commonly-used tools used in the construction of succinct proof systems may be viewed as basic consequences of linear algebra over finite fields. We introduce notation which considerably simplifies these constructions and slowly build a toolkit of useful techniques that can be combined to create different protocols. We also show a simple 'probabilistic calculus' which specifies how to combine these tools and bounds on their resulting security. To show the power of these abstractions and toolkit, we give a short proof of the security of the FRI protocol. Along the way, we discuss some natural generalizations of protocols in the literature and propose a conjecture related to proximity testing using linear error-correcting codes that is of independent interest.

Introduction

Succinct proofs and arguments play an important role in ensuring privacy and integrity of computation, with many applications in blockchains among other fields. To understand succinct proofs, we may contrast them with 'traditional' computational proofs. Traditional computational proofs may be viewed as a certificate that a certain computation was performed correctly. Succinct proofs, compared to computational ones, make the following tradeoff: we allow some very small probability of error that the proof incorrectly verifies (even though the computation was done incorrectly) in exchange for a short, easy to verify certificate—often much shorter and faster to verify than the corresponding traditional proof would have been.

Historically, succinct proofs leverage interaction and randomness to achieve their stated goal. Related to this idea of succinct proofs are the succinct noninteractive arguments of knowledge (SNARKs, for short), which are of growing practical interest [BCCT13,BSCG+13, BBB+18, Nit20, Tha22]. Common to both of these approaches is the use of randomness to reduce complicated statements to simpler ones, which is the focus of this work. Note

that practical implementations of these protocols often involve careful considerations around communication models or cryptographic assumptions, which we elide here.

Our goal in this work is to provide a minimal framework for understanding the tools used in many protocols involving random reductions. To do this, we mostly limit ourselves to linear algebra over finite fields and some basic probability theory. These tools are sufficient to explain a surprising number of results and checks used throughout much of the literature.

Other work. Other authors have noted that, indeed, some of the reductions may be generalized to any number of other domains. For example, a recent line of work, exemplified by [BCS21], [RZ21], and [KP22] provided reductions showing that some important checks in the literature may be generalized not just to linear-algebraic, but more broadly to module-theoretic settings. In order to keep the exposition light, we focus only on the linear algebraic setting, but note that many results here may be naturally extended to the broader case, though we do not do so here.

This work. In this paper we show that many of the tools and 'checks' used in succinct proofs can be viewed as a consequence of basic facts from linear algebra over finite fields. This perspective also suggests some generalizations. For example, one common operation in succinct proof systems is to check if a number of vectors all lie in some subspace, say V. It is possible to show that, with high probability, all vectors lie in this subspace V only when a uniformly random linear combination of the vectors lies in V. In this sense, we have reduced a potentially 'very large' problem (of checking that each vector lies in V) to a much smaller problem (of checking a single vector lying in V) using randomness; many protocols rely on this (and similar) reductions. We show that the 'uniform random linear combination' may be replaced with a more general construction: a uniformly random row of a generator matrix for a linear code with large distance.

1 Preliminaries and notation

In this section, we discuss some basic facts from linear algebra, error correcting codes, and the conventions used in this paper. A reader that is very comfortable with linear algebra and the basics of error correcting codes should feel free to skim this material to understand the notation used in this paper and proceed directly to §1.4.

1.1 Linear algebra

In this paper, we will take the linear-algebraic standard for notation (versus, *e.g.*, some standards in coding theory). In particular, all vectors are column vectors (unless otherwise specified) and matrix-vector products are written in that order. (We can then interpret matrix-vector products as linear combinations of the *columns* of the matrix, rather than the rows.) The presentation here is not intended to be an introduction to linear algebra—for that, we refer the reader to, *e.g.*, [Axl14]—but is simply meant to be a refresher.

Vectors and matrices. Given a finite field $\mathbf{F}$ we write $\mathbf{F}^n$ for the set of n-vectors with elements in $\mathbf{F}$ and write $\mathbf{F}^{m\times n}$ for the set of matrices with m rows and n columns with elements in $\mathbf{F}$. For example, we may have $x \in \mathbf{F}^n$ and $A \in \mathbf{F}^{m\times n}$, then

$$Ax = \sum_{j=1}^{n} x_j a_j,$$

where $a_j \in \mathbf{F}^m$ is the jth column of A, which means that Ax is an m-vector. A common view will be to look at one specific element of the resulting vector Ax. For example, we may write the ith element as

$$(Ax)_i = \tilde{a}_i^T x,$$

where $\tilde{a}_i$ denotes the ith row of A (viewed as a column vector) and $\tilde{a}_i^T$ is the transpose of $\tilde{a}_i$ (which is a row vector). We will identify the n-by-1 matrices with the n-vectors, when the meaning is clear.

Vector spaces and subspaces. Any subset $V \subseteq \mathbf{F}^n$ which is itself a vector space (*i.e.*, V is closed under linear combinations of its elements) is called a *subspace*. (If $V' \subseteq V$ and V' is a vector space, we say that V' is a subspace of V.) Examples of vector subspaces are, of course, $\mathbf{F}^n$ or the singleton $\{0\}$. Another important example is, given a vector $x \in \mathbf{F}^n$, the ray

$$\{\alpha x \mid \alpha \in \mathbf{F}\}$$

is a subspace. Some useful and important operations on vector spaces are sums and intersections. In particular, if $U, V \subseteq \mathbf{F}^n$ are both subspaces, then their sum (sometimes called their Minkowski sum), defined

$$U + V = \{u + v \mid u \in U,\ v \in V\}, \tag{1}$$

is also a subspace. (We purposefully overload the $+$ operation for convenience.) Additionally, their intersection $U \cap V$ is also a subspace.

Range and nullspace. Every matrix $A \in \mathbf{F}^{m\times n}$ defines two vector subspaces. The first is its *range* (sometimes called the image of A) defined as

$$\mathcal{R}(A) = \{Ax \mid x \in \mathbf{F}^n\}.$$

From its definition, $\mathcal{R}(A) \subseteq \mathbf{F}^m$. In English: the range of a matrix A is the set of all vectors that are linear combinations of the columns of A. The second vector space is its *nullspace* (sometimes called its kernel) written

$$\mathcal{N}(A) = \{x \in \mathbf{F}^n \mid Ax = 0\}.$$

This is the set of vectors which are mapped to zero under the action of A. It is not hard to verify that both $\mathcal{R}(A)$ and $\mathcal{N}(A)$ are vector spaces (subspaces of $\mathbf{F}^m$). A matrix A is

said to be *injective* if $\mathcal{N}(A) = \{0\}$ since, if $y = Ax$ for some $x \in \mathbf{F}^n$, then this vector x is unique. Similarly, we say that a set of n-vectors $\{a_i\}$ is *linearly independent* if, viewed as the columns of a matrix

$$A = \begin{bmatrix} a_1 & a_2 & \dots & a_n \end{bmatrix},$$

then the nullspace of A is only the zero vector. In other words, we say the $\{a_i\}$ are linearly independent if every nonzero linear combination of the vectors a_i is nonzero.

Representations of subspaces. A basic fact from linear algebra is that every subspace $V \subseteq \mathbf{F}^m$ can be written as the range of some injective matrix $A \in \mathbf{F}^{m \times n}$,

$$V = \mathcal{R}(A),$$

where n is called the *dimension* of the subspace V. (The columns of A are called a *basis* for V.) Since A is injective, we have that $m \geq n$. Similarly, the subspace V can be written as the nullspace of some operator $C \in \mathbf{F}^{k \times m}$ such that

$$V = \mathcal{N}(C),$$

and $k = m - n$. This matrix is sometimes called the *parity check matrix*, which is the terminology we adopt here. Another way of stating this definition of C is:

$$x \in V \quad \text{if, and only if,} \quad Cx = 0. \tag{2}$$

Polynomial evaluations. One particular example we will use constantly is the vector space consisting of the evaluations of a polynomial of small degree. In particular, let $\{\alpha_i\} \subseteq \mathbf{F}$, for $i = 1, \dots, m$, be some evaluation points and $n \geq 1$ be some given number, then the set of polynomials of degree less than or equal to $n - 1$, evaluated at the points α_i, forms a vector space V since it can be written as the range of the matrix

$$A = \begin{bmatrix} 1 & \alpha_1 & \alpha_1^2 & \dots & \alpha_1^{n-1} \\ 1 & \alpha_2 & \alpha_2^2 & \dots & \alpha_2^{n-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \alpha_m & \alpha_m^2 & \dots & \alpha_m^{n-1} \end{bmatrix}. \tag{3}$$

(This matrix is called the *Vandermonde matrix* of degree $n - 1$ for the evaluation points α_i.) To see this, note that every polynomial of degree at most $n - 1$, say $f : \mathbf{F} \to \mathbf{F}$, can be written as

$$f(\beta) = x_1 + x_2\beta + x_3\beta^2 + \cdots + x_n\beta^{n-1},$$

where $x \in \mathbf{F}^n$. So, we may write

$$f(\alpha_i) = x_1 + x_2\alpha_i + \cdots + x_n\alpha_i^{n-1} = (Ax)_i,$$

for any polynomial of degree at most $n - 1$. Because of this, the set of evaluations of all possible low-degree polynomials f, written as vectors $(f(\alpha_1), \dots, f(\alpha_m))$, is just

$$V = \mathcal{R}(A).$$

Finally, because we know that the range of any matrix is a subspace and since V is the range of the matrix A, then the set V must also be a subspace.

Direct sums. If $V \subseteq \mathbf{F}^n$ is a subspace and $T, U \subseteq V$ are subspaces of V satisfying the property that $T \cap U = \{0\}$ and $V = T + U$, where

$$T + U = \{y + z \mid y \in T,\ z \in U\},$$

then we say that V is the *direct sum* [Axl14, §1] of T and U. We write this as $V = T \oplus U$, for shorthand. The interpretation of this is that, if $x \in V$, then it can be written uniquely as

$$x = y + z,$$

where $y \in T$ and $z \in U$. We may extend this definition to any number of subspaces, $V_k \subseteq V$ with $k = 1, \ldots, s$, and in this case we write

$$V = V_1 \oplus V_2 \oplus \cdots \oplus V_s,$$

if $V = V_1 + \cdots + V_s$ and $V_k \cap V_l = \{0\}$ for every $k \neq l$ with $k, l = 1, \ldots, s$.

1.2 Linear error correcting codes

In this section, we introduce some basic definitions from linear error correcting codes, along with some examples of codes that we use throughout. Though we will not use any deep results from error correcting codes, other than the definition of distance, we refer readers to the excellent series of lecture notes [Woo22] for more.

1.2.1 Weight and distance

The *weight* of a vector $x \in \mathbf{F}^n$, defined

$$\|x\|_0 = |\{i \mid x_i \neq 0\}|,$$

is the number of nonzero entries of a vector x. (This is often referred to as the Hamming weight, but we simply use the term weight for this paper.) We write this as $\|\cdot\|_0$, as this is sometimes called the ℓ_0-'norm', since it satisfies the triangle inequality

$$\|x + y\|_0 \leq \|x\|_0 + \|y\|_0,$$

for any $y \in \mathbf{F}^n$. It also satisfies definiteness in that $\|x\|_0 = 0$ if and only if $x = 0$, and satisfies 0-homogeneity in that, for any $\alpha \in \mathbf{F}$ with $\alpha \neq 0$,

$$\|\alpha x\|_0 = \|x\|_0.$$

(This 'norm' is not a true norm as norms must usually be 1-homogeneous.) Using this 'norm', the *distance* between two vectors x and y can then be written as $\|x - y\|_0$.

Notation. We overload this norm notation for convenience to also work over sets: given a set $S \subseteq \mathbf{F}^n$, we write

$$\|S\|_0 = \min_{x \in S} \|x\|_0,$$

such that the weight of a set S is the minimum weight of any vector in the set. This notation is quite convenient: writing, as in (1),

$$x - S = \{x - y \mid y \in S\},$$

we may then view

$$\|x - S\|_0 = \min_{y \in S} \|x - y\|_0,$$

as the minimum distance between x and any vector in the set S.

1.2.2 Linear codes

A *linear code* is defined as a matrix $G \in \mathbf{F}^{m \times n}$. (Technically, many matrices G may generate the same code, which is usually defined as $\mathcal{R}(G)$, but we ignore the distinction here.) We say that n is the *message length* and m is the *block length* of the code given by G. This is called a linear code since we can take a message $x \in \mathbf{F}^n$, which is simply an n-vector over the field $\mathbf{F}$, and *encode* it by applying the matrix G to get a length-m codeword Gx. It is linear since the codeword corresponding to any linear combination of messages is just the same linear combination of the individual codewords.

Distance. The main definition that we will use throughout this paper is the *distance* of the code G, defined

$$d = \min_{x \in \mathbf{F}^n \setminus \{0\}} \|Gx\|_0,$$

where $\|z\|_0$ denotes the number of nonzero entries in $z \in \mathbf{F}^m$. This is called the distance since any two distinct messages $x, y \in \mathbf{F}^n$ with $x \neq y$ will differ in at least d places after being encoded; *i.e.*,

$$\|Gx - Gy\|_0 = \|G(x - y)\|_0 \geq d,$$

since $x - y \neq 0$.

Note that G has linearly dependent columns (*i.e.*, nontrivial nullspace) if, and only if, the distance $d = 0$. In other words, if the distance $d > 0$ then G is injective; this implies that the distance d can be positive only when $m \geq n$.

Discussion. At the highest level, we may view codes with large distance d as those which encode 'errors' in the original message in many places. More specifically, if we have some expected message, say $x \in \mathbf{F}^n$, but, instead, receive a corrupted message $x + \delta$ where $\delta \in \mathbf{F}^n$ has a small (but nonzero) number of corruptions, we know that

$$\|G(x + \delta) - Gx\|_0 = \|G\delta\|_0 \geq d.$$

So if d is roughly of the size of m (where m is the size of the codeword) then, if we expect message x, but instead receive $\tilde{x} = x + \delta$, it only suffices to check that Gx (the encoded expected message) differs from $G\tilde{x}$ (the encoded received message) in a few entries, to 'catch' the fact that $\tilde{x} \neq x$. This rather simple fact, and its consequences, will be the basis for the entire remainder of the paper.

1.3 Examples of linear codes

There are a number of important linear codes that are used in the broader literature. We show a few examples, along with some properties, here.

1.3.1 Repeated code

A very simple (and silly) family of codes are the k-repeated codes, given by the matrix $G \in \mathbf{F}^{kn \times n}$, defined

$$G = \begin{bmatrix} I_n \\ \vdots \\ I_n \end{bmatrix},$$

where $I_n \in \mathbf{F}^{n \times n}$ is the identity matrix of size $n \times n$. This code simply takes some message x of length n, repeats it k times, and stacks the result into a vector of size kn; *i.e.*, given some vector $x \in \mathbf{F}^n$:

$$Gx = \begin{bmatrix} x \\ \vdots \\ x \end{bmatrix},$$

where the vector x is stacked k times

Distance. This code has distance $d = k$, which is easy to see from its definition.

1.3.2 Reed–Solomon codes

A more interesting (and common) choice of codes are the *Reed–Solomon codes*. Assuming a message of length n and given some block size $m \geq n$, we may pick any subset of evaluation points $\{\alpha_1, \ldots, \alpha_m\} \subseteq \mathbf{F}$. We can then define the matrix as $G \in \mathbf{F}^{m \times n}$ in the following way:

$$G_{ij} = \alpha_i^{j-1},$$

for $i = 1, \ldots, m$, and $j = 1, \ldots, n$. (The definition here is exactly the definition of the Vandemonde matrix given in (3), written in index notation.)

Distance. Since a nonzero degree d polynomial has at most d roots (over any field $\mathbf{F}$) then the distance of this code is at least $m - n + 1$ since

$$(Gx)_i = \sum_{j=1}^{n} x_j \alpha_i^{j-1},$$

is a degree at most $n-1$ polynomial with coefficients x_j, evaluated at each of the α_i. It is not hard to construct a polynomial of degree $n-1$ which has exactly $m-n+1$ zeros when evaluated over the points $\alpha_1, \ldots, \alpha_m$, which makes the distance of this code exactly equal to $m - n + 1$.

1.3.3 Hadamard code

Another common code is the *Hadamard code*, which is defined by a matrix $G \in \mathbf{F}^{m \times n}$ whose rows contain all possible tuples from $\mathbf{F}^n$. (Here, then, $m = |\mathbf{F}|^n$.) Drawing a row uniformly at random corresponds to drawing a uniformly random tuple from $\mathbf{F}^n$.

Distance. The distance of this code is $d = |\mathbf{F}|^n - |\mathbf{F}|^{n-1}$. To see this, consider any row i of G, which we write as y. Then, given a message $x \neq 0$, we have that at least one index, say j, has $x_j \neq 0$. So, if the ith symbol of Gx is zero, we have

$$(Gx)_i = y_j x_j + \tilde{y}^T \tilde{x} = 0,$$

where $\tilde{x}$ is the vector x with entry j removed, and similarly for y. We can write this as

$$y_j = -x_j^{-1}(\tilde{y}^T \tilde{x}).$$

Note that any choice of $\tilde{y}$ fixes y_j from the above equation. Since there are $|\mathbf{F}|^{n-1}$ possible options for $\tilde{y}$ satisfying this condition, then there are at most $|\mathbf{F}|^{n-1}$ possible symbols of Gx that are zero, or, equivalently, at least $|\mathbf{F}|^n - |\mathbf{F}|^{n-1}$ symbols that are nonzero in Gx, whenever $x \neq 0$.

1.4 Probabilistic implications

It will be useful to define some basic notation which will clean up the proofs that follow. In particular, this notation can be seen as compact way of expressing logical statements (such as implications) in a probabilistic setting, where there is some probability that a statement is not true. Even in this case, we will show that analogues of the basic rules of logic still hold.

Implications. Given two statements P_r and $Q_{r'}$ (in other words, two Boolean functions taking on values 0 or 1) that depend on random variables r and r' drawn from some set, we write

$$P_r \underset{p}{\implies} Q_{r'},$$

if $\Pr(P_r \wedge \neg Q_{r'}) \leq p$ over randomness r and r'; *i.e.*, if the probability that P_r is true, yet $Q_{r'}$ is not, is no larger than some value p. In general, we make no assumptions about the distribution of r and r' except that it should be explicit in the text. Almost universally, in the remainder of this work, we will either have $r = r'$ or r and r' independently and uniformly drawn from some set. When it is clear from context what the randomness is over, we will sometimes write this as P_r *implies* $Q_{r'}$ *with error (at most)* p.

The basic idea behind this notation is to think of p as the probability that the statement is 'wrong'. If $p = 0$, then this reduces exactly to the usual definition in propositional logic, while, if p is very small, we can think of the statements as being 'almost equivalent' under the randomness. In general, this relation will satisfy many similar rules to those of propositional logic, with additional 'error' terms that satisfy some basic rules we discuss below.

Chaining implications. Given three statements with

$$P_r \underset{p}{\implies} Q_{r'}, \quad \text{and} \quad Q_{r'} \underset{p'}{\implies} T_{r''},$$

over randomness r, r', and r'', then

$$P_r \underset{p+p'}{\implies} T_{r''}.$$

This is not hard to see from the union bound, but we provide the easy (if verbose) proof in appendix A. Note that, since 'normal' implication can be written as $\underset{0}{\implies}$, then this bound says, if

$$P_r \underset{p}{\implies} Q_{r'}, \quad \text{and} \quad Q_{r'} \text{ implies } T_{r''},$$

then $P_r \underset{p}{\implies} T_{r''}$. (In the 'usual' logical implications, we will have that $r' = r''$.)

Contrapositives. It is also similarly easy to see that, if

$$P_r \underset{p}{\implies} Q_r,$$

then

$$\neg Q_r \underset{p}{\implies} \neg P_r,$$

from the definition.

Conjunctions. Another interesting point is, given some 'deterministic' claim Q (that does not depend on randomness; it is either true or false), and the following implications

$$P_r \underset{p}{\implies} Q, \quad \text{and} \quad T_{r'} \underset{p'}{\implies} Q,$$

then, if r and r' are independent, we have

$$P_r,\ T_{r'} \underset{pp'}{\implies} Q.$$

This follows essentially from the definition of independence and the probabilistic implications above. In English: if we have two claims we can independently verify, each of which implies Q with low probability of error p and p', then verifying both (with independent randomness) must imply Q with much lower probability of error pp'. A special case of this is: repeating a check with independent and identical randomness decreases the probability of failure of the check from p to p^2.

Discussion. While the notation (and some of the basic implications) described here look simple, they will reduce the complexity of the discussions by allowing us to talk about, and compose, statements in a very compact way, without requiring additional overhead. We will make constant use of the rules described here throughout the remainder of this paper. Interestingly, as far as the authors know, this subject has only been explored in the context of data analysis and inference, as a special case of fuzzy logic implications [BJ08], but, to the authors' knowledge, little of it has been used in the general proof setting (along with its implications).

2 The structure of checks

This section explains the general model and assumptions that go into many of the tools used in succinct proof systems, along with explaining how randomness is useful in making these tools practical. Those familiar with the structure of succinct proof systems should feel free to skim this section and proceed to the next section directly.

We will start this section with a very simple example, known as the zero check and its variation, the sparsity check, both of which we analyze later in more detail. This will then help motivate the remainder of the section and show how claims that are expensive to verify may be successively 'reduced' to smaller claims that are cheaper to verify.

2.1 The zero check

In the examples that follow, we seek to answer the following question: given two vectors x and y (say, in $\mathbf{F}^n$), are the two vectors equal? Of course, we may verify whether $x = y$ by checking if each of their n elements are equal. Indeed, this is the best we can hope to do if we must be absolutely certain whether $x = y$, yet only have access to x and y by querying individual entries of each vector. If n is large—in many practical cases, n is on the order of 2^{20} or so—this could prove to be an expensive procedure if fetching each element incurs some reasonable cost.

2.1.1 Sparse checks

One 'relaxation' of the above question is to check if, instead, x is 'close to' y. That is, we would like to guarantee that

$$\|x - y\|_0 \leq q, \tag{4}$$

for some threshold $q \geq 0$, set ahead of time. From before, if we must be absolutely certain that x and y are no more than q apart from each other, we must check that at least $n - q$ entries of x and y are equal. (We recommend the reader convince themselves of this before proceeding.)

In practice, though, we rarely care that a statement is perfectly true. For example, if a statement is false with probability no more than 2^{-100}, it might as well be true for all intents and purposes. Of course, this assumes that the probability is measured over some 'reasonable' notion of randomness, but we will see that this can be reasonable achieved in what follows.

Probabilistic sparse checks. Perhaps the simplest reasonable 'check' is to uniformly randomly draw indices $r \sim \{1, \ldots, n\}$ and verify that $x_r = y_r$ at each of these indices. If the desired condition (4) is not true, that is, if $\|x - y\|_0 \geq q + 1$, then the probability that a uniformly randomly chosen index lands on one of the entries with $x_r \neq y_r$ is at least

$$1 - \frac{q+1}{n}.$$

If we repeat this procedure ℓ times, uniformly and independently drawing indices from $1, \ldots, n$, the probability that we never 'catch' at least one of the indices for which $x_r \neq y_r$ is easily bounded from above by

$$\left(1 - \frac{q+1}{n}\right)^{\ell}.$$

To guarantee a probability of failure no more than p, then, it suffices to repeat the experiment

$$\ell \geq \frac{n}{q+1} \log\left(\frac{1}{p}\right)$$

times. (We have used the fact that $\log(1-x) \leq -x$ in this bound.) If n is large (say $n = 2^{20}$) and $q = n/10$, yet we wish the probability of failure to be no more than 2^{-100} then, repeating the experiment

$$\ell \approx 700$$

times suffices. In other words, only about 700 queries to elements of x and y are needed to verify the claim. On the other hand, checking that this condition holds exactly requires checking that $n - q$ entries are equal, which is still on the order of about a million queries. While this particular observation is not 'deep' in any meaningful sense, the basic idea is generalized to a much broader setting in §3.2.

2.1.2 Models and the 'exact' zero check

In the second setting, we wish to check whether x is exactly equal to y with high probability. Checking exact equality with high probability using the procedure described previously essentially boils down to setting the threshold $q = 0$, which means that the number of queries

is roughly n. (Indeed, a simple argument shows that, to guarantee any probability of failure smaller than 1, $p < 1$, we always need on the order of n queries.) This means that a new interaction model is required if we want to do better than n queries, which we cover in a brief aside below.

Direct access model. As a baseline, we can view the communication model discussed up until this point as a simple *direct access model*. In that model, x and y are vectors stored in some drive handed to us, say. We then pay some fixed cost for each query, which accesses a single element of x or y. The drive is assumed to be 'truthful' in that it simply relays information about x and y as queried. (Such a guarantee is not obvious: for example, one could easily imagine a malicious 'smart' drive that attempts to adapt its responses based on the previous queries we made.)

Coding model. We can propose a different model by slightly generalizing the direct access model, which we call the *coding model*. In this model, instead, we are handed a black box. This black box has a fixed code matrix $G \in \mathbf{F}^{m \times n}$, known to us, along with some message $x \in \mathbf{F}^n$, unknown to us. We then are allowed to query some index i for symbol $(Gx)_i$ from the encoded message Gx, paying a fixed cost for each query performed. The direct access model is the special case where the code matrix G is the identity $G = I$, such that querying the ith symbol of Gx is the same as querying the ith symbol from the message x.

Model discussion. We note that these models assume that the queries are answered truthfully; *i.e.*, that the black boxes being handed to us answer queries in a way consistent with their definitions above. This, of course, seems like a very difficult restriction. Indeed, if a stranger (or, more specifically, a paper author) hands us such a black box on the street with a promise that it satisfies these definitions, we should be extremely skeptical of these promises and the authors more generally. Luckily, in many practical cases, we can replace black boxes with cryptographic protocols that have certain guarantees. For example, the direct access model can be replaced with a Merkle commitment [KL21, §6.6.2], while the coding model can be replaced with any number of possible polynomial commitments [KZG10] (when the matrix G corresponds to a Reed–Solomon code), inner product commitments [BCC$^+$16] (for general matrices G), among many others.

Probabilistic zero check. Now that we have defined the coding model, we will show that, surprisingly, we can construct a simple procedure in this model to check if $x \in \mathbf{F}^n$ is equal to $y \in \mathbf{F}^n$ that succeeds with high probability, assuming that the code matrix G has relatively large distance, using only a single query.

To see this, let $G \in \mathbf{F}^{m \times n}$ be a m-by-n matrix with distance d. The procedure is as follows:

1. Uniformly randomly sample an index r from $1, \ldots, m$
2. Query the rth symbol of Gx and Gy to receive $(Gx)_r$ and $(Gy)_r$

3. Verify that $(Gx)_r = (Gy)_r$

If $x = y$, then this procedure correctly verifies this fact. On the other hand, if $x \neq y$, the probability that $(Gx)_r = (Gy)_r$ is very small. Since

$$(Gx)_r - (Gy)_r = (G(x - y))_r,$$

and $x - y \neq 0$, then $G(x - y)$ has at least d nonzero entries, by the definition of distance. The probability that a uniformly randomly chosen index r lands in one of those entries is at least d/m, or, conversely, the probability that r 'misses' one of these nonzero entries is at most $1 - d/m$. If d is very close to m, then this probability is nearly zero. To make concrete what 'nearly zero' means, consider a Reed–Solomon code matrix (see §1.3.2) whose subset of evaluations is every element of $\mathbf{F}$ (such that $m = |\mathbf{F}|$). If $n = 2^{20}$ (as our previous example) and $|\mathbf{F}| = 2^{255} - 19$ (as in some standard protocols [BS23, §15.3.3]), then $d = m - n + 1$, so the probability that this procedure errs is no more than

$$1 - \frac{d}{m} \approx 2^{-235},$$

which is effectively zero for all practical purposes.

2.2 Succinctness and zero knowledge

The procedures above have two interesting properties. First, in a very general sense, we started with the task of checking whether two potentially very long vectors, x and y, are equal. By using some randomness, we have reduced this task to a much smaller one: checking whether a small number of values are equal to one another. This latter 'simple' claim, if true, lets us claim the 'global' statement that the two vectors x and y, each composed of many field elements (in fact, n of them), must be equal or 'close'. The second interesting point is that, in a certain sense, we learn very little about the vectors x and y (with a small modification, it is possible to ensure we actually learn practically nothing) other than the original fact we wished to verify.

We will focus almost universally on the first part: taking a large claim and reducing it to a much smaller one, such that it suffices only to verify the smaller claim, which, in turn, implies the original one with high probability. The second part, while interesting in its own right, is mostly avoided in the remainder of this text. We refer the reader to [Tha22] for (much) more.

Reductions. In a certain sense, we may view the above procedures in §2.1 as a type of 'lossy' reduction. The procedures begin with a certain very large claim, and this claim is reduced to a much simpler one that is easier to verify, but in doing so, we accept some probability of failure. (This probability of failure cannot be avoided [BS23, §20] and is essential in making the reduction 'easier' than the original problem.) Indeed, any reduction we present here can be split into three steps: first, we are handed some black box, we then

query the black box, and finally we verify that its result is consistent with some statement. Ideally, verifying the result in the last step is much easier than the original claim we wished to show; in some cases, though, it is only marginally easier. In these cases, we can then replace this last step with another reduction (and accept some additional error for doing so) that is also marginally easier, and so on, until the claim has been reduced to a very simple one, which also has a very small probability of error. This is the basic structure of a number of succinct proof systems, a few of which we generalize here.

Size of the generator. Another important point that could be raised is that, for example, the matrix in §2.1.2 is very large: the number of rows of G is $m = |\mathbf{F}|$, which is on the order of 2^{255}, far too large to fit in any computer. The important thing to keep in mind is that the whole matrix should never be constructed. The only way that the matrix G is 'accessed' is in finding a specific symbol, say r of the encoding, $(Gx)_r$, which means that it suffices to only construct the rth row of G, written $\tilde{g}_r^T$, which we can use to compute the symbol by noting that

$$(Gx)_r = \tilde{g}_r^T x.$$

Indeed, in many of the checks that follow, we only need the ability to query the inner product of the message x with a single row of the matrix G, even when this matrix is very large.

3 Standard checks

In the remainder of this paper, we will build up a library of such 'checks', beginning with the zero checks presented above. Using the probabilistic implications presented in §1.4, it will then be possible to compose these checks in a variety of ways to achieve certain outcomes. Along the way, we will give natural linear-algebraic generalizations of many checks and tools commonly used in succinct proof systems, along with some conjectures that greatly generalize a number of standard protocols.

3.1 Classic checks

In this section we focus on what we call the 'classic' checks: those which check exact inclusion, exact equality, and so on. This should be compared to the 'sparse checks' presented later in this section, which only seek to verify that vectors are 'close enough' to others. This is presented first as some of the tools used in the checks here are used in the later section.

For the remainder of the section, $G \in \mathbf{F}^{m \times n}$ will be a matrix representing a linear code with message length n and block length m. We will assume that the distance of this code is $d \geq 0$. It will sometimes be convenient to consider the rows of G, which we write as $\tilde{g}_r^T$ for the rth row.

3.1.1 Zero check

This is a rewriting of the zero check presented in §2.1.2, using the notation defined in §1.4. One particular reduction of interest is to note that checking that two vectors are equal is, roughly speaking, the same as checking if a single vector is nonzero. In particular, checking that $x = y$ is identical to checking that $x - y = 0$, and, using the linearity of G, as in the proof given in §2.1.2, it suffices only to check this latter claim.

Check. Given a linear code with generator $G \in \mathbf{F}^{m \times n}$ and distance d, we wish to show that $x = 0$ with high probability, by checking a much smaller claim. We can write the procedure specified in §2.1.2 compactly using this notation. For any $x \in \mathbf{F}^n$, the following is true

$$(Gx)_r = 0 \quad \underset{p}{\Longrightarrow} \quad x = 0, \tag{5}$$

where the randomness is over r, uniformly sampled from $\{1, \ldots, m\}$, and $p \leq 1 - d/m$. (The reverse direction is obviously always true.) Parsing the symbols in expression (5) carefully, it may be read as: if $(Gx)_r = 0$, for a uniformly randomly chosen r in $1, \ldots, m$, then the probability that $x \neq 0$ is no more than p. In other words, it suffices to check that a single random symbol of the encoding of x is zero to conclude that x is the zero vector, with high probability (at least $1 - p$). Note that this is exactly the claim we showed in §2.1.2, but the notation here is considerably more compact. As a useful exercise, note that the check (5) can also be written

$$\tilde{g}_r^T x = 0 \quad \underset{p}{\Longrightarrow} \quad x = 0, \tag{6}$$

where $\tilde{g}_r^T$ denotes the rth row of G.

Example. A special case of this relation, used in a number of zero knowledge protocols, is known as the *polynomial zero check* [Tha22, §2.2]. This check says that, given a polynomial $f : \mathbf{F} \to \mathbf{F}$ of degree no more than $n - 1$, if we uniformly randomly sample a point $r \in \mathbf{F}$ and find that $f(r) = 0$, then, with probability at least $1 - (n-1)/|\mathbf{F}|$, we know that $f = 0$ everywhere. We may write this in the above notation as:

$$f(r) = 0 \quad \underset{p}{\Longrightarrow} \quad f = 0,$$

where $p \leq (n-1)/|\mathbf{F}|$. Note that this is the special case of the check provided in (5), where the coefficients of the polynomial f are represented as a vector $x \in \mathbf{F}^n$ such that

$$f(\beta) = x_1 + x_2\beta + \cdots + x_n\beta^{n-1},$$

for $\beta \in \mathbf{F}$, and the matrix $G \in \mathbf{F}^{m \times n}$ is the Reed–Solomon code matrix §1.3.2 where every field element in $\mathbf{F}$ is an evaluation point. This means that $m = |\mathbf{F}|$, and, using the probability bound in (5) we find

$$p \leq 1 - \frac{d}{m} = 1 - \frac{|\mathbf{F}| - n + 1}{|\mathbf{F}|} = \frac{n-1}{|\mathbf{F}|},$$

where the first equality uses the distance of a Reed–Solomon code $d = m - n + 1$. This matches the bound of the original polynomial zero check (*cf.*, [Tha22, §2.1]) exactly, as expected.

3.1.2 Matrix zero check

We may check that a matrix $X \in \mathbf{F}^{k \times n}$ is zero by reducing the claim to checking that a single k-vector is zero.

Check. Let X be an $k \times n$ matrix, with columns $x_1, \ldots, x_k$. We may then check that X is zero by noting that

$$G_{r1}x_1 + \cdots + G_{rn}x_n = 0 \quad \underset{p}{\Longrightarrow} \quad X = 0, \tag{7}$$

where the randomness is again over r, uniformly sampled from $\{1, \ldots, m\}$, and we have that $p = 1 - d/m$. We may view the check (7) as uniformly randomly drawing a row of G, say $\tilde{g}_r^T$, using this row to take a linear combination of the columns of X, and checking that the result is zero; we may write this as,

$$X\tilde{g}_r = 0 \quad \underset{p}{\Longrightarrow} \quad X = 0, \tag{8}$$

where, again, $\tilde{g}_r$ is a uniformly randomly selected row of G, viewed as a column vector, and $p \leq 1 - d/m$.

Proof. The proof follows by reducing this case to the previously presented zero check. Let $\tilde{x}_1^T, \ldots, \tilde{x}_k^T$ be the k rows of the matrix X:

$$X = \begin{bmatrix} \tilde{x}_1^T \\ \vdots \\ \tilde{x}_k^T \end{bmatrix}.$$

Now, let's say that for uniformly randomly chosen r from $1, \ldots, m$, we have that the left-hand-side of the claim (7) is true,

$$G_{r1}x_1 + \cdots + G_{rn}x_n = 0, \tag{9}$$

then, this is the same as saying that

$$(G\tilde{x}_1)_r = 0, \ \ldots, \ (G\tilde{x}_k)_r = 0.$$

(All we have done is reinterpret the sum (9) in terms of the rows of X, as opposed to the columns.) From before, we know that each of these has

$$(G\tilde{x}_j)_r = 0 \quad \underset{p}{\Longrightarrow} \quad \tilde{x}_j = 0,$$

where $p \leq 1 - d/m$, for $j = 1, \ldots, k$. But then, if every row of X is zero, $\tilde{x}_j = 0$, this implies $X = 0$, which gives the result that

$$G_{r1}x_1 + \cdots + G_{rn}x_n = 0 \quad \underset{p}{\Longrightarrow} \quad X = 0,$$

with the same p as above, $p \leq 1 - d/m$.

Example. This reduction is very similar to a number of 'common' checks: for example, to check that a list of vectors is zero, many protocols [BCMS20, KST22] will take a uniformly random linear combination of these vectors and check that the result of this linear combination is zero. We note that this is exactly the special case of the check (7) when the matrix G is a Hadamard code matrix (see §1.3.3), in which case randomly drawing a row of G corresponds exactly to randomly drawing a uniform vector in $\mathbf{F}^n$ and so the check (7) corresponds to taking a uniformly random linear combination of the columns of X. In this case, we recover the 'standard' error bound:

$$p \leq 1 - \frac{d}{m} = 1 - \frac{|\mathbf{F}|^n - |\mathbf{F}|^{n-1}}{|\mathbf{F}|^n} = \frac{1}{|\mathbf{F}|},$$

where we have used the fact that the Hadamard code of message length n with field $\mathbf{F}$ has block size $m = |\mathbf{F}|^n$ and distance $d = |\mathbf{F}|^n - |\mathbf{F}|^{n-1}$, as shown in §1.3.3.

3.1.3 Reduced matrix zero check

The previous section §3.1.2 showed that we can reduce checking that a matrix is zero to checking that a single vector is zero. We also have, from §3.1.1, a way to reduce checking that a vector is zero to checking that a single field element is zero. The natural next step is to join the two checks together and therefore reduce checking that a matrix is zero to checking that a single field element is zero, with some additional error. Recalling the previous set up, we assume that we are interested in checking that some matrix $X \in \mathbf{F}^{k \times n}$ is zero.

Check. For this check, we introduce a second code matrix $G' \in \mathbf{F}^{m' \times k}$ (which will serve as our second check's code matrix) with distance $d' \geq 0$. The check can then be written:

$$\left(G' \left(\sum_{i=1}^{k} G_{ri} x_i \right) \right)_{r'} = 0 \quad \underset{p+p'}{\Longrightarrow} \quad X = 0,$$

where x_i denotes the ith column of X, while r is uniformly randomly drawn from $1, \ldots, m$ and r' is uniformly randomly drawn from $1, \ldots, m'$. Here, $p \leq 1 - d/m$ and $p' \leq 1 - d'/m'$ Another way of writing this check is, letting $\tilde{g}_r^T$ denote the rth row of G and $\tilde{g}_{r'}^{\prime T}$ denote the r'th row of G',

$$\tilde{g}_{r'}^{\prime T} X \tilde{g}_r = 0 \quad \underset{p+p'}{\Longrightarrow} \quad X = 0. \tag{10}$$

We will use this more compact statement to prove the claim.

Proof. The check is essentially also the proof of its correctness, using the notation and implications of §1.4, along with the proofs presented previously. Starting with the left hand side of (10)

$$\tilde{g}_{r'}'^{T}(X\tilde{g}_r) = 0 \quad \underset{p'}{\Longrightarrow} \quad X\tilde{g}_r = 0,$$

using the basic zero check (6), where $p' \leq 1 - d'/m'$ and r' uniformly randomly selected from $1, \ldots, m'$. Note that this statement is true for any r. Now, from the matrix check (8), we know that

$$X\tilde{g}_r \quad \underset{p}{\Longrightarrow} \quad X = 0,$$

where r is uniformly randomly selected from $1, \ldots, m$. Using the chained implications of §1.4, we therefore have that

$$\tilde{g}_{r'}'^{T}(X\tilde{g}_r) = 0 \quad \underset{p+p'}{\Longrightarrow} \quad X = 0.$$

where r and r' are uniformly and independently chosen from $1, \ldots, m$ and $1, \ldots, m'$, respectively, while p and p' are as defined above, which is what we wished to prove.

Discussion. We may view the check above as a type of 'generalized Schwartz–Zippel' lemma [Tha22, §3.4], for arbitrary linear codes. Indeed, when the matrices G and G' are Reed–Solomon code matrices and every field element is an evaluation point, we may interpret the check as exactly evaluating a bivariate polynomial, with coefficients X, at two uniformly randomly chosen points in $\mathbf{F}$. In this case, $m = m' = |\mathbf{F}|$, and, since the matrix $X \in \mathbf{F}^{k \times n}$, then this means that $n-1$ is the max degree of the first variable and $k-1$ is the max degree of the second. The bound here then says that, if the polynomial is nonzero (*i.e.*, if $X \neq 0$) then the probability that the polynomial, evaluated at a uniformly randomly chosen point in $\mathbf{F}^2$, evaluates to zero is no more than

$$p + p' \leq \left(1 - \frac{d}{m}\right) + \left(1 - \frac{d'}{m'}\right) = \frac{(n-1) + (k-1)}{|\mathbf{F}|},$$

exactly matching the bound of Schwartz–Zippel. Here, we have used the fact that $d = m - n + 1$ for Reed–Solomon codes (and similarly for d'). Surprisingly, it is not hard to show that this bound is loose (and therefore Schwartz–Zippel is, as well) except when the codes are trivial, and it is possible to do better by using the fact that the codes are linear. In this case, the bound can be improved by an additive factor of around $(1 - d/m)(1 - d'/m')$ which is very small when d/m and d'/m' are very close to 1. (See appendix B.3.)

Generalization. A natural question is, of course, can we generalize this procedure from vectors and matrices to some sort of higher-order mathematical structure? A natural idea would be to introduce tensors and so on, but there is a slightly simpler approach which we can explore via the *Kronecker product*. (See appendix B.1 for a definition.) In particular, we may identify the $k \times n$ matrix X with a (long) vector $x \in \mathbf{F}^{kn}$ by stacking the columns

of X. The procedure above, performed over X, is equivalent to running the following check over the vector x,

$$((G' \otimes G)x)_{\tilde{r}} = 0 \quad \underset{p}{\Longrightarrow} \quad x = 0,$$

where $G' \otimes G$ is the Kronecker product of G and G', and $\tilde{r}$ is uniformly randomly chosen from $1, \ldots, mm'$, while $p \leq 1 - dd'/mm'$. Indeed, this 'equivalence' gives us a tighter bound, which we show in appendix B.2, than the one derived above. Of course, given any higher order tensor of s dimensions, say, (n_j) for $j = 1, \ldots, s$, we may always 'stack' the tensor into a vector x of size $\prod_j n_j$ and apply a similar procedure, given codes G_j of dimension $\mathbf{F}^{m_j \times n_j}$ for each $j = 1, \ldots, s$. This leads to a 'general check'

$$((G_1 \otimes G_2 \otimes \cdots \otimes G_s)x)_r = 0 \underset{p}{\Longrightarrow} x = 0,$$

where r is uniformly sampled from $1, \ldots, \prod_j m_j$, while

$$p \leq 1 - \prod_{j=1}^{s} \frac{d_j}{m_j}.$$

If the matrices G_j are Vandemonde matrices, where every element in the field $\mathbf{F}$ is an evaluation point, then Schwartz–Zippel would imply that

$$p \leq \sum_{j=1}^{s} \left(1 - \frac{d_j}{m_j}\right) = \frac{\sum_{j=1}^{s}(n_j - 1)}{|\mathbf{F}|}.$$

Note that $n_j - 1$ denotes the degree of the jth variable, so the sum can be recognized as the total degree of the s-variate polynomial. On the other hand, this bound would imply,

$$p \leq 1 - \frac{\prod_{j=1}^{s}(|\mathbf{F}| - n_j + 1)}{|\mathbf{F}|^s} = 1 - \prod_{j=1}^{s}\left(1 - \frac{n_j - 1}{|\mathbf{F}|}\right).$$

These two values are, of course, very close if $n_j \ll |\mathbf{F}|$, but the latter is always a better bound unless $n_j = 1$ for all but one index j.

3.1.4 Vector subspace check

Finally, we can construct the most general form of the above checks by strengthening the procedures slightly. We may verify that the columns of a matrix $X \in \mathbf{F}^{k \times n}$ all belong to some vector subspace $V \subseteq \mathbf{F}^k$ by verifying that only one vector belongs in such a subspace.

Check. Using the same set up as the matrix zero check §3.1.2, let $x_i \in \mathbf{F}^k$ denote the ith column of the matrix X. The subspace check may be written as follows:

$$G_{r1}x_1 + \cdots + G_{rn}x_n \in V \quad \underset{p}{\Longrightarrow} \quad x_i \in V \;\text{ for }\; i = 1, \ldots, n, \tag{11}$$

where r is chosen uniformly at random from $1, \ldots, m$, and $p \leq 1 - d/m$. In a similar way to the previous, we may interpret this check as picking a random row of G, say $\tilde{g}_r^T$, uniformly over all rows, and then using this row as the coefficients of a linear combination of the columns of X. We may equivalently write this statement as

$$X\tilde{g}_r \in V \quad \underset{p}{\Longrightarrow} \quad X \in V_n, \tag{12}$$

where V_n is the set of $k \times n$ matrices whose columns lie in the vector space V.

Proof. The proof follows nearly immediately from the original matrix zero check. Since V is a vector space, we know there exists a parity check matrix $C \in \mathbf{F}^{s \times n}$ such that $y \in V$ if, and only if, $Cy = 0$, from (2). Given this, consider the left hand side of (11). If this is true, then

$$G_{r1}x_1 + \cdots + G_{rn}x_n \in V \quad \text{implies} \quad C(G_{r1}x_1 + \cdots + G_{rn}x_n) = 0.$$

We may rewrite the right hand side of this expression to:

$$G_{r1}(Cx_1) + \cdots + G_{rn}(Cx_n) = 0,$$

for any r. But, note that, if r is chosen uniformly at random from $1, \ldots, m$, then, from the matrix zero check §3.1.2, we know

$$G_{r1}(Cx_1) + \cdots + G_{rn}(Cx_n) = 0 \quad \underset{p}{\Longrightarrow} \quad Cx_i = 0, \text{ for } i = 1, \ldots, n,$$

where $p \leq 1 - d/m$. By definition of the parity check matrix C, the right hand side of this expression can be written

$$Cx_i = 0 \quad \text{implies} \quad x_i \in V,$$

for each $i = 1, \ldots, n$. Putting it all together gives the final result:

$$G_{r1}x_1 + \cdots + G_{rn}x_n \in V \quad \underset{p}{\Longrightarrow} \quad x_i \in V \;\; \text{for} \;\; i = 1, \ldots, n.$$

Discussion and extensions. We may view this check as a generalization of the matrix zero check, which is the special case of this check when the subspace $V = \{0\}$ (and therefore one possible parity check matrix for this subspace would be $C = I$). We may also ask if it is possible to extend the above check in a similar way to the reduced matrix zero check §3.1.3. Indeed, using the check (12) and the definition of the parity check matrix C, we can write the left hand side as

$$X\tilde{g}_r \in V \quad \text{if, and only if,} \quad CX\tilde{g}_r = 0,$$

where $\tilde{g}_r$ is the rth row of G, viewed as a column vector. Given a second code matrix $G' \in \mathbf{F}^{m' \times s}$ with distance d', we can reduce checking the right hand side of this statement, whether $CXg_r = 0$, to a 'simpler' check over a single field element, since:

$$\tilde{g}_{r'}^{\prime T} CX\tilde{g}_r = 0 \quad \underset{p'}{\Longrightarrow} \quad CXg_r = 0,$$

where $p' \leq 1 - d'/m'$ and $\tilde{g}_{r'}^T$ is a uniformly randomly sampled row of G'. (This follows from the zero check (6), using the code generated by G'.) The remainder of the check proceeds in the same way as (11) and has total error no more than $p + p'$. If it is easy to efficiently compute the matrix vector product $C^T \tilde{g}'_{r'}$ (it need not be, as this product may be large), then this check may be achieved with an inner product commitment.

3.2 Sparse checks

In this section, we present what we call the *sparse checks*. In comparison to the previous section §3.1, which mostly dealt with whether a vector is, say, all zero, or included in some vector space, this section deals with notions related to the sparsity of a vector or the distance from a vector to a vector space.

As in the previous section, we assume that $G \in \mathbf{F}^{m \times n}$ is a matrix generating a code with block size m, for messages of size n. We let $d \geq 0$ be the distance of this code, as defined previously.

3.2.1 Sparsity check

The first useful tool is the fact that we can check whether a given vector $x \in \mathbf{F}^k$ is *sparse*, that is, if most of its entries are equal to zero.

Check and proof. The simplest idea is to check that a randomly chosen entry of $x \in \mathbf{F}^k$ is zero: if this is true, then, with some probability, the vector x must be somewhat sparse. Writing this out, we have,

$$x_r = 0 \quad \underset{p}{\Longrightarrow} \quad \|x\|_0 \leq q,$$

with probability $p \leq 1 - (q+1)/k$, where r is uniformly randomly sampled from $1, \ldots, k$. To see this, note that if $x_r = 0$ yet, $\|x\|_0 > q$, then x has at least $q + 1$ nonzero entries. The probability we land on one of these is at least $(q+1)/k$ so the probability we land on none is no more than $1 - (q+1)/k$, as given above.

If the vector x is very sparse, then q is much smaller than k and this check is unlikely to succeed with high probability. Of course, we can simply repeat the check over a number of uniformly randomly-chosen indices $S \subseteq \{1, \ldots, k\}$ of some fixed size, then

$$x_S = 0 \quad \underset{p}{\Longrightarrow} \quad \|x\|_0 \leq q,$$

with $p \leq (1 - (q+1)/k)^{|S|}$, by a nearly identical argument. (The analysis can be made slightly tighter by noting that S does not have repeated indices, but we usually assume that $q \ll k$.)

Discussion. This is the same argument as the one presented in §2.1.1 except with the notation of §1.4. The main idea here is that we may take a check that succeeds with

relatively small probability and repeat it. Assuming that the samples are uniformly random and independent, this, of course, decreases the probability that the check fails.

Additionally, and for fun, we may combine this check with the previous zero check of §3.1.1. If $|S| = n$, then, using the definition of G at the beginning of this subsection,

$$(Gx_S)_r = 0 \quad \underset{p'}{\Longrightarrow} \quad x_S = 0 \quad \underset{p}{\Longrightarrow} \quad \|x\|_0 \leq q,$$

with $p' \leq 1 - d/m$ over a uniformly chosen r from $1, \ldots, m$, and S a uniform random subset of $\{1, \ldots, k\}$ and p is the same as above. This means that

$$(Gx_S)_r = 0 \quad \underset{p+p'}{\Longrightarrow} \quad \|x\|_0 \leq q,$$

where r is chosen from $1, \ldots, m$ and $S \subseteq \{1, \ldots n\}$ are both uniformly randomly chosen.

3.2.2 Matrix sparsity check

Another simple idea is to note that we can reduce the sparsity check for a list of vectors into a single sparsity check for a single vector. In particular, given a matrix $X \in \mathbf{F}^{k \times n}$, whose columns are vectors $x_i \in \mathbf{F}^k$ for $i = 1, \ldots, n$, the following check holds:

$$\left\| \sum_{i=1}^{k} G_{ri} x_i \right\|_0 \leq q \quad \underset{p}{\Longrightarrow} \quad X \text{ has at most } q \text{ nonzero rows},$$

where $p = (q+1)(1 - d/m)$. Note that this implies that each vector has $\|x_i\|_0 \leq q$ for $i = 1, \ldots, k$, but is, in general, a stronger statement. For example, this implies something roughly like: either all vectors are extremely sparse, or their nonzero entries are aligned.

Proof. We will give a simple bound on the probability that the inequality fails. In particular, given some X with at least $q + 1$ nonzero rows, we need to show that the probability that

$$\left\| \sum_{i=1}^{k} G_{ri} x_i \right\|_0 \leq q \tag{13}$$

is small, for a uniformly randomly chosen r (from $1, \ldots, m$). To see this, we will reduce the matrix sparsity check to the zero check provided in §3.1.1.

First, consider the expression in (13). We may remove rows of X to get some shorter matrix $\hat{X}$. Note that

$$\left\| \sum_{i=1}^{k} G_{ri} \hat{x}_i \right\|_0 \leq \left\| \sum_{i=1}^{k} G_{ri} x_i \right\|_0,$$

for any choice of r, where $\hat{x}_i$ denotes the ith column of the shorter matrix $\hat{X}$. So, the probability that the event (13) happens is at most the probability that the event

$$\left\| \sum_{i=1}^{k} G_{ri} \hat{x}_i \right\|_0 \leq q,$$

happens over a uniform choice of r from $1, \ldots, m$.

Now, let X have more than q nonzero rows. We may pick any $q+1$ nonzero rows of X to get the matrix $\hat{X} \in \mathbf{F}^{(q+1)\times k}$. Since every row of $\hat{X}$ is nonzero, if

$$\left\| \sum_{i=1}^{k} G_{ri} \hat{x}_i \right\|_0 \leq q,$$

then there is at least one row of $\hat{X}$, say $\tilde{x}_j^T$, which satisfies

$$(G\tilde{x}_j)_r = 0.$$

Of course, this happens with probability at most $1 - d/m$ by the zero check in (5). Finally, since there are $q+1$ possible choices of rows, we have that

$$p \leq (q+1)\left(1 - \frac{d}{m}\right),$$

by the union bound, as required.

Discussion. While likely not very useful by itself, we can think of this particular check as a type of 'lemma' which we can strengthen slightly to be part of the following check: the subspace distance check.

3.2.3 Subspace distance check

An important consequence of the sparsity check above is the ability to check that a matrix $X \in \mathbf{F}^{k\times n}$ has columns close to a vector subspace $V \subseteq \mathbf{F}^k$, by reducing this to checking only that a single vector is close to the subspace V.

Definitions. For this check, we have to introduce a few more definitions, similar in spirit to the previous ones. First, we will define the *distance* of a subspace V as

$$d' = \min_{x \in V \setminus \{0\}} \|x\|_0.$$

(Note that this is the definition of distance of a code G whenever $V = \mathcal{R}(G)$ and G is injective.) Now, we define what it means for a matrix X to be q-close to a vector subspace V. Let Y be any matrix whose columns lie in V, then we say X is *q-close to V* if there exists some matrix Y, meeting the above conditions, such that $X - Y$ has at most q nonzero rows.

Check. We can now state the *subspace distance check*: let $q < d'/2$ then

$$\left\| \sum_{i=1}^{n} G_{ri} x_i - V \right\|_0 \leq q \quad \underset{p}{\Longrightarrow} \quad X \text{ is at most } q\text{-far from } V, \tag{14}$$

where $p \leq (q+1)(1-d/m)$. We leave the general case of this check as an open conjecture, proving only the special case of $n = 2$ and $q < d'/4$ below, which we use later in this paper.

We note that this conjecture, in particular, generalizes some of the work from [AHIV17], which shows the special case where G is the Hadamard code (see §1.3.3) and $q < d'/3$, with the same probability of error. (This work was extended for a different code G, with slightly higher error probability, in [DP23].) In the special case that V is the subspace generated by Reed–Solomon codes, the bound is slightly improved to $q < d'/2$, but the probability p is slightly different [BCI+23]. We show the 'easy' part of the general proof in appendix C.1, but leave the second part of this proof as an open conjecture we call the *distance-preserving encoding conjecture*. See the appendix C.2 for more details.

Proof outline. As mentioned previously, we provide a proof of the special case where $n = 2$ and $q < d'/4$, which we use in the next section to generalize the FRI protocol and provide a simple proof of its security. Let

$$R = \{r \in \{1, \ldots, m\} \mid \|Xg_r - V\|_0 \leq q\}.$$

In other words, R is the set of indices at which the left hand side of the check (14) passes, and $|R|/m$ is the probability that the check indeed passes (independent of whether X is q-close to V). We will break the proof up into two cases. First, we will show that, if $|R| > m - d$ (equivalently, if the probability that the left hand side of the check is true is greater than $1 - d/m$), then for all $z \in \mathbf{F}^2$ we have

$$\|Xz - V\|_0 \leq 2q < d'/2.$$

That is, there is no linear combination of columns of X which can certify that the matrix X is more than $d'/2$-far from V. (The remaining case, if $|R| \leq m - d$, would mean we are done by definition.) We will then show that, if this is true, then the check given in (14) is correct with high probability. It is interesting that, at a very high level, the structure of the proof given here is similar to that of [AHIV17], but the mechanics of the actual proof are surprisingly different. We share more notes on the similarities (and differences) in appendix C.2.

Proof, part one. Since, by assumption, $|R| > m - d$, this means that at least two rows with indices $r, r' \in R$, are linearly independent. To see this, assume the contrary. Let $\tilde{g}_r$ be a nonzero row of G for $r \in R$, and, by assumption, for each other row, $\tilde{g}_{r'}$ with $r' \in R$, there exists some α, β, not both zero, such that $\alpha\tilde{g}_r + \beta\tilde{g}_{r'} = 0$. Let $x \in \mathbf{F}^2$ be a nonzero vector satisfying $\tilde{g}_r^T x = 0$, then we also have that $\tilde{g}_{r'}^T x = 0$. This would, in turn, imply that Gx is zero at more than $m - d$ indices, so the distance of G must be smaller than d, contradicting the assumption. Now, let $\tilde{g}_r$ and $\tilde{g}'_r$ be two linearly independent rows, with $r, r' \in R$, then

$$\|X(\alpha\tilde{g}_r + \beta\tilde{g}_{r'}) - V\|_0 \leq \|X\tilde{g}_r - V\|_0 + \|X\tilde{g}_{r'} - V\|_0 \leq 2q.$$

But since $\tilde{g}_r$ and $\tilde{g}_{r'}$ are linearly independent, then every vector $z \in \mathbf{F}^2$ can be written as a linear combination of these two vectors, $z = \alpha g_r + \beta g_{r'}$. This means that every vector $z \in \mathbf{F}^2$ has

$$\|Xz - V\|_0 \leq 2q,$$

whenever $|R| > m - d$.

Proof, part two. Now, if $|R| > m - d$, we will show that the current check reduces to the matrix sparsity check, which would show that $p \leq |R|/m \leq (q+1)(1 - d/m)$ if X is more than q-far from V, as required. From the assumption that $|R| > m - d$ and part one of the proof, we know that for any $z \in \mathbf{F}^2$, we have

$$\|Xz - V\|_0 \leq 2q. \tag{15}$$

Since we know that $q < d'/4$, then we may write the columns of X, x_1 and x_2, uniquely as

$$x_i = y_i + \xi_i,$$

where $y_i \in V$ and $\|\xi_i\|_0 \leq 2q$, using the fact any linear combination of the columns of X is a distance of at most $2q$ from V, see (15), and $2q < d'/2$, so we are in the unique decoding radius. Of course, note that, since $y_i \in V$, then any linear combination of these vectors is also in V. This means:

$$\|Xz - V\|_0 = \|z_1(y_1 + \xi_1) + z_2(y_2 + \xi_2) - V\|_0 = \|z_1\xi_1 + z_2\xi_2 - V\|.$$

So it suffices to consider only the latter quantity.

As before, let R denote the set of indices $1, \ldots, m$ over which the left hand side of the check is true (independent of the right hand side). Using the above, this is the same as defining R as

$$R = \{r \in \{1, \ldots, m\} \mid \|G_{r1}\xi_1 + G_{r2}\xi_2 - V\|_0 \leq q\}.$$

For each r, we have a corresponding unique $\bar{y}_r \in V$ such that

$$\|G_{r1}\xi_1 + G_{r2}\xi_2 - \bar{y}_r\|_0 \leq q, \tag{16}$$

as $q < d'/4$. Now, consider any pair $r, r' \in R$ and note that, by definition

$$\|G_{r1}\xi_1 + G_{r2}\xi_2 - \bar{y}_r\|_0 \leq q \quad \text{and} \quad \|G_{r'1}\xi_1 + G_{r'2}\xi_2 - \bar{y}_{r'}\|_0 \leq q.$$

Since scaling any vector by any constant does not increase the distance, we scale the first term by $G_{r'1}$ and the second by G_{r1} respectively. This gives:

$$\|G_{r'1}(G_{r1}\xi_1 + G_{r2}\xi_2 - \bar{y}_r)\|_0 \leq q \quad \text{and} \quad \|G_{r1}(G_{r'1}\xi_1 + G_{r'2}\xi_2 - \bar{y}_{r'})\|_0 \leq q.$$

Adding both of these terms and using the triangle inequality gives:

$$\|(G_{r'1}G_{r2} - G_{r1}G_{r'2})\xi_2 - (G_{r'1}\bar{y}_r - G_{r1}\bar{y}_{r'})\|_0 \leq 2q.$$

Applying the reverse triangle inequality to the terms above gives the following claim about $\bar{y}_r$ and $\bar{y}_{r'}$,

$$\|G_{r'1}\bar{y}_r - G_{r1}\bar{y}_{r'}\|_0 \leq 2q + \|(G_{r'1}G_{r2} - G_{r1}G_{r'2})\xi_2\|_0 \leq 4q < d'.$$

So, since V has distance d' we must have that the two vectors are equal; *i.e.*,

$$G_{r'1}\bar{y}_r = G_{r1}\bar{y}_{r'}.$$

Repeating this argument, but multiplying the inequalities with G_{r2} and $G_{r'2}$ instead, gives

$$G_{r'2}\bar{y}_r = G_{r2}\bar{y}_{r'}.$$

so, in particular, we may take a linear combination of these two equalities; *i.e.*, for any $s \in \mathbf{F}$ we have

$$(G_{r'1} + sG_{r'2})\bar{y}_r = (G_{r1} + sG_{r2})\bar{y}_{r'}. \tag{17}$$

Now, either for all $r \in R$ we have that $y_r = 0$ or there is at least one $\tilde{r} \in R$ such that $y_{\tilde{r}} \neq 0$. In the former case, this is the same as saying, from (16),

$$\|G_{r1}\xi_1 + G_{r2}\xi_2\|_0 \leq q.$$

From the matrix sparsity check presented in §3.2.2, if this is true, then, since $\xi_i = x_i - y_i$,

$$\|G_{r1}\xi_1 + G_{r2}\xi_2\|_0 \leq q \quad \underset{p}{\Longrightarrow} \quad X - Y \text{ has at most } q \text{ nonzero rows,}$$

where $Y \in \mathbf{F}^{k\times 2}$ is the matrix with columns y_1 and y_2. Equivalently, if $X - Y$ has more than q nonzero rows, then $|R|/m \leq (q+1)(1-d/m)$.

We will show that the remaining case, where some $\bar{y}_{\tilde{r}} \neq 0$, is impossible. Let $\tilde{r} \in R$ be an index such that $\bar{y}_{\tilde{r}} \neq 0$, and let $r \in R$ be any other index. Set $s \in \mathbf{F}$ in (17) such that

$$G_{\tilde{r}1} + sG_{\tilde{r}2} = 0.$$

Using the equality (17), every r must satisfy

$$(G_{r1} + sG_{r2})\bar{y}_{\tilde{r}} = (G_{\tilde{r}1} + sG_{\tilde{r}2})\bar{y}_r = 0,$$

but, since $\bar{y}_{\tilde{r}} \neq 0$, then $G_{r1} + sG_{r2} = 0$ for every $r \in R$. But, from the beginning of this section, $|R| > m - d$, so the distance of G is at most $m - |R| < d$, which contradicts our assumption.

4 A distance check protocol

We will use some of the checks presented above to present a protocol that successively reduces checking that some (potentially very large) vector is close to a subspace, to checking that an appropriately-chosen smaller vector is close to a vector subspace. This construction is essentially a linear-algebraic generalization of the Fast Reed–Solomon Interactive Oracle Proof of Proximity [BBHR18] (also called FRI) to vector spaces with certain recursive substructure. Along the way, we will give basic bounds on its probability of error and give a way of practically implementing the protocol, along with some basic query complexity bounds.

4.1 A basic reduction

In the simple protocol, we would like to reduce checking that a particular vector y is close to some vector subspace $V \subseteq \mathbf{F}^n$ to the fact that it is close to a (smaller) vector subspace $V' \subseteq \mathbf{F}^k$. We will start first with the 'exact inclusion' case (*i.e.*, checking that $y \in V$), which is easier to write, and then develop the 'distance check' (*i.e.*, whether $\|y - V\|_0$ is not too large) as a generalization.

Set up. We will assume that the vector space V has the following recursive substructure:

$$V = T_1 V' \oplus T_2 V' \oplus \cdots \oplus T_\ell V',$$

where $T_i \in \mathbf{F}^{n \times k}$ are some given matrices. Here, $\oplus$ denotes the fact that V is a direct sum of the indicated subspaces (as defined in §1.1), and we have defined, for $i = 1, \ldots, \ell$,

$$T_i V' = \{T_i x \mid x \in V'\},$$

which will be very convenient notation in what follows. This implies that any vector $y \in V$ can be written uniquely as

$$y = \sum_{i=1}^{\ell} T_i x_i,$$

where $x_i \in V$. (This is, roughly speaking, saying that V has some sort of recursive substructure; examples include the fact that $\mathbf{F}^{2k} \sim \mathbf{F}^k \times \mathbf{F}^k$ among others.) Similar to the previous section, we will assume that we are given some matrix $G \in \mathbf{F}^{m \times n}$ that generates a code with distance d, and a matrix $G' \in \mathbf{F}^{m' \times k}$ that generates a code with distance d'.

Aside. While this substructure may seem like a very strong requirement, we note that it always exists for any vector space V. If, for example, V is of even dimension, say $2k$ for some k, then there exists a matrix $A \in \mathbf{F}^{n \times 2k}$ such that $\mathcal{R}(A) = V$. In this case, we define the matrices $T_1, T_2 \in \mathbf{F}^{n \times k}$ such that

$$A = \begin{bmatrix} T_1 & T_2 \end{bmatrix},$$

and $V' = \mathbf{F}^k$. (That is, the matrix T_1 contains the first k columns of A, while the matrix T_2 contains the last k columns.) This would then show that

$$V = T_1 V' \oplus T_2 V'.$$

Note that we do not require the matrices T_i to be injective, so a similar argument would work for cases where the dimension of V is odd. Of course, in many cases, much more structure can be exploited, but this varies on a case-by-case basis.

4.1.1 Exact inclusion reduction

Indeed, we may reduce the claim of checking that $y \in V$ to the (hopefully easier) single claim of checking that a single element is in the smaller vector space V'. To do this, note that $y \in V$ if, and only if there exist $x_i \in V'$ such that

$$y = \sum_{i=1}^{\ell} T_i x_i. \tag{18}$$

In other words, to verify that $y \in V$, it suffices to be given $x_i \in \mathbf{F}^k$, verify that $x_i \in V'$ for each $i = 1, \ldots, \ell$, and verify that (18) holds. Consider the claims in reverse order. Claim (18) can be verified by using a zero check (§3.1.1); *i.e.*, letting $\tilde{g}_r^T$ denote a uniformly chosen random row of G,

$$\tilde{g}_r^T y = \sum_{i=1}^{\ell} \tilde{g}_r^T T_i x_i \quad \underset{p}{\Longrightarrow} \quad y = \sum_{i=1}^{\ell} T_i x_i,$$

where $p \leq 1 - d/m$. Similarly, we may reduce the first claim of verifying that $x_i \in V'$ for each i to verifying a single inclusion via the vector subspace check of §3.1.4:

$$\sum_{i=1}^{\ell} G'_{r'i} x_i \in V \quad \underset{p'}{\Longrightarrow} \quad x_i \in V', \text{ for } i = 1, \ldots, \ell.$$

Here, as shown in the check, $p' \leq 1 - d'/m'$ and r' is uniformly randomly chosen from $1, \ldots, m'$. Combining these two statements gives the final result.

Discussion. In a sense, all we are doing is exploiting the structure we assumed about the space V to reduce checking whether $y \in V$, to (a) checking whether y is equal to some linear combination of vectors, and (b) checking whether these individual vectors are each in the smaller subspace V'. The former we may perform via a zero check, while the latter we may perform via a vector subspace check. If both of these checks are cheaper to perform than the original, then, in a sense, we have reduced the problem to a simpler one.

On the other hand, unless we have some interesting cryptographic primitives (such as homomorphic inner product commitments [BCC$^+$16], *e.g.*), it is not immediately clear how to make this protocol practically useful. To this end, we generalize this basic building block to the *distance check*, which will, in turn, yield a practical protocol.

4.1.2 Distance reduction

In this check, we will attempt to show that the provided vector y is not too far away from $V \subseteq \mathbf{F}^n$. Because of this, the exact choice of basis $T_i \in \mathbf{F}^{n \times k}$ matters.

Basis alignment. To this end, we will define a notion of alignment for the basis $T_1, \ldots, T_\ell$, as follows. We say the T_i are *basis aligned* if, given any $X \in \mathbf{F}^{k \times \ell}$, a matrix with columns x_i, that is q-close to V, we have that

$$\|T_1 x_1 + \cdots + T_\ell x_\ell - V\|_0 \leq q',$$

for some fixed q'. (We must specify the q and q' in the definition, of course.) The interpretation of this statement is that, if the at-most-q 'errors' in the columns of X are aligned (where 'errors' are the number of indices where at least one column of X differs from any closest vector in V'), then the total number of errors over the larger vector space must be similarly small; *i.e.*, at most q'.

While this seems like a strong restriction, we note that just the sparsity of the matrices T_i would imply this claim. For example, if the matrices T_i are diagonal, we would have that $q' \leq q$, or, if each T_i is nonzero in at most two rows, and the T_i all have the same sparsity pattern, we would have that $q' \leq 2q$, and so on. We use this fact in the protocol presented below.

Reduction. Given a set of matrices T_i which are basis aligned in the sense above (and using the same notation q and q' as defined above), then note that, if X is q-far from V, and

$$\|y - (T_1 x_1 + \cdots + T_\ell x_\ell)\|_0 \leq q'', \tag{19}$$

then

$$\|y - V\|_0 \leq q' + q'',$$

via the triangle inequality. To test the first claim (that X is q-far from V), we may apply the subspace distance check of §3.2.3, while the second (19), can be tested via the sparsity check of §3.2.1; both combined imply that the original vector y is indeed close to V. To write this out explicitly, first set

$$\tilde{y} = T_1 x_1 + \cdots + T_\ell x_\ell, \tag{20}$$

for notational convenience. Then, assuming the conjecture of §3.2.3 holds for general codes $G \in \mathbf{F}^{m \times \ell}$ with distance d, we have

$$\|G_{r1} x_1 + \cdots + G_{r\ell} x_\ell - V'\|_0 \leq q \quad \underset{p}{\Longrightarrow} \quad X \text{ is } q\text{-far from } V',$$

where $p \leq (q+1)(1 - d/m)$ and r is uniformly sampled from $1, \ldots, m$. We also have that

$$y_S = \tilde{y}_S \quad \underset{p'}{\Longrightarrow} \quad \|y - \tilde{y}\|_0 \leq q'',$$

where $S \subseteq \{1, \ldots, n\}$ is uniformly randomly sampled, while $p' \leq (1 - (q''+1)/n)^{|S|}$. Combining these two statements, we have that

$$y_S = \tilde{y}_S \text{ and } \|G_{r1} x_1 + \cdots + G_{r\ell} x_\ell - V'\|_0 \leq q \quad \underset{p+p'}{\Longrightarrow} \quad \|y - V\|_0 \leq q' + q''.$$

Here, as before, we have that $S \subseteq \{1, \ldots, n\}$ is uniformly randomly sampled, r is uniformly sampled from $1, \ldots, m$, and $\tilde{y}$ is as defined in (20), while $p \leq (q+1)(1 - d/m)$ and $p' \leq (1 - (q''+1)/n)^{|S|}$.

Discussion. At a high level, we have just reduced checking that a potentially very large vector y is close to a vector space to checking that (a) two vectors are close to each other (via random sampling) and (b) that a linear combination of much smaller vectors is close to some subspace V'. If this subspace also has a similar recursive substructure; *i.e.*, if there exist matrices T_i' and a subspace V'' such that

$$V' = T_1'V'' \oplus \cdots \oplus T_s'V'',$$

then, instead of directly checking that

$$\|G_{r1}x_1 + \cdots + G_{r\ell}x_\ell - V'\|_0 \leq q$$

we may set $y' = G_{r1}x_1 + \cdots + G_{r\ell}x_\ell$ and repeat the above procedure again, except over y' and V', making the appropriate replacements.

Unfortunately, the general reduction presented prior to this discussion paragraph depends on the unsolved conjecture of §3.2.3 (though special cases have been solved). In what follows, we will use the special case where $\ell = 2$, shown earlier, to construct a linear-algebraic generalization of the FRI protocol, along with its security proof.

4.2 The FRI protocol

In this section we will, using the statements above, present a small generalization of the FRI protocol along with a proof of its security.

Set up. In FRI, we have the following construction: $\ell = 2$ and the matrices $T_1, T_2 \in \mathbf{F}^{n\times n}$ are the identity ($T_1 = I$) and diagonal, respectively. (We will define T_2 in what follows.)

In this scenario, the vector space V is the set of evaluations of polynomials of degree $\leq s$ at some fixed set of points $\{\alpha_1, \ldots, \alpha_n\} \subseteq \mathbf{F}^n$. (This forms a vector space by the same reasoning as §1.1.) The 'smaller' vector space V' is the set of evaluations of polynomials of degree $\leq s$ with only even-degree terms, over the same set of fixed points. (This is a vector space, as summing two polynomials with only even-degree terms results in another polynomial with terms of only even degree, as does scaling these polynomials.) Setting

$$T_2 = \begin{bmatrix} \alpha_1 & 0 & \ldots & 0 \\ 0 & \alpha_2 & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & \alpha_n \end{bmatrix},$$

we may then write

$$V = T_1V' \oplus T_2V'.$$

This decomposition has the following interpretation: we may decompose the vector space of polynomials of degree $\leq s$ into two parts: the set of polynomials of degree $\leq s$ with only even-degree terms, and the set of polynomials of degree $\leq s$ with only odd-degree terms. (The latter, of course, is just the set of polynomials with only even-degree terms, multiplied by the polynomial $f(x) = x$.)

Squared evaluations. It is worth noticing that, since $\beta^2 = (-\beta)^2$ for any $\beta \in \mathbf{F}$, in many cases, the set of squared evaluation points, $\{\alpha_i^2\}$, is much smaller than the set of all evaluation points $\{\alpha_i\}$. (If the $\{\alpha_i\}$ form a subfield of $\mathbf{F}$, it is about half, unless $\mathbf{F}$ is a binary field.) In fact, if the evaluation points are chosen such that $\alpha_{i+n/2} = -\alpha_i$ for $i = 1, \ldots, n/2$, then the number of unique evaluation points is halved, and we may assume that $V' \subseteq \mathbf{F}^{n/2}$. In this case, we may write the decomposition with the following matrices:

$$T_1 = \begin{bmatrix} I \\ I \end{bmatrix}, \quad T_2 = \begin{bmatrix} D \\ -D \end{bmatrix},$$

where $D = \mathbf{diag}(\alpha_1, \ldots, \alpha_{n/2})$ and $T_1, T_2 \in \mathbf{F}^{n \times (n/2)}$, where V' is the set of all polynomial evaluations at the points $\{\alpha_i^2\}$ for $i = 1, \ldots, n/2$. This gives yet another way of writing

$$V = T_1 V' \oplus T_2 V'.$$

Note that these matrices are aligned in that, letting the matrix $[x_1 \; x_2]$ be q-close to V', then

$$\|T_1 x_1 + T_2 x_2 - V\|_0 \leq 2q,$$

so, in the notation above, we have $q' = 2q$.

Reduction step. From here it is easy to (a) read off the algorithm for reducing the queries to a smaller subspace, given the prior discussion and (b) to read off its probability of error. From the previous discussion in §4.1.2, we know

$$\|G_{r1} x_1 + G_{r2} x_2 - V'\|_0 \leq q \;\text{ and }\; y_S = (T_1 x_1 + T_2 x_2)_S \quad \underset{p+p'}{\Longrightarrow} \quad \|y - V\|_0 \leq 3q,$$

over uniformly randomly chosen $r = 1, \ldots, m$ and $S \subseteq \{1, \ldots, n\}$, where $p \leq (q+1)(1-d/m)$ and $p' \leq (1 - (q+1)/n)^{|S|}$, and we have used the fact that $q' = 2q$, so $q + q' = 3q$.

Complete reduction. At a high level, we may view the above check as: we begin by wishing to check that some vector is q-close to a subspace (of whatever dimension) embedded in $\mathbf{F}^n$. We then reduce this to checking that some other vector is in a subspace of $\mathbf{F}^{n/2}$, while incurring some additional error term $p + p'$ (as given above) by doing so. We may continue to perform this procedure, say, k times, until we are left with verifying that some vector, embedded in $\mathbf{F}^{n/2^k}$ that we must verify lies in a subspace. (We assume n is divisible by 2^k in what follows.) If $n/2^k$ is small, then every subspace must be similarly 'small', and it suffices to simply directly verify that this last vector is part of the subspace directly. If so, this will imply that the original vector (embedded in the 'large' vector space $\mathbf{F}^n$) must be similarly close to the original vector space V with high probability, controlled by the above terms, which is what we desired to check in the first place.

Now, consider the first 'step' of the protocol. In order to have that y is $3q$-close to V, we must have that

$$\|G_{r1} x_1 + G_{r2} x_2 - V'\|_0 \leq q \;\text{ and }\; \|y - (T_1 x_1 + T_2 x_2)\| \leq q,$$

with high probability. If we then set

$$y' = G_{r1}x_1 + G_{r2}x_2,$$

then, to guarantee that y' is q-close to $V' = T_1'V'' \oplus T_2'V''$ (*i.e.*, to guarantee the left hand side of the first step) we must have that

$$\|G_{r1}x_1' + G_{r2}x_2' - V''\|_0 \leq q/3 \text{ and } \|y' - (T_1'x_1' + T_2'x_2')\| \leq q/3,$$

with high probability. Repeating this process, we see that at the kth round, we must have that both queries must ensure that the respective vectors are no more than $q/3^k$-far from the vector space. By some basic accounting, using the implications of §1.4, it is not hard to show that the probability of error is at most

$$\sum_{i=1}^{k}\left(\frac{q}{3^i}+1\right)\left(1-\frac{d}{m}\right)+\sum_{i=1}^{k}\left(1-\frac{q/3^i+1}{n/2^i}\right)^{|S_i|},$$

where $|S_i|$ is the size of the randomly drawn subset for round i. The first term in this sum is easily bounded from above since

$$\sum_{i=1}^{k}\left(\frac{q}{3^i}+1\right) \leq \frac{3}{2}q + k.$$

The second term is slightly trickier, but, since $\log(1-\lambda) \leq -\lambda$ for $0 \leq \lambda < 1$, then for any nonnegative γ,

$$(1-\lambda)^{\gamma} \leq \exp(-\gamma\lambda),$$

which means that the right-hand-term, inside of the sum, is bounded above by

$$\left(1-\frac{q/3^i+1}{n/2^i}\right)^{|S_i|} \leq \exp\left(-\frac{q}{n}\left(\frac{2}{3}\right)^i|S_i|\right).$$

Setting $|S_i| = \eta(3/2)^i$, for some $\eta \geq 0$ to be established, simplifies the expression slightly to

$$\sum_{i=1}^{k}\left(1-\frac{q/3^i+1}{n/2^i}\right)^{|S_i|} \leq k\exp\left(-\eta\frac{q}{n}\right),$$

which means the probability of error is bounded above by

$$\left(\frac{3}{2}q+k\right)\left(1-\frac{d}{m}\right)+k\exp\left(-\eta\frac{q}{n}\right).$$

Concrete instance. The parameters presented above are relatively abstract. If we set G to be the matrix whose rows are all of the pairs $(1, \alpha)$ for each $\alpha \in \mathbf{F}$ (as in the original FRI protocol [BBHR18]) we know that $1 - d/m = 1/|\mathbf{F}|$, from §1.3.2. We may similarly choose q to be, say, $q = n/8$. If the field size is relatively large compared to n (we will say that $n = 2^{32}$ for this instance and the field size is, say $|\mathbf{F}| \approx 2^{128}$) then, to have an error of around 2^{-80} after $k = 28$ rounds, we must have

$$k \exp\left(-\frac{\eta}{8}\right) \leq 2^{-80},$$

which means that we may set $\eta = 470$. This gives a total error probability of slightly above 2^{-80}.

Query complexity. In this model, the number of queries that must be made to either the vector y or its constituents x_1 or x_2 at round i is given by $|S_i| = \eta(3/2)^i$. This means that the total number of queries is

$$\sum_{i=1}^{k} |S_i| = 2\eta \left(\left(\frac{3}{2}\right)^{k+1} - 1 \right).$$

With the provided parameters, the total number of queries is then $\sim 2^{26}$, which is much smaller than naïvely querying all $n - q \approx 2^{32}$ possible entries to verify the claim. Indeed, it is not hard to show that, if $n/2^k$ remains constant, along with all of the other parameters, then, as n grows, this difference becomes arbitrarily large. (This is easy to see: the query complexity grows as $(3/2)^{k \log(k)}$ whereas the message size grows as 2^k, which marks an exponential(!) gap between the two as k, or, equivalently, n, becomes large.)

Optimizing constants. Note that, in the above construction, we did not make use of the freedom that many parameters, such as the distances between different elements, could be arbitrarily chosen. Indeed, we did not optimize for any constants at all, and, using the bounds presented here, the proof system would be unlikely to be useful for practical applications. We suspect that using the framework presented here, but with much more careful accounting, it is possible to achieve similar constants as those derived by directly analyzing the protocols [BBHR18].

Discussion. Surprisingly, it is possible to reduce any (finite) computation to a small number of queries of the form: is a vector y close to a Reed–Solomon codeword of small degree? (This is not obvious, but is, in a sense, straightforward, given the above; we direct the reader to [Tha22] for much more). This has an extremely interesting consequence. Given the result of some (potentially incorrect) computation: how do we verify its correctness? Intuitively, one would expect that the only way to do so is, roughly speaking, to run the same computation locally and verify that the result is the same. This particular claim shows that it is possible to, asymptotically, perform exponentially better than this, given only the ability to

query evaluations of codes at specified places. This fact, and many of its consequences, are used throughout in a number of applications in blockchains, and, in the not too far future, we expect will be much more widely applied.

5 Conclusion

In this paper, we have shown that a number of the basic tools used in succinct proof systems may be recast, almost entirely, in terms of linear algebra and error correcting codes. This continues a long line of tradition in the cryptography space which attempts to separate out and individually consider the components which make up a succinct proof system. For example, it was possible to mostly elide the use of cryptography by pushing most of the cryptographic complexity into the models of §2.1.2. Focusing on just the reductions used here then leads to a clean abstraction which may be considered in its own right. Indeed, an interesting consequence of this line of work is that it suggests natural generalizations of known statements, leading to, say, the conjecture provided in §3.2.3. In a certain sense, this work also suggests the following high level idea: one may, in many cases, replace a very specific notion of randomness (such as, say, random linear combinations) with the much more structured 'randomness' coming from rows of the generator matrix for an error correcting code of large distance. In many ways, the main point of this paper is to show that even this notion of randomness not only preserves most 'natural' properties of objects in vector spaces, but indeed serves as a way of verifying these properties.

References

[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*, pages 2087–2104, Dallas Texas USA, October 2017. ACM.

[Axl14] Sheldon Axler. *Linear Algebra Done Right*. Springer, New York, 2014.

[BBB+18] Benedikt Bünz, Jonathan Bootle, Dan Boneh, Andrew Poelstra, Pieter Wuille, and Greg Maxwell. Bulletproofs: Short proofs for confidential transactions and more. In *2018 IEEE Symposium on Security and Privacy (SP)*, pages 315–334, 2018.

[BBHR18] Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, and Michael Riabzev. Fast Reed-Solomon Interactive Oracle Proofs of Proximity. page 17 pages, 2018.

[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, volume 9666, pages 327–357. Springer Berlin Heidelberg, Berlin, Heidelberg, 2016.

[BCCT13] Nir Bitansky, Ran Canetti, Alessandro Chiesa, and Elan Tromer. Recursive Composition and Bootstrapping for SNARKs and Proof–Carrying Data. In *STOC'13: Proceedings of the Forty-Fifth Annual ACM Symposium on Theory of Computing*, June 2013.

[BCI+23] Eli Ben-Sasson, Dan Carmon, Yuval Ishai, Swastik Kopparty, and Shubhangi Saraf. Proximity Gaps for Reed–Solomon Codes. *Journal of the ACM*, page 3614423, August 2023.

[BCMS20] Benedikt Bünz, Alessandro Chiesa, Pratyush Mishra, and Nicholas Spooner. Recursive Proof Composition from Accumulation Schemes. In Rafael Pass and Krzysztof Pietrzak, editors, *Theory of Cryptography*, volume 12551, pages 1–18. Springer International Publishing, Cham, 2020.

[BCS21] Jonathan Bootle, Alessandro Chiesa, and Katerina Sotiraki. Sumcheck arguments and their applications. In *Advances in Cryptology – CRYPTO 2021: 41st Annual International Cryptology Conference, CRYPTO 2021, Virtual Event, August 16–20, 2021, Proceedings, Part I*, page 742–773, Berlin, Heidelberg, 2021. Springer-Verlag.

[BF23] Benedikt Bunz and Ben Fisch. Multilinear Schwartz-Zippel mod N and Lattice-Based Succinct Arguments. 2023.

[BJ08] Michał Baczyński and Balasubramaniam Jayaram. *Fuzzy Implications*, volume 231 of *Studies in Fuzziness and Soft Computing*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.

[BS23] Dan Boneh and Victor Shoup. *A Graduate Course in Applied Cryptography*. 2023.

[BSCG+13] Eli Ben-Sasson, Alessandro Chiesa, Daniel Genkin, Eran Tromer, and Madars Virza. Snarks for c: Verifying program executions succinctly and in zero knowledge. In *Advances in Cryptology–CRYPTO 2013: 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part II*, pages 90–108. Springer, 2013.

[DP23] Benjamin E. Diamond and Jim Posen. Proximity testing with logarithmic randomness. Cryptology ePrint Archive, Paper 2023/630, 2023. `https://eprint.iacr.org/2023/630`.

[KL21] Jonathan Katz and Yehuda Lindell. *Introduction to Modern Cryptography*. Chapman & Hall/CRC Cryptography and Network Security Series. CRC Press Taylor & Francis Group, Boca Raton London New York, third edition edition, 2021.

[KP22] Abhiram Kothapalli and Bryan Parno. Algebraic Reductions of Knowledge. 2022.

[KST22] Abhiram Kothapalli, Srinath Setty, and Ioanna Tzialla. Nova: Recursive Zero-Knowledge Arguments from Folding Schemes. In Yevgeniy Dodis and Thomas Shrimpton, editors, *Advances in Cryptology – CRYPTO 2022*, volume 13510, pages 359–388. Springer Nature Switzerland, Cham, 2022.

[KZG10] Aniket Kate, Gregory Zaverucha, and Ian Goldberg. Constant-size commitments to polynomials and their applications. In *Advances in Cryptology–ASIACRYPT 2010: 16th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 5-9, 2010. Proceedings 16*, pages 177–194. Springer, 2010.

[Nit20] Anca Nitulescu. Zk-SNARKs: A Gentle Introduction. 2020.

[RZ21] Carla Rafols and Arantxa Zapico. An Algebraic Framework for Universal and Updatable SNARKs. 2021.

[Tha22] Justin Thaler. Proofs, Arguments, and Zero-Knowledge. *Foundations and Trends® in Privacy and Security*, 4(2–4):117–660, 2022.

[Woo22] Mary Wootters. CS250/EE387 Lecture Notes. *Stanford University*, 2022. Online.

[Woo23] Mary Wootters. Personal communication, 2023.

A Chaining probabilistic implications

Given two statements $P_r \underset{p}{\Longrightarrow} Q_{r'}$ and $Q_{r'} \underset{p'}{\Longrightarrow} T_{r''}$, we will show that

$$P_r \underset{p+p'}{\Longrightarrow} T_{r''}.$$

where the randomness is, as before, over r, r', and r'', with no assumptions on their dependence. Using the definitions for the given notation, this claim may be written as, knowing

$$\Pr(P_r \wedge \neg Q_{r'}) \leq p, \quad \text{and} \quad \Pr(Q_{r'} \wedge \neg T_{r''}) \leq p'$$

implies that

$$\Pr(P_r \wedge \neg T_{r''}) \leq p + p'.$$

Proof. To see this, we may use basic logical implications. Note that, for any r, r', and r'', the following implications are true

$$P_r \wedge \neg T_{r''} \quad \text{implies} \quad (P_r \wedge \neg Q_{r'}) \vee (Q_{r'} \wedge \neg T_{r''}).$$

This follows from the fact that P_r and $\neg T_{r''}$. Since $Q_{r'}$ is either true or false, and both of the previous statements are true, then, necessarily, one of the statements in the disjunction must be true. Since this is true for any r, r', and r'', we then have that

$$\Pr(P_r \wedge \neg T_{r''}) \leq \Pr((P_r \wedge \neg Q_{r'}) \vee (Q_{r'} \wedge \neg T_{r''})) \leq \Pr(P_r \wedge \neg Q_{r'}) + \Pr(Q_{r'} \wedge \neg T_{r''}),$$

where the last inequality follows from the union bound. Using the definitions in the statement of the claim above gives the result.

B Kronecker product of codes

B.1 Kronecker product

Given two matrices $G \in \mathbf{F}^{m \times n}$ and $G' \in \mathbf{F}^{m' \times k}$, we define their *Kronecker product*, written $G' \otimes G \in \mathbf{F}^{mm' \times nk}$, as

$$G' \otimes G = \begin{bmatrix} G'_{11}G & \dots & G'_{1k}G \\ \vdots & \ddots & \vdots \\ G'_{m'1}G & \dots & G'_{m'k}G \end{bmatrix}.$$

This definition may be interpreted in a number of ways. The simplest, perhaps, is to note that, for a list of vectors $x_1, \dots, x_k \in \mathbf{F}^n$, not all zero, then

$$(G' \otimes G) \begin{bmatrix} x_1 \\ \vdots \\ x_k \end{bmatrix} = \mathbf{vec}(G' X^T G^T), \tag{21}$$

where $\mathbf{vec}(G'X^TG^T)$ is a vector corresponding to stacking the columns of $G'X^TG^T$ into one large vector, while

$$X = \begin{bmatrix} x_1 & x_2 & \dots & x_k \end{bmatrix},$$

is a matrix with columns x_i for $i = 1, \dots, k$. If the matrices G and G' define codes, we may view the right hand side of (21) as performing the following series of steps. First, we encode the columns of X using G, giving a big matrix GX. Then, we take the transpose of this matrix and encode its columns, using G', *i.e.*, $G'(GX)^T = G'X^TG$. (Equivalently, this is just encoding the rows of GX to get a new matrix $(GX)G'^T$ and taking the transpose of the result.) Finally, we stack the columns of the resulting matrix $G'X^TG^T$ into one large vector, which gives the final result.

Tensor interpretation. In a certain sense, this may be viewed as the natural 'structure preserving map' which takes operations on tensors and maps them to corresponding operations on vectors. Specifically, given a matrix $X \in \mathbf{F}^{n \times k}$, which may be viewed as a rank 2 tensor, there is an invertible linear map $X \to \mathbf{vec}(X)$ which gives a 'vector' element $\mathbf{F}^{nk}$. (There are many such maps; here, we take a simple one, which is just stacking the columns of X into one large vector.) Applying linear operations to the matrix X then corresponds to a linear operation on $\mathbf{vec}(X)$. In the case that these operations are 'axis' aligned, *i.e.*, that these linear operations only apply column-wise or row-wise on X, then the Kronecker product of the operators (in the tensor space, which deals with the matrix X) is the corresponding linear operator in the vector representation (which deals with the vector $\mathbf{vec}(X)$). It is not hard to see that this generalizes to arbitrary-rank tensors, though we do not show this here.

B.2 Kronecker product distance

Given two codes $G \in \mathbf{F}^{m \times n}$ and $G' \in \mathbf{F}^{m' \times k}$ with distance d and d', respectively, we will show that the distance of the Kronecker product $G' \otimes G \in \mathbf{F}^{mm' \times nk}$ is at least dd'. To see this, we will use a simple counting argument. Consider a list of vectors $x_1, \dots, x_k \in \mathbf{F}^n$, not all zero, then

$$(G' \otimes G) \begin{bmatrix} x_1 \\ \vdots \\ x_k \end{bmatrix} = \mathbf{vec}(G'X^TG^T), \tag{22}$$

where $\mathbf{vec}(G'X^TG^T)$ is a vector corresponding to stacking the columns of $G'X^TG^T$, while

$$X = \begin{bmatrix} x_1 & x_2 & \dots & x_k \end{bmatrix},$$

is a matrix with columns x_i for $i = 1, \dots, k$. Since $\|\mathbf{vec}(G'X^TG^T)\|_0$ is just the number of nonzero entries of the matrix $G'X^TG^T$, we will only consider the latter object. Defining

$\tilde{X} = X^T G^T$, we can then rewrite $G' X^T G^T$ as

$$G' X^T G^T = G' \tilde{X} = G' \begin{bmatrix} (Gx_1)^T \\ \vdots \\ (Gx_k)^T \end{bmatrix}.$$

Since at least one index i has $x_i \neq 0$, then, at least d columns of $\tilde{X}$ will be nonzero, since the distance of G is d by assumption. This implies that each of the (at least) d nonzero columns of $\tilde{X}$ will be encoded into a vector of weight at least d', since G' has distance d', which means that we have at least dd' nonzero entries in $G' X^T G^T$; or, in other words, that the distance of $G' \otimes G$ is at least dd'.

B.3 Tightness

The derivation presented in the reduced matrix zero check §3.1.3 is strictly weaker than the one above as it does not use the linearity properties of the encoding. We show that the bound on the probability of failure derived there, which is

$$1 - \frac{d}{m} + 1 - \frac{d'}{m'},$$

(using the same m and m' and d, d' as above) is essentially always worse than the bound derived here, of

$$1 - \frac{dd'}{mm'}. \tag{23}$$

To see this, note that

$$\left(1 - \frac{d}{m}\right)\left(1 - \frac{d'}{m'}\right) \geq 0, \tag{24}$$

which, after rearranging, implies that

$$1 - \frac{d}{m} + 1 - \frac{d'}{m'} \geq 1 - \frac{dd'}{mm'}.$$

Whenever the quantity in the left hand side of (24) is positive, which always true unless $d = m$ or $d = m'$, the latter quantity holds strictly. In general, we find that chained implications yield much simpler proofs at the expense of some potential error. (Roughly speaking: chained implications assume that the worst-case errors are disjoint, since the argument used is a union-bound one. This is not true, *e.g.*, in the case of linear codes, from the proof above.) It is of course, not hard to show that (23) is the best we can hope for since, given $x \in \mathbf{F}^n$ which has Gx with weight d and $x' \in \mathbf{F}^k$ which has $G'x'$ with weight d', then $x' \otimes x \in \mathbf{F}^{nk}$ will result in a codeword

$$(G' \otimes G)(x' \otimes x) = (G'x') \otimes (Gx),$$

with weight dd'. (The equality follows from the definition (22), as does the weight.) It is not hard to show that the matrix X with $\mathbf{vec}(X) = x' \otimes x$ is given by

$$X = xx'^T,$$

which gives the result.

C Subspace distance check

This appendix section contains some notes on the subspace distance check and the associated distance-preserving encoding conjecture. We use the same definitions and notation as §3.2.3 for the remainder of this appendix section. As a reminder, we are attempting to check if some matrix $X \in \mathbf{F}^{k \times n}$ has columns that are q-close to some vector subspace $V \subseteq \mathbf{F}^k$ whose distance is at least d'. We assume that $q < d'/2$ and rewrite the check below for convenience:

$$\left\| \sum_{i=1}^{n} G_{ri} x_i - V \right\|_0 \leq q \quad \underset{p}{\Longrightarrow} \quad X \text{ is at most } q\text{-far from } V. \tag{25}$$

C.1 Easy case

We may split the proof of the check presented in the paper into two cases. In the first, we assume that X is less than $d'/2$-far from the vector space V; that is, there exists a matrix $Y \in \mathbf{F}^{k \times n}$, with columns in V such that $X - Y$ has less than $d'/2$ nonzero rows. (This is the part of the proof we present below.) In the second case, we assume that X is at least $d'/2$-far from V. This means that, for any matrix Y whose columns lie in V, the matrix $X - Y$ has at least $d'/2$ nonzero rows. (In a certain sense, that the columns of X are 'far' from the vector space V.)

First case proof. This part assumes that X is less than $d'/2$-far from V. This means that there exists a matrix Y such that its columns y_i lie in V, $y_i \in V$, and $X - Y$ has less than $d'/2$ nonzero rows. First, set

$$\bar{y}_r = \sum_{i=1}^{n} G_{ri} y_i, \tag{26}$$

for each $r = 1, \ldots, m$. We will show that this is the unique closest vector in the subspace V to the left hand side of the claim (25):

$$\bar{x}_r = \sum_{i=1}^{m} G_{ri} x_i, \tag{27}$$

where x_i is the ith column of X. This is not hard to see: since $X - Y$ has less than $d'/2$ nonzero rows, then any linear combination of the columns of $X - Y$ must have less than $d'/2$

nonzero entries; *i.e.*,

$$\left\| \sum_{i=1}^{m} G_{ri}(x_i - y_i) \right\|_0 < d'/2,$$

but the term inside of the norm can be split into

$$\left\| \sum_{i=1}^{m} G_{ri}x_i - \sum_{i=1}^{m} G_{ri}y_i \right\|_0 < d'/2,$$

where the second sum can be recognized as $\bar{y}_r \in V$. Since the distance of V is at least d', then this vector is within the unique decoding radius and is therefore the unique vector in V closest to $\bar{x}_r$; *i.e.*, for any choice of r,

$$\left\| \sum_{i=1}^{m} G_{ri}x_i - V \right\|_0 = \left\| \sum_{i=1}^{m} G_{ri}(x_i - y_i) \right\|_0.$$

Armed with this fact, if

$$\left\| \sum_{i=1}^{m} G_{ri}x_i - V \right\|_0 \leq q,$$

then

$$\left\| \sum_{i=1}^{m} G_{ri}(x_i - y_i) \right\|_0 = \left\| \sum_{i=1}^{m} G_{ri}x_i - V \right\|_0 \leq q.$$

But, from the folded sparsity check above, we have that

$$\left\| \sum_{i=1}^{m} G_{ri}(x_i - y_i) \right\|_0 \leq q \quad \underset{p}{\Longrightarrow} \quad X - Y \text{ has at most } q \text{ nonzero rows,}$$

with $p \leq (q+1)(1 - d/m)$ when r is uniformly randomly selected from $1, \ldots, m$. In other words, that $X - Y$ have at most q nonzero rows, or, alternatively that X is q-close to V, as required.

Strengthening this proof. Wootters [Woo23] proved a slightly stronger version of this case in private communication: instead of assuming that X is less than $d'/2$-far from V, it suffices to assume that X is less than $(d'-q)$-far from V, which is at least as good (and often much better) than the original proof, as $q < d'/2$ by assumption. Similar to the previous, we let Y be a matrix whose columns lie in V and $X - Y$ has at most $d' - q$ nonzero rows. The proof is very similar to the above and we reproduce it here for completeness, using the same notation and definitions given above.

Let the left hand side of (25) be satisfied, *i.e.*,

$$\left\| \sum_{i=1}^{n} G_{ri}x_i - V \right\|_0 \leq q,$$

then we will show that the unique vector closest to the linear combination of the columns of X above is $\bar{y}_r$, as defined in (26). To see this, let $y \in V$ be any vector in the subspace, then, if $y \neq \bar{y}_r$, we have

$$\|\bar{x}_r - y\|_0 \geq \|y - \bar{y}_r\|_0 - \|\bar{x}_r - \bar{y}_r\|_0 > d' - (d' - q) \geq q,$$

where $\bar{x}_r$ is also as defined previously in (27), so no other $y \in V$ is closer to $\bar{x}_r$ than $\bar{y}_r$. The first inequality follows from the reverse triangle inequality, while the second follows from the fact that V has distance d', so $\|y - \bar{y}_r\| \geq d'$, and the fact that $X - Y$ has less than $d' - q$ nonzero entries, by assumption on X. So:

$$\left\| \sum_{i=1}^{n} G_{ri}(x_i - y_i) \right\|_0 = \left\| \sum_{i=1}^{n} G_{ri} x_i - V \right\|_0 \leq q,$$

but, from the matrix sparsity check, we know that this implies

$$\left\| \sum_{i=1}^{m} G_{ri}(x_i - y_i) \right\|_0 \leq q \quad \underset{p}{\Longrightarrow} \quad X - Y \text{ has at most } q \text{ nonzero rows},$$

when r is uniformly randomly drawn from $1, \ldots, m$, and where $p \leq (q+1)(1 - d/m)$. In English, this simply says that X is q-close to V, as required.

C.2 Distance-preserving encoding conjecture

The remaining case then, is to prove that, if X is at least $(d' - q)$-far from V, then the probability that

$$\left\| \sum_{i=1}^{n} G_{ri} x_i - V \right\|_0 \leq q,$$

with respect to a uniformly randomly drawn r from $1, \ldots, m$, and where $q < d'/2$ is very low. Indeed, we conjecture that the probability this is true, call it p, satisfies

$$p \leq (q+1)\left(1 - \frac{d}{m}\right),$$

where d is the distance of the code generated by $G \in \mathbf{F}^{m \times n}$. We may write this in the notation of §1.4 as

$$\left\| \sum_{i=1}^{n} G_{ri} x_i - V \right\|_0 \leq q \quad \underset{p}{\Longrightarrow} \quad X \text{ is not } (d' - q)\text{-far from } V,$$

where $p \leq (q+1)(1 - d/m)$.

Notes. Such a statement is similar in spirit to that of [AHIV17]. In a certain sense, the above statement means that we may replace the uniform randomness of the check presented in [AHIV17] with the much more structured 'randomness' coming from picking a uniformly random row of an error correcting code matrix G with reasonable distance. For example, an interesting statement, similar in spirit to those used in the proof of the Hadamard case in [AHIV17], is the following. Given any nonzero vector $y \in \mathcal{R}(X)$, let $V^\perp \subseteq \mathcal{R}(X)$ be the space such that any vector $v \in \mathcal{R}(X)$ may be written uniquely as

$$v = \alpha y + z,$$

for some $z \in V^\perp$ and $\alpha \in \mathbf{F}$. (In other words, $V^\perp$ is the remainder of the basis completion of the matrix containing only the column y; in some cases the subspace $V^\perp$ is called the *rejection* of the ray given by y.) Then, for a randomly drawn row $\tilde{g}_r$ of G, if

$$X\tilde{g}_r = \alpha_r y + z_r,$$

where $z_r \in V^\perp$ and $\alpha_r \in \mathbf{F}$, then the probability that $\alpha_r = 0$ is no more than $1 - d/m$. In other words, given any possible 'direction' y in the range of X, we know that, with high probability, any structured linear combination of the columns of X always has a component in this direction. Unfortunately, unlike the proof of [AHIV17], we do not know much about the relationship between z_r and α_r, which makes it difficult to conclude something about the probability that a given vector is far from $\mathcal{R}(X)$. (In their proof, α_r and z_r are known to be independent variables, with respect to the randomness r.) We may view this statement as saying that the structured linear combinations given by a uniformly sampled row, $\tilde{g}_r$, are, in a sense, 'good testers' of the range of X.