Fast Bounded-Independence Functions and Their Duals*

Martijn Brehm† Yuval Ishai‡ Nicolas Resch§

June 5, 2026

Abstract

We continue the study of *fast* functions, computable by linear-size circuits, that share useful properties of random functions. Motivated by cryptographic applications, we generalize and improve on previous results in this area, obtaining the following results:

- For any constant t, we construct a fast t-wise independent hash function with algebraic degree $\log_2 t$ (over $\mathbb{F}_2$), simultaneously optimizing both asymptotic circuit size and degree.
- We simplify and improve a recent construction (ITCS 2026) of a family of fast codes with fast duals, both meeting the Gilbert-Varshamov bound. Unlike the previous construction, our construction has negligible failure probability, can accommodate general fields and rates, supports a systematic encoding, and admits fast universal encoders.
- We strengthen the above to support stronger random-like properties, such as optimal combinatorial list-decoding. This is achieved by constructing, for any constant t, a family of fast linear functions that map any t linearly independent inputs to uniform and statistically independent outputs. Prior to our work, this was only known for $t = 1$.

We demonstrate the usefulness of the above results to cryptography. This includes the first nontrivial protocols for perfectly secure multiparty computation whose circuit complexity scales linearly with the number of parties, as well as protocols for computing encrypted matrix-vector products with optimal asymptotic circuit complexity.

*This is a full version of [BIR26].

†Informatics Institute, University of Amsterdam. m.a.brehm@uva.nl.

‡Technion and AWS. yuvali@cs.technion.ac.il. This work is not associated with Amazon. Research supported by ISF grant 3527/24 and BSF grant 2022370.

§Informatics Institute, University of Amsterdam. n.a.resch@uva.nl. Research supported by an NWO (Dutch Research Council) grant with number C.2324.0590.

Contents

1 Introduction **3**
- 1.1 Our Results . . . 3
- 1.2 Technical Overview . . . 5
- 1.3 Related Work . . . 7

2 Preliminaries **8**
- 2.1 Coding Theory . . . 8
- 2.2 Computational Model . . . 9
- 2.3 Hash Functions . . . 10
- 2.4 The q-ary XOR Lemma . . . 11

3 Constructions of Fast Codes with Fast Duals **11**
- 3.1 Fast Good Code with Fast Good Dual . . . 13
- 3.2 Fast List-Decodable Code with Fast List-Decodable Dual . . . 15

4 Fast Bounded-Independence Functions **18**
- 4.1 Fast t-Wise Independent Hash Functions . . . 18
- 4.2 Fast Bounded-Independence LUOF . . . 22

5 Cryptographic Applications **24**
- 5.1 Fast Information-Theoretic MPC . . . 25
- 5.2 Fast Encrypted Matrix-Vector Product . . . 27
- 5.3 Fast and Conservative Substitutes for Random Linear Codes . . . 27
- 5.4 MPC-Friendly t-Wise Hash Functions . . . 28

A Deferred Proofs **32**
- A.1 Proof of Lemma 5.3 . . . 32
- A.2 Analysis of Protocol 1 . . . 33

1 Introduction

An important trend in cryptographic research is to develop cryptographic schemes realizing functionalities as efficiently as possible. Ideally, the cost of performing a task *with* security should be on the order of the cost of performing the same task *without* security. That is, security just results in a *constant* multiplicative computational overhead. Originating from [IKOS08], this has been the theme of a line of work on cryptography with constant computational overhead; see [BCG+23] and the references therein.

As a concrete example, consider the task of hashing a large amount of data, where one generates a short digest from a long string. One typically wants the hash function to satisfy certain desirable properties of a random function. Ideally, the cost of performing the hashing should only scale linearly with the size of the data. Which "random-like" properties can be satisfied while achieving constant computational overhead compared to just reading the input?

Similarly, one can consider the task of *encoding* a large vector in a way that satisfies desirable properties of a random linear code. In this case, one is sometimes also required to apply the encoding function of the *dual* code. Is it possible to implement *both* the primal and the dual encoding with constant overhead?

The above hashing and encoding primitives serve as useful building blocks in many algorithmic and cryptographic applications. Minimizing their overhead is relevant to all of these applications.

1.1 Our Results

To make progress on the above questions, we provide new constructions that achieve *constant computational overhead* for random-like functions. This refers by default to implementations by (fan-in 2) Boolean or arithmetic circuits whose size scales linearly with the input and output length; see Section 2.2 for discussion. We refer to such an implementation as being *fast*.

For hash functions, the work of Ishai, Kushilevitz, Ostrovsky, and Sahai [IKOS08] demonstrated the existence of fast *pairwise independent* hash functions with arbitrary input and output length.

We first demonstrate that one can build on this result to instead obtain fast *t-wise independent* hash functions. That is, even after seeing $t-1$ hash values, the t-th value is completely unpredictable (assuming the input is distinct from the previous $t-1$ inputs). This result has already been discussed informally in talks by authors of [IKOS08]. Here we spell it out in detail, also achieving near-optimal algebraic degree as a function of t, which is useful for cryptographic applications. This result also acts as a nice warmup for our later results.

Theorem 1.1 (Fast t-wise hash function; informal version of Theorem 4.3)**.** For any constant $t \geq 2$, finite field $\mathbb{F}_q$, input length k, and output length r, there exists a construction of a fast t-wise independent hash function family $\mathcal{H}$ of functions $h : \mathbb{F}_q^k \to \mathbb{F}_q^r$. For $q = 2$, this can be done with (optimal) algebraic degree of $\log_2 t$.

Fast dual codes. Partially motivated by cryptographic applications, a recent work of Brehm and Resch [BR26] constructed a family of linear codes $\mathcal{C} \leq \mathbb{F}_2^n$ of rate $1/2$ with the following properties:

- Both the code and its dual admit *fast* encoding;
- With probability $\geq 1 - 1/\mathrm{poly}(n)$, the code *and its dual* $\mathcal{C}^\perp$ achieve distance ε-close to the Gilbert-Varshamov (GV) bound.

Recall that a code $\mathcal{C}$ of rate R achieves distance ε-close to the GV bound, if its (relative) distance is at least $h_q^{-1}(1-R-\varepsilon)$, where h_q is the q-ary entropy function (and h_q^{-1} is its inverse); see Section 2.1 for more discussion. The GV bound captures the distance achieved by *uniformly* random linear codes, which is typically the best distance one could reasonably hope for.

The [BR26] result was based on a rather involved analysis of a code constructed in a manner inspired by repeat-multiple-accumulate codes [DJM98]. While it provided good concrete efficiency that can be useful for applications (see Remark 1.3), this result also has several drawbacks:

- The code is only constructed over the binary field $\mathbb{F}_2$;
- The code (and its dual) need to have rate exactly $1/2$;
- The randomized construction of the code has a non-negligible failure probability;
- The encoding map is not systematic (which is a problem for some applications);
- The construction does not provide a fast *universal circuit* for encoding given both the message and the randomness defining the code.

Building on the work of Druk and Ishai [DI14] (which, in turn, builds on [IKOS08]), we provide a simpler alternative construction that does not suffer from any of these drawbacks.

Theorem 1.2 (Fast code with fast dual; informal version of Theorem 3.1)**.** For any $R \in (0,1)$ and any prime power q, there is a family of codes $\mathcal{C} \leq \mathbb{F}_q^n$ of rate R along with their duals $\mathcal{C}^\perp$, for which:

- Both $\mathcal{C}$ and $\mathcal{C}^\perp$ admit fast systematic encoding;
- $\forall \varepsilon > 0$, $\mathcal{C}$ and $\mathcal{C}^\perp$ are both ε-close to the GV bound with probability $\geq 1 - 2q^{-\varepsilon n}$.

Furthermore, the encoding matrices for $\mathcal{C}$ and $\mathcal{C}^\perp$ are determined by a linear-sized universal circuit.

Remark 1.3. A significant advantage of the construction from [BR26] is concrete efficiency: encoding can be done by circuits of size $12n$. In contrast, the current approach involves much bigger hidden constants that we did not attempt to optimize.

As an additional comment, for both constructions the encoding circuit has depth $O(\log n)$, which is asymptotically optimal in the bounded fan-in case. However, here too we expect our construction to have a much bigger hidden constant than the previous construction from [BR26].

Fast, random-like codes. By building off the concept of *local similarity* [MRR+24, GM22, MRSY25] – which informally allows one to argue that A code ensemble "behaves like" a uniformly random linear code – we can show that we can extend upon this construction, to obtain $\mathcal{C}$ and $\mathcal{C}^\perp$ possessing stronger properties. For example, they can be proved to have very good *list-decodability*, up to capacity, with optimal list sizes. Recall that a code $\mathcal{C} \leq \mathbb{F}_q^n$ is said to be (ρ, L)-list-decodable if for all $z \in \mathbb{F}_q^n$ the number of codewords within distance ρ from z is at most L. Additionally, the rate $1 - h_q(\rho)$ is called the list-decoding capacity: informally, it is the maximum rate at which a code can be (ρ, L)-list-decodable with any "reasonable" (i.e., sub-exponential) list-size L. We say a code is ε-close to list-decoding capacity if it has rate at least $1 - h_q(\rho) - \varepsilon$.

Theorem 1.4 (Fast list-decodable code with fast dual; informal version of Theorem 3.2)**.** For any $R \in (0,1)$ and any prime power q, there is a randomized procedure producing a code $\mathcal{C} \leq \mathbb{F}_q^n$ of rate R for which:

- Both $\mathcal{C}$ and $\mathcal{C}^\perp$ admit fast systematic encoding;
- $\forall \varepsilon > 0$, both $\mathcal{C}$ and $\mathcal{C}^\perp$ are ε-close to capacity, with list-size $O(1/\varepsilon)$ with probability $\geq 1 - q^{-\Omega(\varepsilon n)}$.

Furthermore, the encoding matrices for $\mathcal{C}$ and $\mathcal{C}^\perp$ are determined by a linear-sized universal circuit.

We comment here that list-size $O(1/\varepsilon)$ is the best one can "reasonably" hope for, in the sense that there is no known construction of a code ε-close to list-decoding capacity with list-size $o(1/\varepsilon)$. For context, prior work had managed to construct linear-time encodable codes with such list-decodability [GM22], and also managed to construct a code where it and its dual achieve such list-decodability [MRSY25]: getting both properties simultaneously is, to the best of our knowledge, novel.

Cryptographic applications. While our results are broadly useful in any contexts in which error-correcting codes or hash functions are used, they are particularly motivated by several cryptographic applications.

In the context of information-theoretic cryptography, any linear code along with a dual code can support *secure multiplication* in secure multiparty computation (MPC) [Mas95, CDM00]. Our results imply the first perfectly secure MPC protocols in which both the security threshold and the computational complexity (measured by Boolean circuit size) scale linearly with the number of parties. Concretely, for any constant-size function f whose inputs and outputs are owned by a constant number of parties, we get a perfectly secure n-party protocol, with a near-optimal (passive) corruption threshold of $t = (1/2 - \varepsilon) \cdot n$, in which the total circuit size of all parties is $O(n)$. Our result on fast and minimal-degree t-wise hash functions can also be useful for simultaneously minimizing communication and round complexity in distributed MPC implementations.

In the context of complexity-based cryptography, our work (as well as the related [BR26]) is motivated by a recent technique for computing *encrypted matrix-vector products* [BCH+25]. This technique relies on two dual codes, where one code is used for encrypting the matrix and the other is used for encrypting the vector. Our construction of fast dual codes gets around some of the limitations of the previous construction from [BR26] which are relevant to this use case.

See Section 5 for details on these and other cryptographic applications.

1.2 Technical Overview

In this section, we briefly discuss how we obtain our main results. First, we recall the IKOS construction of linear-sized pairwise independent hash function [IKOS08], and then the DI construction of a linear-time encodable code getting ε-close to the GV bound [DI14], which itself builds off the IKOS construction.

Firstly, the IKOS construction essentially goes as follows:

1. First, apply a fast code with large distance to the input. The alphabet for this code will be large, but constant.
2. Next, apply a (constant-sized) pairwise-independent hash function to each of the coordinates.
3. Lastly, apply an extractor for bit-fixing sources, which is obtained from the transpose of a generator matrix for a good code. In particular, one can take the transpose of the code applied

in the first step (this also guarantees this step is fast, thanks to a "transposition" principle – cf. Lemma 3.4).

A visual depiction of this construction can be found in Figure 1.

The intuition for the pairwise independence is as follows. Given two distinct inputs, the first encoding step produces codewords that have many distinct coordinates. On these distinct coordinates, the constant-sized hash functions map the coordinates to independent, uniform values. The last step then extracts from these many coordinates with independent values a single string of independent values.

To get the t-wise independent hash function family, we simply have to replace the constant-sized pairwise-independent hash functions by constant-sized t-wise independent hash functions. The reasoning now is very similar. Suppose now we are given t distinct inputs. If the initial code distance is good enough (greater than $1 - 1/t^2$), then in many of the coordinates all the values of the encodings will be distinct, allowing the inner hash functions to output independent, uniform randomness which can be extracted in the last step.

We now turn to the constructions of the codes. Here, we make use of the concept of a *linear uniform output family (LUOF)*, as defined in [DI14]. This is a random variable A distributed over $k \times r$ matrices such that, for any fixed *nonzero* vector $x \in \mathbb{F}_q^k$, xA is distributed uniformly over $\mathbb{F}_q^r$. We recall that if A is uniformly random, then it has this property. In this work – as in [DI14] – we look for distributions over $k \times r$ matrices such that the map $x \mapsto xA$ is *fast* (informally, implementable by a linear-sized circuit). For brevity, we will call such an LUOF *fast*.

The basic idea of [DI14] is to construct a fast LUOF over $k \times n$ matrices. It is relatively straightforward that this suffices for obtaining codes ε-close to the GV bound.[1] To get the fast LUOF, [DI14] keep the first and third steps the same as in the IKOS construction, but instead of applying pairwise independent hash functions, they simply apply constant-sized matrices to each of the coordinates, which are viewed as length-β vectors over the base field $\mathbb{F}_q$ (recall the first encoding step may increase the alphabet size, but only by a constant amount). The reasoning for why this gives an LUOF is very similar. Given a nonzero message vector, the first encoding step gives many coordinates on which it is nonzero, and on these coordinates the small matrices map them to uniform values, which the final step can extract into a single uniform string, as required.

Now, to get codes where both it and its dual have good distance, we look at the code with systematic encoding map $x \mapsto (x, xA)$, where now A is supported on $k \times r$ matrices. This gives a code $\mathcal{C}$ of block-length $n := k + r$ and dimension k, and if A is fast, then so is the map $x \mapsto (x, xA)$. Additionally, it is easy to verify that the code defined by the encoding map $y \mapsto (-yA^\top, y)$ is dual to $\mathcal{C}$, and the transposition principle guarantees that computing $y \mapsto -yA^\top$ is also fast. Furthermore, we show that if A is LUOF, then so is $A^\top$: the crucial ingredient in this argument is Vazirani's XOR Lemma [CGH$^+$85, Rao07]. Finally, a simple argument shows then that both $\mathcal{C}$ and $\mathcal{C}^\perp$ get ε-close to the GV bound with high probability, as desired.

To consider more sophisticated properties like list-decodability, we observe that it suffices to consider the following generalization of the LUOF property. We say a random matrix A supported on $k \times r$ matrices is t-LUOF if, given t *linearly independent* vectors $x_1, \ldots, x_t \in \mathbb{F}_q^k$, the vectors $x_1A, \ldots, x_tA$ are uniform and independent over $\mathbb{F}_q^r$.[2] Again, we observe that if A is a uniformly

[1] Indeed, one can inspect the "standard" argument that random linear codes – i.e., codes defined by sampling a uniformly random generator matrix – get ε-close to the GV-bound with high probability, and observe that the only property one uses is that uniformly random matrices give an LUOF.

[2] In this second case, we mean independent in the "stochastic" sense.

random matrix, then it has this property for any $t \leq k$ (for $t > k$, the concept is vacuous, as there are no sets of $k+1$ or more linearly independent vectors in $\mathbb{F}_q^k$).

Our first observation is that, by assuming the first code of the DI construction has sufficiently good distance ($> 1 - 1/q^t$), we can obtain a t-LUOF via their construction. The argument is similar to above: now, we are given t linearly independent vectors $x_1, \ldots, x_t$, and we need to argue that in many coordinates we still have linearly independent vectors (recall that each coordinate of this first encoding step is a length-β vector). This indeed follows assuming the distance is sufficiently large. Then, on these good coordinates, the inner matrices map the linearly independent vectors to independent, uniform values, which are then extracted in the final step.

Having obtained thus a t-LUOF A, we can then consider the (random) code $\mathcal{C}$ with encoding map $x \mapsto (x, xA)$, along with its dual defined by the encoding $y \mapsto (-yA^\top, y)$. That we can then prove these codes achieve sophisticated properties like list-decoding then follows from the theory of local similarity, as developed in recent works [MRR+24, GM22, MRSY25]. Morally speaking, one observes that if one wishes to prove a code is (ρ, L)-list-decodable, then one is interested in computing quantities like $\mathbb{P}[x_1, \ldots, x_{L+1} \in \mathcal{C}]$, where $x_1, \ldots, x_{L+1}$ are all within distance ρ of some vector z. Suppose one has an argument that shows uniformly random linear codes $\mathcal{C}'$ are with high probability (ρ, L)-list-decodable, where $\mathcal{C}'$ is defined as the kernel of a uniformly random $(n-k) \times n$ matrix. In this case, we know that $\mathbb{P}[x_1, \ldots, x_{L+1} \in \mathcal{C}'] = q^{-(n-k)\dim(\mathrm{span}\{x_1, \ldots, x_{L+1}\})}$. One can verify that for the random $\mathcal{C}$ defined as above, we have $\mathbb{P}[x_1, \ldots, x_{L+1} \in \mathcal{C}'] = q^{-(n-k)\dim(\mathrm{span}\{x_1, \ldots, x_{L+1}\})}$, assuming A is $(L+1)$-LUOF. That is, if we take $t = L+1$, then we get list-decodability for $\mathcal{C}$ (or, more precisely, whatever list-decodability is known to hold for $\mathcal{C}'$ whp, also holds for $\mathcal{C}$ whp). Also, generalizing the reasoning above we can show that if A is t-LUOF, then so is $A^\perp$, in which case we can derive analogous list-decodability guarantees for $\mathcal{C}^\perp$.

1.3 Related Work

While we have already mentioned the papers most directly connected to our work, we now mention a few more relevant related works.

Work by Gál, Hansen, Koucký, Pudlák and Viola [GHK+13] studies circuit sizes for encoding maps of asymptotically good codes over the binary alphabet. However, their model allows for gates with *unbounded* fan-in (whereas we allow only fan-in 2), and additionally they insist of small depth (constant, or $\log^* n$), whereas we allow for larger depth (say, $O(\log n)$).

Another related work by Li [Li26] considers the *unbounded* arithmetic circuit complexity (i.e., gates are arbitrary and have unbounded fan-in) of secret-sharing schemes. The main result is a demonstration that the graph underlying the circuit computing the shares for such a scheme must be a *superconcentrator*. Based on this fact, lower bounds on the size of such a sharing circuit are derived.

A final related work is due to Fan, Li and Yang [FLY22], which builds a pairwise independent hash function with optimal size of $\approx 2n$ (where n is the input length) in the B_2 circuit model (i.e., Boolean circuits with arbitrary fan-in 2 gates).[3] Unlike the IKOS construction (and ours), their construction only yields highly compressive mappings (that is, the output length is $\approx n^{0.1}$), it is not *perfectly* pairwise independent (only statistically close), and only achieves linear size when the circuit can depend on the key (rather than having a single universal circuit of linear size that takes both the key and the input).

[3] They obtain additional bounds for other circuit classes.

2 Preliminaries

We use $\mathbb{N}$ to denote the set of positive integers, and for $n \in \mathbb{N}$ we abbreviate $[n] := \{1, 2, \ldots, n\}$. Throughout, q denotes a prime power, and $\mathbb{F}_q$ a finite field with q elements. By default, we view vectors $x \in \mathbb{F}_q^n$ as row vectors. We will also use tuple notation to denote concatenation: given $x \in \mathbb{F}_q^n$ and $y \in \mathbb{F}_q^m$, $z = (x, y)$ denotes the length $n + m$ vector such that $z_i = x_i$ for $i \in [n]$, and $z_i = y_{i-n}$ for $i \in \{n+1, \ldots, n+m\}$.

For a distribution $\mathcal{D}$ over a finite set U (which itself is a function $\mathcal{D} : U \to [0, 1]$ satisfying $\sum_{u \in U} \mathcal{D}(u) = 1$), we denote by $X \sim \mathcal{D}$ a random variable distributed according to $\mathcal{D}$, i.e., for all $u \in U$, $\mathbb{P}[X = u] = \mathcal{D}(u)$. For a finite set S, we write $X \sim S$ to denote that X is a random variable distributed uniformly at random over the set S, i.e., $\mathbb{P}[X = u] = \frac{1}{|S|}$ if $u \in S$, and 0 otherwise.

For vectors $x, y \in \mathbb{F}_q^n$, their *inner-product* is defined as $\langle x, y \rangle := \sum_{i=1}^n x_i y_i \in \mathbb{F}_q$. We recall that the inner-product thus defined is commutative ($\langle x, y \rangle = \langle y, x \rangle$), and that additionally it is linear in both arguments, e.g., $\langle \alpha x + \beta y, z \rangle = \alpha \langle x, z \rangle + \beta \langle y, z \rangle$, where also $z \in \mathbb{F}_q^n$ and $\alpha, \beta \in \mathbb{F}_q$. For a subspace $V \leq \mathbb{F}_q^n$, we denote its *dual space* by $V^\perp := \{u \in \mathbb{F}_q^n : \forall v \in V,\ \langle u, v \rangle = 0\}$, which we recall is also a subspace and has dimension $\dim(V^\perp) = n - \dim(V)$, and that additionally $(V^\perp)^\perp = V$. Lastly, we recall that given a matrix $M \in \mathbb{F}_q^{a \times b}$ and vectors $x \in \mathbb{F}_q^a$ and $y \in \mathbb{F}_q^b$, we always have $\langle xM, y \rangle = \langle x, yM^\top \rangle$, where $M^\top \in \mathbb{F}_q^{b \times a}$ is the transpose of M.

2.1 Coding Theory

An $\mathbb{F}_q$-*linear code* $\mathcal{C}$ is an $\mathbb{F}_q$-linear subspace of $\mathbb{F}_q^n$ for some $n \in \mathbb{N}$. The *block-length* of $\mathcal{C}$ is the value n. If $\dim(\mathcal{C}) = k$, then $\mathcal{C}$'s *rate* is $R(\mathcal{C}) := \frac{k}{n}$. Given two vectors $x, y \in \mathbb{F}_q^n$, the *(relative) Hamming distance* between them is $d(x, y) := \frac{1}{n} |\{i \in [n] : x_i \neq y_i\}|$. Additionally, for a vector $x \in \mathbb{F}_q^n$ its *(relative) Hamming weight* is $\mathrm{wt}(x) := \frac{1}{n} |\{i \in [n] : x_i \neq 0\}| = \frac{1}{n} |\mathrm{supp}(x)|$, where we've also defined the *support* of a vector as the set of nonzero coordinates. Note that $\mathrm{wt}(x) = d(x, 0)$ and that $d(x, y) = \mathrm{wt}(x - y)$. Because of this latter equality, for any $\mathbb{F}_q$-linear code we have that the *(relative) minimum distance* $\delta(\mathcal{C}) := \min\{d(c, c') : c, c' \in \mathcal{C}, c \neq c'\}$ is also equal to $\min\{\mathrm{wt}(c) : c \in \mathcal{C} \setminus \{0\}\}$. Lastly, for a linear code $\mathcal{C}$, we refer to $\mathcal{C}^\perp$ as its *dual code*.

Every $\mathbb{F}_q$-linear code of dimension k admits a generator matrix $G \in \mathbb{F}_q^{k \times n}$, which is a rank-$k$ matrix such that $\mathcal{C} = \{xG : x \in \mathbb{F}_q^k\}$. It additionally admits a parity-check matrix $H \in \mathbb{F}_q^{(n-k) \times n}$, which is a rank-$n - k$ matrix such that $\mathcal{C} = \{x \in \mathbb{F}_q^n : Hx^\top = 0\}$. It follows that if H is a parity-check matrix for $\mathcal{C}$, then H is a generator matrix for $\mathcal{C}^\perp$. Additionally, the matrices satisfy $GH^\top = 0$. If G is of the form $G = \left[I_k \mid A\right]$ where $I_k \in \mathbb{F}_q^{k \times k}$ is the $k \times k$ identity matrix and $A \in \mathbb{F}_q^{k \times (n-k)}$, then G is said to be *systematic*. Note that if $G = \left[I_k \mid A\right]$ is a systematic generator matrix for $\mathcal{C}$, then $H = \left[-A^\top \mid I_{n-k}\right]$ is a parity-check matrix for $\mathcal{C}$, as

$$GH^\top = \left[I_k \mid A\right] \begin{bmatrix} -A \\ I_{n-k} \end{bmatrix} = -A + A = 0 \ .$$

We will also be interested in codes where the underlying alphabet is in fact a vector space over $\mathbb{F}_q$. Specifically, for $\beta \in \mathbb{N}$ we will refer to an $\mathbb{F}_q$-subspace $\mathcal{C} \leq \left(\mathbb{F}_q^\beta\right)^n$ as an $\mathbb{F}_q$-*additive code over* $\mathbb{F}_q^\beta$. The block-length is still n, while the rate of such a code is now $\frac{k}{\beta n}$, and we define its minimum distance with respect to the alphabet $\mathbb{F}_q^\beta$. That is, given a codeword $c \in \mathcal{C}$, writing it as $c = (c^{(1)}, \ldots, c^{(n)})$ with each $c^{(i)} \in \mathbb{F}_q^\beta$, we define $\mathrm{wt}_\beta(c) := \frac{1}{n} |\{i \in [n] : c^{(i)} \neq 0\}|$, and then

$\delta(\mathcal{C}) := \min\{\mathrm{wt}_\beta(c) : c \in \mathcal{C} \setminus \{0\}\}$. Note that all $\mathbb{F}_q$-additive codes over $\mathbb{F}_q^\beta$ admit a generator matrix G of size $k \times \beta n$.

We often deal not with a single code $\mathcal{C}$, but rather with an infinite family of codes $\{\mathcal{C}_i\}_{i\geq 1}$ for an increasing sequence of block-lengths $n_i \in \mathbb{N}$, each with dimension $\{k_i\}_{i\geq 1}$ and distance $\{\delta_i\}_{i\geq 1}$. The rate of the family of codes can then be defined as $R := \liminf_{i\to\infty} k_i/n_i$ and the relative distance as $\delta := \liminf_{i\to\infty} \delta_i$. We call a family of codes *asymptotically good* whenever both $R > 0$ and $\delta > 0$. We assume the field $\mathbb{F}_q$ is the same within a family.

The q-ary Gilbert-Varshamov bound gives a lower bound on the asymptotic relative distance δ given the rate R for a family of codes over $\mathbb{F}_q$. In particular, it represents the minimum distance achieved by uniformly random (linear) codes (defined, e.g., by sampling a uniformly random generator matrix). Its definition makes use of the *q-ary entropy function* $h_q : [0, 1-1/q] \to [0, 1]$, which itself is defined as

$$h_q(x) := x \log_q(q-1) + x \log_q \frac{1}{x} + (1-x) \log_q \frac{1}{1-x}$$

for $x \in (0, 1-1/q]$ and $h_q(0) = 0$ at the left endpoint. This function increases continuously on its domain, and therefore admits a continuous inverse $h_q^{-1} : [0, 1] \to [0, 1-1/q]$. The *Gilbert-Varshamov (GV) bound* states that there exist codes of distance $\geq \delta$ and rate $\geq 1 - h_q(\delta)$. In other words, codes of rate R can achieve distance $\geq h_q^{-1}(1-R)$. If a rate R code achieves distance $h_q^{-1}(1-R-\varepsilon)$ for some $\varepsilon > 0$, we will say that it is *ε-close to the GV bound*. We recall that uniformly random codes are ε-close to the GV bound with probability $\geq 1 - q^{-\varepsilon n}$.

Lastly, we recall the concept of *list-decodability*. For $\rho \in (0, 1-1/q)$ and $L \in \mathbb{N}$, we say that a code $\mathcal{C} \subseteq \Sigma^n$ (for an arbitrary finite alphabet Σ) is *(ρ, L)-list-decodable* if, for all $z \in \Sigma^n$, we have

$$|\{c \in \mathcal{C} : d(c, z) \leq \rho\}| \leq L \ .$$

It is known codes of rate R can achieve decoding radius ρ so long as $\rho < h_q^{-1}(1-R)$. More precisely, there exist codes of rate $1 - h_q(\rho) - \varepsilon$ that are $(\rho, O(1/\varepsilon))$-list-decodable; however, any code of rate $1 - h_q(\rho) + \varepsilon$ is not (ρ, L)-list-decodable for any $L \leq q^{o(n)}$. For this reason, $1 - h_q(\rho)$ is called *list-decoding capacity*. We will say that a code is *ε-close to list-decoding capacity* if it has rate $1 - h_q(\rho) - \varepsilon$. It is known that there exist codes ε-close to list-decoding capacity with list-size $O(1/\varepsilon)$; in fact, random linear codes achieve this with high probability [GHK11, LW20]. Furthermore, it is not known if codes ε-close to list-decoding capacity exist with list-size $o(1/\varepsilon)$; thus, hoping for list-size $O(1/\varepsilon)$ is a reasonable target.

2.2 Computational Model

We will be interested in encoding functions and hash functions that admit "fast" algorithms. This refers by default to having a uniform family of linear-size arithmetic circuits, but can also refer to other computational models in which Spielman's error correcting codes can be implemented in linear time. See [Spi96] for discussion.

More concretely, our default complexity measure refers to the size of an arithmetic circuit over a finite field $\mathbb{F}_q$. Wires carry field elements into gates. Gates have two input wires and output the addition, multiplication or subtraction of the incoming field elements. We allow gates to have arbitrary fan-out. We also allow gates with a fan-in of 0, which represent either an input variable, a constant scalar, or (for randomized circuits) an independently uniformly random value from $\mathbb{F}_q$. We will typically consider *linear* circuits in which one of the inputs for each multiplication gate is a scalar

The *size* of the circuit is defined as the number of wires, the *depth* as the length of the longest path from an input to an output, and the *algebraic degree* as the highest (total) degree of an output as an $\mathbb{F}_q$-polynomial in the inputs.

We consider infinite families of circuits C_k over $\mathbb{F}_q$ where C_k has k input variables. We consider uniform families, which means there is a polynomial time algorithm that maps 1^k to a description of C_k. We call a function $f : \mathbb{F}_q^k \to \mathbb{F}_q^r$ *fast* if there exists a uniform circuit family C_k which computes f, where each C_k has size $O(k)$. We will sometimes apply this notion also to families of *randomized* functions computed by randomized circuits. Finally, for a family of *linear* functions, we assume by default that the family of linear-size circuits are *linear* in the sense defined above.

2.3 Hash Functions

First, we recall the definition of a hash function with bounded independence.

Definition 2.1 (t-wise independent hash function)**.** Let $q, k, r \in \mathbb{N}$ and let $\mathcal{H}$ denote a distribution over functions from $[q]^k \to [q]^r$. We say that $\mathcal{H}$ is a *t-wise independent hash function family* if, for all $x_1, \ldots, x_t \in [q]^k$ *distinct*, we have, over $H \sim \mathcal{H}$,

$$(H(x_1), \ldots, H(x_t)) \sim [q]^{rt} \ .$$

In other words, for all $y_1, \ldots, y_t \in [q]^r$ it holds that

$$\mathbb{P}\left[\bigwedge_{i=1}^{t} H(x_i) = y_i\right] = q^{-rt} \ .$$

We recall the following classical construction of t-wise independent hash functions, based on degree $\leq (t-1)$-polynomials [CW79, Vad12].

Proposition 2.2 (Construction of t-wise independent hash functions)**.** Let q be a prime power, let $\beta \in \mathbb{N}$ with $q^\beta \geq t$, and let $\mathcal{H}$ denote the uniform distribution over functions $\mathbb{F}_{q^\beta} \to \mathbb{F}_{q^\beta}$ of the form $h(x) = \sum_{i=0}^{t-1} h_i x^i$. That is, the domain and range are interpreted as the finite field with q^β elements, and the sampled functions are all degree $\leq (t-1)$ polynomials. Then $\mathcal{H}$ is a t-wise independent hash function family.

Observe that sampling according to $\mathcal{H}$ as defined above requires $t\beta \log_2 q$ bits of uniform randomness. By considering the entropy of the resulting distribution $(H(x_1), \ldots, H(x_t))$, this is optimal, as the uniform distribution over $\mathbb{F}_q^{\beta \cdot t}$ has entropy $\beta t \log_2(q)$, and by the data-processing inequality entropy cannot increase.

In particular, if $\mathcal{H}$ only ever output functions represented by polynomials of degree $\leq t'-1$ with $t' < t$, then $\mathcal{H}$ would be supported on a set of size $\leq q^{\beta t'}$, and therefore $\mathcal{H}$ would have entropy at most $\log q^{\beta t'} = \beta t' \log(q)$, a contradiction to the above. Hence, guaranteeing that every hash function has degree $\leq t-1$ as a polynomial over $\mathbb{F}_{q^\beta}$ is optimal.

However, we can also think of the functions as being maps from $\mathbb{F}_q^\beta \to \mathbb{F}_q^\beta$ – that is, inputs and outputs are length-β vectors with coordinates in $\mathbb{F}_q$ – then the functions can be chosen to have smaller degree. That is, one can fix an $\mathbb{F}_q$-linear isomorphism $\mathbb{F}_q^\beta \to \mathbb{F}_{q^\beta}$ such that, under this identification, the function

$$F : \mathbb{F}_{q^\beta} \to \mathbb{F}_{q^\beta}, \quad x \mapsto \sum_{i=0}^{t-1} F_i \cdot x^i$$

is represented by a function $G = (G_1, \ldots, G_\beta) : \mathbb{F}_q^\beta \to \mathbb{F}_q^\beta$ where each G_j has degree $\leq (q-1)\log_q(t)$ [BC12, Section 2]. More precisely, one can bound the degree of each G_j by

$$\max\{|i|_q : F_i \neq 0\} ,$$

where we've defined $|i|_q$ as $\sum_{j=0}^{\beta-1} x_j$ where $i = \sum_{j=0}^{\beta-1} x_j q^j$. In other words, it is the sum of the coordinates in the base-q expansion of i.

Thus, we can have t-wise independent hash functions with degree $(q-1)\log_q(t)$: if $i \leq t-1$, one can verify that $|i|_q \leq (q-1)\log_q(t)$. When $q = 2$, this bound is just $\log_2(t)$. We record this fact below.

Lemma 2.3. Let q be a prime power and let $\beta \in \mathbb{N}$ with $q^\beta \geq t$. There exists a t-wise independent hash function family $\mathcal{H}$ of functions from $\mathbb{F}_q^\beta \to \mathbb{F}_q^\beta$ such that every function $h \in \mathrm{supp}(\mathcal{H})$ has algebraic degree $\leq (q-1)\log_q(t)$.

2.4 The q-ary XOR Lemma

Finally, we record a useful lemma, sometimes referred to as Vazirani's XOR Lemma [CGH$^+$85]. We will use the generalization to larger fields, for which a proof can be found in Rao's exposition of Bourgain's 2-source extractor [Rao07, Lemma 4.2]. In fact, we just need the $\varepsilon = 0$ version of that statement, which results in the following.

Lemma 2.4. Let x be a random vector distributed over $\mathbb{F}_q^n$. Then $x \sim \mathbb{F}_q^n$ if and only if for all $\xi \in \mathbb{F}_q^n \setminus \{0\}$, $\langle \xi, x \rangle \sim \mathbb{F}_q$.

In words: in order to test if a random vector is uniform over $\mathbb{F}_q^n$, it suffices to check that its inner-product with each (fixed) non-zero vector is uniform over $\mathbb{F}_q$.

3 Constructions of Fast Codes with Fast Duals

In this section we concern ourselves with the construction of a "fast good code" with a "fast good dual," as coined by [BR26]. These are pairs of linear codes that are dual to each other, that are both fast, and that are both asymptotically good with a rate-distance tradeoff at the GV-bound. Applications of such a pair of objects include protocols for information-theoretic secure multiparty computation (MPC) and encrypted matrix vector products with asymptotically optimal circuit size, as we discuss further in Section 5.

Recalling from the introduction, such a pair of codes was first given by [BR26], but their construction had a number of downsides, like being restricted to binary alphabets and both codes needing to have rate 1/2. We present a new such construction in Section 3.1 which amends the five drawbacks described in the introduction. The construction relies on a fast *linear uniform-output family* (LUOF), a distribution over matrices A such that xA is uniform for any nonzero x (see Definition 3.3). The properties of this new construction are summarized in the following theorem. When we refer to a map as being *systematic* we mean that it maps an input vector x to an output vector of the form (x, y) or (y, x), i.e. the output vector contains the input vector.

Theorem 3.1. Let q be a prime power and fix $R, \varepsilon \in (0, 1)$ with $\varepsilon < \min\{R, 1-R\}$. There exist uniform families of randomized linear arithmetic circuits $\{C_i : \mathbb{F}_q^{R_i n_i} \to \mathbb{F}_q^{n_i}\}_{i\in\mathbb{N}}$ and $\{C_i^\perp : \mathbb{F}_q^{(1-R_i)n_i} \to \mathbb{F}_q^{n_i}\}_{i\in\mathbb{N}}$, where $(n_i)_{i\in\mathbb{N}}$ is an increasing sequence of positive integers and $(R_i)_{i\in\mathbb{N}}$ is a sequence of real numbers with each $R_i n_i \in \mathbb{N}$, satisfying the following properties.

- **Rate:** $\lim_{i\to\infty} R_i = R$ (and hence $\lim_{i\to\infty}(1-R_i) = 1-R$).
- **Efficient encodability:** For each $i \in \mathbb{N}$, $C_i : \mathbb{F}_q^{R_i n_i} \to \mathbb{F}_q^{n_i}$ is of size $O(n_i)$ with $O(n_i)$ uniformly random field elements. The map is systematic.
- **Dual efficient encodability:** For each $i \in \mathbb{N}$, $C_i^\perp : \mathbb{F}_q^{(1-R_i)n_i} \to \mathbb{F}_q^{n_i}$ of size $O(n_i)$ with $O(n_i)$ uniformly random field elements. The map is systematic.
- **Good distance:** For each $i \in \mathbb{N}$, let $\mathcal{C}_i$ denote the random code with encoding map determined by C_i, where the randomness is over the internal randomness of the circuit. Then $\mathcal{C}_i$ has rate R_i and distance $h_q^{-1}(1-R_i-\varepsilon)$ except with probability $q^{-\varepsilon n_i}$ (GV-bound).
- **Good dual distance:** For each $i \in \mathbb{N}$, let $\mathcal{C}_i^\perp$ denote the random code with encoding map determined by $C_i^\perp$, where the randomness is over the internal randomness of the circuit. Then $\mathcal{C}_i^\perp$ has rate $1-R_i$ and distance $h_q^{-1}(R_i-\varepsilon)$ except with probability $q^{-\varepsilon n_i}$ (GV-bound).
- **Duality:** For each $i \in \mathbb{N}$, the codes $\mathcal{C}_i$ and $\mathcal{C}_i^\perp$ are dual to one another, assuming the encoder circuits use the same randomness.

The distance analysis of these codes is much simpler than that of the construction of [BR26]. This gives a better prospect of generalizing to other properties; to prove that our code construction doesn't just generate a "fast good code" with "fast good dual," but a code with additional interesting properties. We indeed manage to do so, and in Section 3.2 prove that our construction yields a pair of codes that additionally are *t-locally similar* to a random linear code (albeit, at some cost in terms of parameters). Intuitively, this means that any *t-local property* of random linear codes also applies to this pair of codes, which are features that can be ruled out by a small, size t subset of vectors. This includes distance ($t = 1$) and list-of-L-decodability ($t = L+1$). Note that this generalizes the construction from the first subsection, which we leave mostly as a warm-up.

We conclude with the following result, which, to the best of our knowledge, is the first (randomized) construction of a code achieving list-decoding capacity with fast (*linear-size*) encoding. This holds for both the code we construct *and* its dual. In fact, the codes we construct have the best-known list-size of $L = O(1/\varepsilon)$.

Theorem 3.2. Let q be a prime power and fix $R, \varepsilon \in (0,1)$. There is a $c > 0$ such that, assuming $\min\{1-R-c\varepsilon, R-c\varepsilon\} > 0$, the following holds. There exist uniform families of randomized linear arithmetic circuits $\{C_i : \mathbb{F}_q^{R_i n_i} \to \mathbb{F}_q^{n_i}\}_{i\in\mathbb{N}}$ and $\{C_i^\perp : \mathbb{F}_q^{(1-R_i)n_i} \to \mathbb{F}_q^{n_i}\}_{i\in\mathbb{N}}$, where $(n_i)_{i\in\mathbb{N}}$ is an increasing sequence of positive integers and $(R_i)_{i\in\mathbb{N}}$ is a sequence of real numbers with each $R_i n_i \in \mathbb{N}$, satisfying the following properties.

- **Rate:** $\lim_{i\to\infty} R_i = R$ (and hence $\lim_{i\to\infty}(1-R_i) = 1-R$).
- **Efficient encodability:** For each $i \in \mathbb{N}$, $C_i : \mathbb{F}_q^{R_i n_i} \to \mathbb{F}_q^{n_i}$ is of size $O(n_i)$ with $O(n_i)$ uniformly random field elements. The map is systematic.
- **Dual efficient encodability:** For each $i \in \mathbb{N}$, $C_i^\perp : \mathbb{F}_q^{(1-R_i)n_i} \to \mathbb{F}_q^{n_i}$ of size $O(n_i)$ with $O(n_i)$ uniformly random field elements. The map is systematic.
- **Good list-decodability:** For each $i \in \mathbb{N}$, let $\mathcal{C}_i$ denote the random code with encoding map determined by C_i, where the randomness is over the internal randomness of the circuit. Then $\mathcal{C}_i$ has rate R_i and is $(h_q^{-1}(1-R_i-c\varepsilon), \lceil 1/\varepsilon \rceil)$-list-decodable except with probability $q^{-\Omega(n_i)}$ (list-decoding capacity).

- **Good dual list-decodability:** For each $i \in \mathbb{N}$, let $\mathcal{C}_i^\perp$ denote the random code with encoding map determined by $C_i^\perp$, where the randomness is over the internal randomness of the circuit. Then $\mathcal{C}_i^\perp$ has rate $1 - R_i$ and is $(h_q^{-1}(R_i - c\varepsilon), \lceil 1/\varepsilon \rceil)$-list-decodable except with probability $q^{-\Omega(n_i)}$ (list-decoding capacity).
- **Duality:** For each $i \in \mathbb{N}$, the codes $\mathcal{C}_i$ and $\mathcal{C}_i^\perp$ are dual to one another, assuming the encoder circuits use the same randomness.

3.1 Fast Good Code with Fast Good Dual

Our construction for a "fast good code" with "fast good dual" is based on an object known as a linear uniform output family. This is any matrix A which maps any fixed non-zero input vector to a uniformly random vector in the output space.

Definition 3.3. (Linear Uniform Output Family, LUOF) Let A be a random matrix distributed over $\mathbb{F}_q^{k \times r}$. We call A a *linear uniform output family (LUOF)* if, for all $x \in \mathbb{F}_q^k \setminus \{0\}$, we have $xA \sim \mathbb{F}_q^r$.

Classical constructions of LUOF, such as the family of Toeplitz matrices, require circuits of quasilinear size. Here we use the fast (i.e., linear-size universal circuit) LUOF construction of Druk and Ishai [DI14], which works over any finite field $\mathbb{F}_q$ and for any choice of k and r. This LUOF can be used to construct a fast good code: simply use the LUOF $A \in \mathbb{F}_q^{k \times n}$ itself as a generator matrix for a linear code. Since the LUOF is fast, so is the code. The code also is asymptotically good, with rate-distance tradeoff at the GV-bound, which follows in a standard way from the fact that each message is mapped to a uniformly random vector in the output space. We briefly recall this argument as we will use a similar argument for our code constructions.

The probability the code attains some distance δ can then (by a union bound) be upper bounded by the probability that any message attains weight at most δ, times the number of messages. Since each message ends up as a uniformly random output vector, the probability it has weight at most δ is equal to the number of weight at most δ output vectors divided by the size of the output space: $q^{n \cdot h_q(d/n) - n}$, where h_q is the q-ary entropy function. Multiplying by the number of messages q^k gives an upper bound on the probability the code attains distance δ of $q^{n(h_q(\delta) - 1 + k/n)}$, which is $q^{-\Omega(n)}$ (negligible in n) whenever $k/n < 1 - h_q(\delta)$. This tradeoff between the rate $R := k/n$ and distance δ is exactly the q-ary GV-bound.

The fast LUOF of Druk and Ishai thus generates a fast good code. But the authors note that the dual of any code generated by a LUOF likewise is good: it also attains the GV-bound. However, it is not clear what the encoding function of such a dual code looks like, and in particular, whether this dual code is fast. We now give a slightly more complicated code construction which does explicate what the dual code looks like, and indeed lets us guarantee that the dual code is fast.

Instead of having the primal code be generated by a $k \times n$ LUOF, we suppose that we have a $k \times r$ LUOF called A. We then consider the linear maps $x \mapsto (x, xA)$ and $x \mapsto (-xA^\top, x)$. These are implemented by the generator matrices $G = [I_k | A]$ and $H = [-A^\top | I_r]$. Here, I_k and I_r are the $k \times k$ and $r \times r$ identity matrices. Note that G is a $k \times (r + k)$ matrix and H is a $r \times (r + k)$ matrix. Let us write $n := r + k$ for the block length, so that the primal code generated by G has rate k/n and the dual code generated by H has rate $(n - k)/n$.

We now claim that G is a fast good code with a fast good dual H satisfying the properties listed in Theorem 3.1. We spend the rest of this subsection verifying each of the properties listed in this theorem.

Proof of Theorem 3.1. To start, we can choose G to have any rate $R \in (0,1)$, the codes G and H clearly have systematic encoding maps, and G and H generate dual codes as H is the parity-check matrix of G:

$$GH^\top = [I_k \mid A] \begin{bmatrix} -A \\ I_r \end{bmatrix} = -I_k A + A I_r = -A + A = 0 \ .$$

The two harder properties that remain are the fact that G and H admit fast encoding maps (arithmetic circuits of linear size in the input) and attain the GV-bound. Starting with the fast encoding maps, recall that that $xG = (x, xA)$ and that $xH = (-xA^\top, x)$. Copying x and applying a minus sign can of course both be done fast. If we sample the LUOF A according to the construction of [DI14], then computing xA is fast too. From this fact it actually follows that computing $xA^\top$ is fast too:

Lemma 3.4 (Transposition principle [Bor56])**.** There exists a constant $c_4 > 0$ such that for every finite field $\mathbb{F}$ the following holds. Let C be an arithmetic circuit over $\mathbb{F}$ of size t which consists of only addition and scalar multiplication gates and computes the function $f(x) = xA$ (where xA is a matrix-vector product). Then there exists an arithmetic circuit C' of size at most $c_4 t$ that computes the function $f'(x) = xA^\top$.

All that remains is to show that G and H attain the GV-bound. We start with the code G, which we recall encoded a vector x to (x, xA). What we need to show is that the probability that G fails to have distance $h_q^{-1}(1 - R - \varepsilon)$ for some $\varepsilon > 0$ is at most $q^{-\varepsilon n}$, where we recall that n is the block length of G. Since our code is linear, this event is equivalent to G having a codeword of weight at most $h_q^{-1}(1 - R - \varepsilon)$.

There are at most $q^{nh_q(\delta)}$ length n vectors of weight at most δ. Recall the earlier high-level proof that a random linear code attains the GV-bound (in fact, any generator matrix that is a LUOF). There, we could argue that any message vector is mapped to a uniformly random output vector, so that any specific vector $y \in \mathbb{F}_q^n$ is in the code with probability q^{k-n}: there are q^k messages, and each has a probability of q^{-n} of being mapped to a some specific output vector. Applying a union bound over all bad codewords (with weight below the target d) yields an upper bound on the probability that such a code fails to have distance δ:

$$q^{nh_q(\delta)+k-n} = q^{n(h_q(\delta)-1+k/n)} = q^{n(h_q(\delta)-1+R)} = q^{n(1-R-\varepsilon-1+R)} = q^{-\varepsilon n} \ ,$$

where in the last step we substituted in our target distance of $\delta = h_q^{-1}(1 - R - \varepsilon)$. Now, we cannot directly apply this reasoning to our code, as our code does not map each message to a uniformly random output vector. The first k entries of our output will be a copy of the message, and only the last r entries will be uniformly random. However, it is not hard to see that we can derive the same upper bound of $q^{k-n} = q^{-r}$ on the probability that a given vector $y \in \mathbb{F}_q^n$ is in our code $\mathcal{C}$. This then immediately yields the GV-bound by the above derivation.

Recall that $y \in \mathcal{C}$ whenever $yH^\top = 0$ where $H = [-A^\top | I_k]$ is the parity-check matrix of $\mathcal{C}$. Let us split up $y = (y^{(1)}, y^{(2)})$ with $y^{(1)} \in \mathbb{F}_q^k$ and $y^{(2)} \in \mathbb{F}_q^r$. Then we can write

$$\mathbb{P}\left[y \in \mathcal{C}\right] = \mathbb{P}\left[yH^\top = 0\right] = \mathbb{P}\left[y^{(2)} - y^{(1)}A = 0\right] = \mathbb{P}\left[y^{(2)} = y^{(1)}A\right] \leq q^{-r} \ ,$$

where the last step is because A is a LUOF, so that the output $y^{(1)}A$ is uniformly random over $\mathbb{F}_q^r$ if $y^{(1)} \neq 0$; if $y^{(1)} = 0$, then $y^{(2)} \neq 0$ (since $y = (y^{(1)}, y^{(2)}) \neq 0$), so then $\mathbb{P}\left[y^{(2)} = y^{(1)}A\right] = 0$. Thus, in our code we obtain the same (upper bound on the) probability that a given codeword is in our

code as we had above for a random linear code. It follows that we obtain the exact same bound on the probability that our code achieves distance d, and hence we recover the GV-bound.

It remains to prove that the dual code, generated by H with the linear map $x \mapsto (-xA^\top, x)$, also attains the GV-bound. To do this, we prove Lemma 3.5 just below which states that if A is a LUOF, then $-A^\top$ is also a LUOF. The proof that H achieves the GV-bound is then analogous to that of G, establishing the theorem. □

Lemma 3.5. Let $r, k \in \mathbb{N}$. Let A be a random matrix distributed over $\mathbb{F}_q^{k \times r}$. If A is LUOF, then so is $-A^\top$.

Proof. It is immediate that if A is LUOF, then so is $-A$. Thus, it suffices to prove that if A is LUOF, then so is $A^\top$. That is, we must show that for all $x \in \mathbb{F}_q^r \setminus \{0\}$, we have $xA^\top \sim \mathbb{F}_q^k$. By Lemma 2.4, it suffices to prove that for all $\xi \in \mathbb{F}_q^k \setminus \{0\}$, we have $\langle \xi, xA^\top \rangle \sim \mathbb{F}_q$. Now, $\langle \xi, xA^\top \rangle = \langle \xi A, x \rangle$. Since $\xi \neq 0$ and A is LUOF, we know that $\xi A \sim \mathbb{F}_q^r$, so now by the other direction of Lemma 2.4 we have $\langle \xi A, x \rangle \sim \mathbb{F}_q$, as desired. □

3.2 Fast List-Decodable Code with Fast List-Decodable Dual

We now generalize the previous construction to yield not just a "fast good code" with "fast good dual", but codes that are in addition *t-locally similar* to a random linear code. We won't define t-local similarity formally, which would require some additional definitions. Instead, we make use of Proposition II.8 of [MRSY25] (a result which was implicit in earlier works [MRR+24, GM22]) which states that a code is t-locally similar to a random linear code whenever any selection of t linearly independent vectors are in $\mathcal{C}$ with probability at most $q^{-n(1-R)t}$, where R is the rate, q is the field size and n is the block length.[4] We will establish that (a slight generalization of) our code construction satisfies this property.

As explained in the introduction to this section, this roughly implies that our code shares with random linear codes all features that can be ruled out by size t subsets of vectors. Our goal will be to show that our code construction yields a pair of codes that have good list-decodability. Recall that a code $\mathcal{C} \subseteq \mathbb{F}_q^n$ is (ρ, L)-list-decodable if for all $z \in \mathbb{F}_q^n$, $|\{c \in \mathcal{C} : d(c, z) \leq \rho\}| \leq L$. Observe that proving a code $\mathcal{C}$ is (ρ, L)-list-decodable amounts to showing that certain sets of vectors of size L are *not* contained in $\mathcal{C}$: in this specific case, the vectors that must be shown to not lie in $\mathcal{C}$ are all $L+1$-subsets of some Hamming ball of radius ρ. This aligns exactly with the intuitive idea that a local property is one that can be ruled out by a small set of vectors.

Proposition II.9 of [MRSY25] makes this precise by showing that any linear code that is t-locally similar to a random linear code has good list-decodability. We state it in full below.

Proposition 3.6. Let q be a prime power. Let $k, n, L \in \mathbb{N}$, let $R = k/n$ and let $\varepsilon > 1/(L+1)$. For n sufficiently large compared to L, there exists a constant $c > 0$ such that the following holds. If $\rho = h_q^{-1}(1 - R - c\varepsilon)$ and $\mathcal{C} \leq \mathbb{F}_q^n$ is a random code of rate R that is $(L+1)$-locally similar to a random linear code, then with probability at least $1 - q^{-\varepsilon(n - o_{n\to\infty}(1))}$, $\mathcal{C}$ is (ρ, L)-list-decodable.

We note that the maximum radius up to which one can hope to list-decode is $h_q^{-1}(1 - R - \varepsilon)$, and existentially at radius $h_q^{-1}(1 - R - \varepsilon)$ lists of size $O(1/\varepsilon)$ are sufficient. Thus, if we can prove

[4] Note that this result requires $\frac{n}{\log_q n} \geq \omega_{n\to\infty}(q^{2t})$, i.e. the block length needs to be large enough compared to q and t.

that our codes are t-locally similar to random linear codes, they essentially obtain the best-known list-decodability. We now turn to proving this, thereby establishing Theorem 3.2.

We recall once more that to prove t-local similarity it suffices to prove that any choice of $b \leq t$ linearly independent vectors are in the codes with probability at most $q^{-n(1-R)b}$. Our original code construction used a fast LUOF which guaranteed that any fixed non-zero input vector is mapped to a uniformly random output vector. This property is not sufficient for our current purposes, as it doesn't guarantee that multiple distinct inputs are mapped to uniform outputs independently.

To obtain this guarantee, we generalize the notion of a LUOF. Instead of requiring a single non-zero input to be mapped to a uniform random output, we require t linearly independent inputs to be mapped to uniformly and independent outputs.

Definition 3.7. Let $t, r, k \in \mathbb{N}$. Let A be a random matrix distributed over $\mathbb{F}_q^{k \times r}$. We call A a *t-wise linear uniform output family (t-LUOF)* if, for all $x_1, x_2, \ldots, x_t \in \mathbb{F}_q^k$ linearly independent, we have $(x_1 A, x_2 A, \ldots, x_t A) \sim \mathbb{F}_q^{tr}$. That is, the collection of random variables $x_i A$ for $i \in [t]$ are all uniform over $\mathbb{F}_q^r$, and independent.

Remark 3.8. Note that if a random matrix is t-LUOF, then it is also s-LUOF for any $s \leq t$. Indeed, if $(x_1 A, x_2 A, \ldots, x_t A) \sim \mathbb{F}_q^{tr}$, then also $(x_1 A, x_2 A, \ldots, x_s A) \sim \mathbb{F}_q^{sr}$.

In Section 4.2 we will show that the DI construction of a fast LUOF can be generalized to yield a fast t-LUOF as well. For now, we show that our code construction, set up with a t-LUOF instead of a 1-LUOF, gives us t-local similarity. Or more specifically, we prove that our construction yields a fast list-decodable code with fast list-decodable dual, as described in Theorem 3.2.

Proof of Theorem 3.2. We use the codes G and H according to the code construction outlined in Section 3.1, but instead of using the fast 1-LUOF of [DI14], we use the fast t-LUOF which we construct in Section 4.2. We recall that proof of Theorem 3.1 in that same subsection which shows us that these codes are dual, can be of any rates (as long as they sum to 1, of course), and have fast, systematic encoding maps. But now, instead of proving good distance, we need to argue that our codes are t-locally similar to random linear codes, for which we recall it suffices to show that any choice of $b \leq t$ linearly independent vectors are in the codes with probability at most $q^{-n(1-R)b}$. Note that $R = k/n$ and $n = k + r$, so that $n(1 - R) = n(1 - k/n) = n - k = r$. The probability upper bound then becomes q^{-rb}.

We now show that G and H indeed have probability $\leq q^{-rb}$ that any fixed selection of b non-zero and linearly independent vectors is in our code, establishing the required list-decodability by Proposition 3.6. This will require not just that A is t-LUOF, but that $A^\top$ is t-LUOF as well, which we establish in Lemma 3.9. Then, in Lemma 3.10 we show that G satisfies the earlier probability bound. A completely analogous proof establishes this result for H, which in turn completed the proof of our theorem. □

Lemma 3.9. Let $t, r, k \in \mathbb{N}$. Let A be a random matrix distributed over $\mathbb{F}_q^{k \times r}$ which is t-LUOF. Then $A^\top$ is also t-LUOF.

Proof. We must show that for all $\xi_1, \ldots, \xi_t \in \mathbb{F}_q^r$ linearly independent, we have $(\xi_1 A^\top, \ldots, \xi_t A^\top) \sim \mathbb{F}_q^{tk}$. By Lemma 2.4, it suffices to argue that for any $x \in \mathbb{F}_q^{tk} \setminus \{0\}$, we have $\langle x, (\xi_1 A^\top, \ldots, \xi_t A^\top) \rangle \sim \mathbb{F}_q$. Write $x = (x_1, \ldots, x_t)$ with each $x_i \in \mathbb{F}_q^k$, then we need to argue

$$\langle x, (\xi_1 A^\top, \ldots, \xi_t A^\top) \rangle = \sum_{i=1}^{t} \langle x_i, \xi_i A^\top \rangle = \sum_{i=1}^{t} \langle \xi_i, x_i A \rangle$$

is uniform. Fix a basis $\{y_i : i \in [s]\}$ for the space $\text{span}\{x_i : i \in [t]\}$, and note that $s \geq 1$ since we assumed that not all the x_i's are 0. We can (uniquely) write each $x_i = \sum_{j=1}^{s} x_{ij} y_j$ with the $x_{ij} \in \mathbb{F}_q$. So then

$$\sum_{i=1}^{t} \langle \xi_i, x_i A \rangle = \sum_{i=1}^{t} \left\langle \xi_i, \left(\sum_{j=1}^{s} x_{ij} y_j \right) A \right\rangle = \sum_{i=1}^{t} \sum_{j=1}^{s} x_{ij} \langle \xi_i, y_j A \rangle = \sum_{j=1}^{s} \left\langle \sum_{i=1}^{t} x_{ij} \xi_i, y_j A \right\rangle \ .$$

Now, since $y_1, \ldots, y_s$ are linearly independent, we know that $y = (y_1 A, \ldots, y_s A) \sim \mathbb{F}_q^{sr}$. Considering now the vector $\xi = \left(\sum_{i=1}^{t} x_{i1} \xi_i, \ldots, \sum_{i=1}^{t} x_{is} \xi_i \right)$, if we show that it is nonzero, then by Lemma 2.4 we will conclude that

$$\sum_{j=1}^{s} \left\langle \sum_{i=1}^{t} x_{ij} \xi_i, y_j A \right\rangle = \langle \xi, y \rangle \sim \mathbb{F}_q \ ,$$

as required. So, we must show $\xi \neq 0$. That is, we must show that for some $j \in [s]$, we have $\sum_{i=1}^{t} x_{ij} \xi_j \neq 0$. Since $\xi_1, \ldots, \xi_t$ are linearly independent, we have $\sum_{i=1}^{t} x_{ij} \xi_i = 0 \iff x_{1j} = x_{2j} = \cdots = x_{tj} = 0$; this implies that we just must show some $x_{ij} \neq 0$. To see this is true, note that some $x_i \neq 0$, and so since the y_i's form a basis that means some $x_{ij} \neq 0$, as desired. □

Lemma 3.10. Let $b, t, r, k \in \mathbb{N}$ with $b \leq t$, and let A be a random matrix distributed over $\mathbb{F}_q^{k \times r}$ which is t-LUOF. Consider systematic linear code $\mathcal{C} := \{(x, xA) : x \in \mathbb{F}_q^k\}$, which has block-length $n := k + r$. Then, for all $x_1, \ldots, x_b \in \mathbb{F}_q^n$ linearly independent, we have $\mathbb{P}[\forall j \in [b], x_j \in \mathcal{C}] \leq q^{-rb}$.

Proof. Since $\mathcal{C}$ is generated by $G = \begin{bmatrix} I_k \mid A \end{bmatrix}$, it is checked by the matrix $H = \begin{bmatrix} -A^\top \mid I_r \end{bmatrix}$. Note that, for all $x \in \mathbb{F}_q^n$, $xH^\top = x^{(2)} - x^{(1)} A$, where $x = (x^{(1)}, x^{(2)})$ with $x^{(1)} \in \mathbb{F}_q^k$ and $x^{(2)} \in \mathbb{F}_q^r$. Hence,

$$\begin{aligned} \mathbb{P}[\forall j \in [b], x_j \in \mathcal{C}] &= \mathbb{P}\left[\forall j \in [b], x_j H^\top = 0\right] \\ &= \mathbb{P}\left[\forall j \in [b], x_j^{(2)} - x_j^{(1)} A = 0\right] = \mathbb{P}\left[\forall j \in [b], x_j^{(1)} A = x_j^{(2)}\right] \end{aligned} \tag{1}$$

Let us first consider the case that $x_1^{(1)}, \ldots, x_b^{(1)}$ are linearly independent. Then, it follows directly from the fact that A is t-wise LUOF (and hence, b-wise LUOF, since $b \leq t$) that the vector $(x_1^{(1)} A^\top, \ldots, x_b^{(1)} A^\top)$ is distributed uniformly over $\mathbb{F}_q^{br}$, and hence the probability it takes on the value $(x_1^{(2)}, \ldots, x_b^{(2)})$ is q^{-rb}. Hence,

$$(1) = \mathbb{P}\left[(x_1^{(1)} A^\top, \ldots, x_b^{(1)} A^\top) = (x_1^{(2)}, \ldots, x_b^{(2)})\right] = q^{-rb} \ .$$

Let us now consider the case that $x_1^{(1)}, \ldots, x_b^{(1)}$ are linearly dependent. Fix $j^* \in [b]$ minimal such that $x_{j^*}^{(1)} \in \text{span}\{x_1^{(1)}, \ldots, x_{j^*-1}^{(1)}\}$ (note that we could have $j^* = 1$, where we've used the convention that $\text{span}(\emptyset) = \{0\}$). Thus, there exist multipliers $\lambda_1, \ldots, \lambda_{j^*-1} \in \mathbb{F}_q$ such that $x_{j^*}^{(1)} = \sum_{i=1}^{j^*-1} \lambda_i x_i^{(1)}$ (again, if $j^* = 1$, this is just the empty sum, which has value 0). We have

$$(1) \leq \mathbb{P}\left[\forall j \in [j^*], x_j^{(1)} A^\top = x_j^{(2)}\right]$$

We claim the above probability is 0, which certainly establishes the required bound $(1) \leq q^{-rb}$.

To see this term is 0, we write

$$\mathbb{P}\left[\forall j \in [j^*],\ x_j^{(1)} A^\top = x_j^{(2)}\right]$$
$$= \mathbb{P}\left[x_{j^*}^{(1)} A^\top = x_{j^*}^{(2)} \middle| \forall i < j^*,\ x_i^{(1)} A^\top = x_i^{(2)}\right] \cdot \mathbb{P}\left[\forall i < j^*,\ x_i^{(1)} A^\top = x_i^{(2)}\right]$$

Note that the event $\forall i < j^*,\ x_i^{(1)} A^\top = x_i^{(2)}$ does indeed have nonzero probability: if $j^* \geq 2$, as the vectors $x_1^{(1)}, \ldots, x_{j^*-1}^{(1)}$ are linearly independent and $A^\top$ is (j^*-1)-LUOF, the above argument shows this event has probability $q^{-r(j^*-1)}$, while if $j^* = 1$ we condition on the empty event, which has probability 1.

Now, conditioned on the event $\forall i < j^*,\ x_i^{(1)} A^\top = x_i^{(2)}$, we have

$$x_{j^*}^{(1)} A^\top = \left(\sum_{i=1}^{j^*-1} \lambda_i x_i^{(1)}\right) A^\top = \sum_{i=1}^{j^*-1} \lambda_i\ x_i^{(1)} A^\top = \sum_{i=1}^{j^*-1} \lambda_i x_i^{(2)}\ .$$

Thus, if $x_{j^*}^{(1)} A^\top = x_{j^*}^{(2)}$, then also $\sum_{i=1}^{j^*-1} \lambda_i x_i^{(2)} = x_{j^*}^{(2)}$, which then means that

$$x_{j^*} = (x_{j^*}^{(1)}, x_{j^*}^{(2)}) = \sum_{i=1}^{j^*-1} \lambda_i (x_i^{(1)}, x_i^{(2)}) = \sum_{i=1}^{j^*-1} \lambda_i x_i\ .$$

Since the $x_1, \ldots, x_b$ were assumed linearly independent, this cannot happen. So the probability is 0, as desired. □

4 Fast Bounded-Independence Functions

In this section we extend previous results from [IKOS08, DI14] on hash functions and linear codes by considering a higher independence parameter t.

4.1 Fast t-Wise Independent Hash Functions

In this section we prove that, with minor modifications, the IKOS construction [IKOS08] can yield t-wise independent hash functions. This generalized variant is still fast as long as t is constant. We summarized the IKOS construction for pairwise independent hash functions ($t = 2$) in Section 1.2. We now briefly recall that the construction consisted first of an encoding step (determined by code with very good distance over a larger alphabet); then, constant-sized hash functions are applied to the coordinates of the encoding; lastly, an extraction step (implemented by the transpose of a generator matrix for a good code) is applied.

To generalize this to t-wise independence, the first natural change that one must make is that the small, constant-sized hashes $h_1, \ldots, h_m$ should now be sampled so as to be t-wise independent (where m is the block-length of the first code), which we can naturally do (and, assuming $t = O(1)$, this will only affect the circuit-size by a constant factor).

Now, recalling the proof of pairwise independence for the IKOS construction, the good distance of the first code guaranteed that, in many coordinates, the encodings of the two distinct inputs will be different. That is, if the code's distance was $1 - \varepsilon$, that exactly means that in at least a $1 - \varepsilon$ fraction of coordinates, the encodings will differ.

For t-wise independence, we are given t inputs $x_1, \ldots, x_t$, and in order to use the t-wise independence of the little hashes $h_1, \ldots, h_m$, we need many coordinates $\ell \in [m]$ in which *all* the encodings differ. In this case, an averaging argument will establish this holds in at least a $1 - \binom{t}{2} \cdot \varepsilon$ fraction of coordinates. Thus, so long as $\varepsilon > 0$ is chosen small enough (which can be done at the cost of increasing the alphabet size for the initial code), the argument goes through.

We now formally prove the construction works. We will use the following codes from [DI14]:

Theorem 4.1 ([DI14, Theorem 2])**.** For every prime power q and $0 < \delta < 1$ there exists $0 < \rho < 1$, $\beta \in \mathbb{N}$ and a family of $\mathbb{F}_q$-additive codes $\mathcal{D}_k \leq (\mathbb{F}_q^\beta)^m$ with rate $R = \frac{k}{\beta m} \geq \rho$ and minimum β-distance $\geq \delta$. Recall this means that, for all $c \in \mathcal{D}_k \setminus \{0\}$, we have $\mathrm{wt}_\beta(c) \geq \delta$.

Furthermore, the encoding map $E_{\mathcal{D}_k}$ can be computed by a uniform family of $O(k)$-size arithmetic circuits over $\mathbb{F}_q$.

Remark 4.2. Theorem 4.1 above promises codes of rate at least some value ρ; however, in some cases we will in fact require codes of a smaller rate, while preserving the same distance and running time. We can easily arrange for this by, say, zeroing out some coordinates from the messages. E.g., if $\mathcal{D}_k$ has dimension k and we'd instead like a code with dimension $k' < k$, we just append $k - k'$ 0's to the message $m \in \mathbb{F}_q^{k'}$ prior to encoding via $E_{\mathcal{D}_k}$.

Theorem 4.3. Let $k, r \in \mathbb{N}$ be growing parameters, fix $t \in \mathbb{N}$, and let q be a prime power. There exists an arithmetic circuit $C_{k,r} : \mathbb{F}_q^k \to \mathbb{F}_q^r$ of size $O(k + r)$ with $O(k + r)$ uniformly random field elements such that, over the uniformly random field elements, the circuit implements a t-wise independent hash function of degree $(q - 1) \log_q(t)$.

Proof. We first describe the construction. Fix constants $0 < \varepsilon, \delta_2 < 1$ with $1 - \binom{t}{2} \cdot \varepsilon + \delta_2 > 1$, and fix $\beta \in \mathbb{N}$ large enough so that the following hold. Let $\mathcal{C}_1 \subseteq \left(\mathbb{F}_q^\beta\right)^m$ and $\mathcal{C}_2 \subseteq \left(\mathbb{F}_q^\beta\right)^m$ be $\mathbb{F}_q$-additive codes of dimension k and r, respectively, and with distance $1 - \varepsilon$ and δ_2, respectively. Such codes are promised to exist by Theorem 4.1 (along with Remark 4.2 in order to reduce the codes' dimensions, if required). Let G_1 and G_2 denote generator matrices for $\mathcal{C}_1$ and $\mathcal{C}_2$, respectively. Let $C_1 : \mathbb{F}_q^k \to \mathbb{F}_q^{m \cdot \beta}$ be the (uniform) arithmetic circuit of size $O(m)$ implementing G_1, and let $C_2 : \mathbb{F}_q^{m \cdot \beta} \to \mathbb{F}_q^r$ be the (uniform) arithmetic circuits of size $O(m)$ implementing $G_2^\top$. That such circuits exist follows from Theorem 4.1 and Lemma 3.4.

Lastly, the circuit samples $t \cdot \beta \cdot m = O(k+r)$ uniformly random symbols which we take to describe m t-wise independent functions $h_1, \ldots, h_m : \mathbb{F}_q^\beta \to \mathbb{F}_q^\beta$. We remark that, thanks to Proposition 2.2, we can indeed sample such constant-sized t-wise hash functions. We imagine composing these functions h_i into the map

$$h : \left(\mathbb{F}_q^\beta\right)^m \to \left(\mathbb{F}_q^\beta\right)^m \ , \quad h\left(x^{(1)}, \ldots, x^{(m)}\right) = \left(h_1(x^{(1)}), \ldots, h_m(x^{(m)})\right) \ .$$

As each h_ℓ can be implemented by a uniform, randomized arithmetic circuit of size $O_{\beta,t}(1)$, h can be implemented by a uniform, randomized arithmetic circuit of size $O(m)$; call this circuit B. Note that B then requires $O(k + r)$ uniformly random field elements. Defining the circuit $C_{k,r}$ now as $C_2 \circ B \circ C_1$, it is immediate that it is uniform and of size $O(m) = O(k+r)$, with $O(k+r)$ uniformly random field elements. Additionally, by Lemma 2.3 we can enforce that all the h_ℓ's have algebraic degree $\leq (q - 1) \log_q(t)$, and so since the other steps are linear, the composite function also has algebraic degree $\leq (q - 1) \log_q(t)$.

Let us write $H : \mathbb{F}_q^k \to \mathbb{F}_q^r$ to denote the function defined by the circuit $C_{k,r}$. We refer the reader to Figure 1 which gives a schematic of H. Recall that H takes $O(k + r)$ uniformly random field

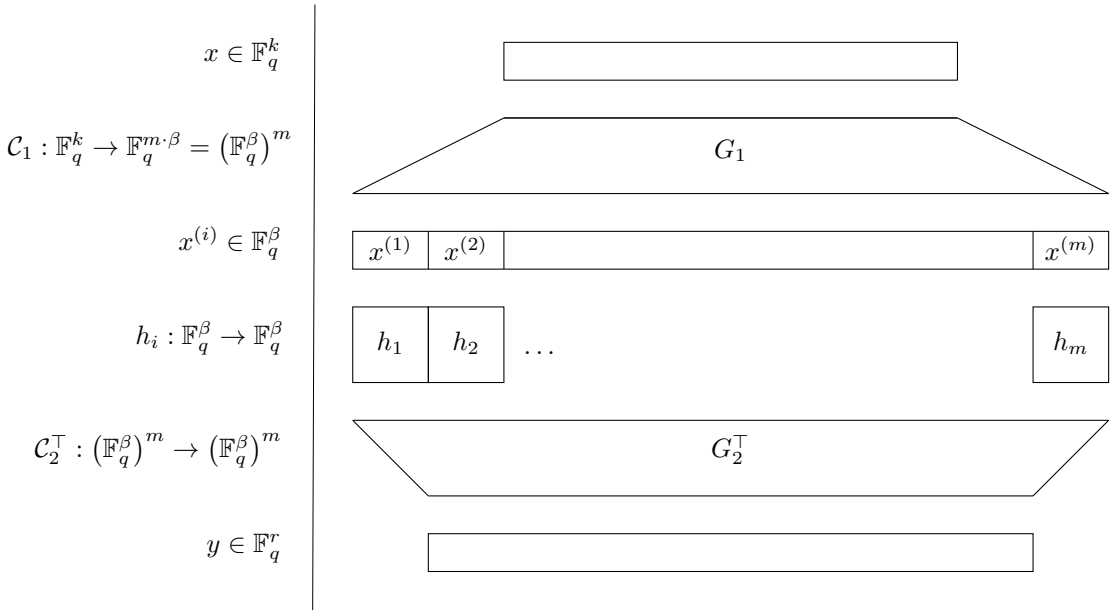


Figure 1: A schematic of the IKOS construction for t-wise independent hash functions. The functions h_i are constant size t-wise independent hash functions which we sample. $\mathcal{C}_1$ is a code with generator matrix G_1 and $\mathcal{C}_2$ is a code with generator matrix G_2.

elements. What we want to prove is that, over these random values (which sample the m t-wise independent hash functions $h_1, \ldots, h_m : \mathbb{F}_q^\beta \to \mathbb{F}_q^\beta$), H is a t-wise independent hash function family. That is, given distinct input vectors $x_1, \ldots, x_t \in \mathbb{F}_q^k$, over the randomness we have that

$$z := (H(x_1), \ldots, H(x_t)) \sim \mathbb{F}_q^{rt} \ .$$

To do this, by Lemma 2.4 it suffices to consider any $y \in \mathbb{F}_q^{tr} \setminus \{0\}$, and to prove that $\langle y, z \rangle \sim \mathbb{F}_q$. Write $y = (y_1, \ldots, y_t)$ with each $y_i \in \mathbb{F}_q^r$, and let $h : \left(\mathbb{F}_q^\beta\right)^m \to \left(\mathbb{F}_q^\beta\right)^m$ be the function determined by sampling the m individual t-wise independent hash function h_i. Then

$$\langle z, y \rangle = \sum_{i=1}^{t} \langle y_i, H(x_i) \rangle = \sum_{i=1}^{t} \left\langle y_i, h(x_i G_1) G_2^\top \right\rangle = \sum_{i=1}^{t} \langle y_i G_2, h(x_i G_1) \rangle \ . \tag{2}$$

Let $c_i = x_i G_1$ and $d_i = y_i G_2$, and note that $c_1, \ldots, c_t$ are distinct codewords in $\mathcal{C}_1$ and $d_1, \ldots, d_t$ are codewords in $\mathcal{C}_2$ which are not all 0. Without loss of generality, assume $d_1 \neq 0$. Write $c_i = (c_i^{(1)}, \ldots, c_i^{(m)})$ and $d_i = (d_i^{(1)}, \ldots, d_i^{(m)})$ with each $c_i^{(\ell)}, d_i^{(\ell)} \in \mathbb{F}_q^\beta$. We can now write

$$(2) = \sum_{\ell=1}^{m} \sum_{i=1}^{t} \left\langle d_i^{(\ell)}, h_\ell(c_i^{(\ell)}) \right\rangle$$

We claim there exists $\ell^* \in [m]$ for which

(a) $d_1^{(\ell^*)} \neq 0$, and

(b) the vectors $c_1^{(\ell^*)}, \ldots, c_t^{(\ell^*)} \in \mathbb{F}_q^\beta$ are *distinct*.

We establish this claim now. Since $\mathcal{C}_2$ has distance $\geq \delta_2$, condition (a) holds for at least a δ_2 fraction of $\ell \in [m]$. As for the second condition, consider the $\binom{t}{2}$ codewords $c_{ij} = c_i - c_j$ for $1 \leq i < j \leq t$. Since $\mathcal{C}_1$ has distance $1 - \varepsilon$, we have that $c_{ij}^{(\ell)} \neq 0$ for at least a $1 - \varepsilon$ fraction of $\ell \in [m]$. Thus, we have $c_{ij}^{(\ell)} = 0$ for at most an ε-fraction of $\ell \in [m]$, implying $c_{ij}^{(\ell)} = 0$ for *some* $1 \leq i < j \leq t$ for at most a $\binom{t}{2} \cdot \varepsilon$ fraction of $\ell \in [m]$. Therefore, condition (b) holds for at least a $1 - \binom{t}{2} \cdot \varepsilon$ fraction of $\ell \in [m]$. Since $1 - \binom{t}{2} \cdot \varepsilon + \delta_2 > 1$ by assumption, by the pigeonhole principle some $\ell \in [m]$ must exist satisfying conditions (a) and (b).

So, fix such an ℓ^*. Since the $h_1, \ldots, h_m$ are sampled independently, if we prove that $\sum_{i=1}^t \left\langle d_i^{(\ell^*)}, h_{\ell^*}(c_i^{(\ell^*)}) \right\rangle \sim \mathbb{F}_q$, then additionally $\sum_{\ell=1}^m \sum_{i=1}^t \left\langle d_i^{(\ell)}, h_\ell(c_i^{(\ell)}) \right\rangle \sim \mathbb{F}_q$, as required. To this end, without loss of generality assume there is $1 \leq s \leq t$ such that $d_i^{(\ell)} \neq 0$ iff $i \leq s$. So then

$$\sum_{i=1}^{t} \left\langle d_i^{(\ell^*)}, h_\ell(c_i^{(\ell^*)}) \right\rangle = \sum_{i=1}^{s} \left\langle d_i^{(\ell^*)}, h_{\ell^*}(c_i^{(\ell^*)}) \right\rangle \ .$$

Since the vectors $c_1^{(\ell^*)}, \ldots, c_s^{(\ell^*)}$ are distinct (and $s \leq t$), the t-wise independence of h_{ℓ^*} guarantees the values $h_{\ell^*}(c_1^{(\ell^*)}), \ldots, h_{\ell^*}(c_s^{(\ell^*)})$ are independent and uniform, which guarantees that $\sum_{i=1}^s \left\langle d_i^{(\ell^*)}, h_{\ell^*}(c_i^{(\ell^*)}) \right\rangle \sim \mathbb{F}_q$, as required. □

Remark 4.4. In Theorem 4.3, we argued that we can achieve $O(k + r)$ circuit size for constant values of t. Naturally, the constant hidden in the big-O notation depends on the choice of t. An inspection of the proof shows that the dependence is roughly $\Theta(t^7)$. Firstly, in order to guarantee $1 - \binom{t}{2} \cdot \varepsilon$ is not too small, we require $\varepsilon = O(1/t^2)$. The codes promised by Theorem 4.1 can be verified to require $\beta = O(1/\varepsilon^3)$ (this itself stems from using the best-known explicit constructions of lossless expander graphs, whose degree are of this order). Thus, in the end we require $\beta = \Omega(t^6)$, and since the total size is about $t \cdot \beta$, we get $\Theta(t^7)$. In particular, with $t = \mathrm{poly}(\log(k+r))$ we could still achieve size $\tilde{O}(k + r)$. This means that when $t = \log^{o(1)} k$, we get better asymptotic size than an FFT-based implementation of a hash function using degree-t polynomials.

Remark 4.5. We remark that having smaller algebraic degree is not possible, at least in the case $q = 2$. Here, we use the following standard fact (see, e.g., [HHL19, Section 3.1]): if $f : \mathbb{F}_2^\beta \to \mathbb{F}_2$ is a polynomial map with algebraic degree at most d and $d < \beta$, then for any $x_0, x_1, \ldots, x_{d+1} \in \mathbb{F}_2^\beta$, we have $\sum_{S \subseteq [d+1]} f\left(x_0 + \sum_{i \in S} x_i\right) = 0$. That is: the sum over any $(d+1)$-dimensional affine space is 0.

Now, suppose one had $\mathcal{H}$ supported on maps $G = (G_1, \ldots, G_\beta) : \mathbb{F}_2^\beta \to \mathbb{F}_2^\beta$, where each G_j is of degree d with $2^{d+1} \leq t$. Then, choose $x_1, \ldots, x_{d+1} \in \mathbb{F}_2^\beta$ linearly independent (this is possible since $d < \beta$, i.e., $d + 1 \leq \beta$), and x_0 arbitrarily; say, $x_0 = 0$. The linear independence implies the values $\sum_{i \in S} x_i$ are distinct for over $S \subseteq [d + 1]$. Then we know that, no matter what G is sampled, we

have

$$\sum_{S\subseteq[d+1]} G\left(\sum_{i\in S} x_i\right) = \sum_{S\subseteq[d+1]} \left(G_1\left(\sum_{i\in S} x_i\right), \ldots, G_\beta\left(\sum_{i\in S} x_i\right)\right)$$
$$= \left(\sum_{S\subseteq[d+1]} G_1\left(\sum_{i\in S} x_i\right), \ldots, \sum_{S\subseteq[d+1]} G_\beta\left(\sum_{i\in S} x_i\right)\right) = (0,\ldots,0) = 0 \ .$$

Thus, we can find $2^{d+1} \leq t$ distinct input values such that, after revealing the hash value on the first $2^{d+1}-1$ of them, the last value is determined (as the sum of the previous values). So the distribution cannot define a t-wise independent hash function family.

4.2 Fast Bounded-Independence LUOF

Recall the definitions of a linear uniform output family (LUOF) (Section 3.1) and a t-wise LUOF (Section 3.2). The t-wise LUOF can be thought of as a linear analogue of t-wise independent hash functions. Of course, when restricting to a linear function, one cannot hope to achieve a t-wise independent hash function, as any distinct yet linearly dependent inputs will be mapped to linearly dependent outputs (and hence not stochastically independent). This motivated us to define a t-LUOF as any matrix which maps t distinct *linearly independent* non-zero inputs to uniformly random and independent outputs.

Our goal in this subsection is to give a construction for a fast t-wise LUOF. Recall that we required this for the fast list-decodable codes with fast list-decodable duals of Section 3.2. Akin to how we showed that the fast t-wise independent hash function by generalizing the IKOS construction for fast pairwise independent hash function (assuming one uses a code with large enough distance), we will generalize the fast LUOF construction of Druk and Ishai [DI14] to give a fast t-LUOF. Here also this will rely on the codes from Theorem 4.1 having sufficiently large distance. We remark that the encoding circuits for the codes we construct only have linear size if $q = O(1)$ (see Remark 4.7).

Theorem 4.6. Let $k, r \in \mathbb{N}$ be growing parameters, fix $t \in \mathbb{N}$, and let q be a prime power. There exists an arithmetic circuit $C_{k,r} : \mathbb{F}_q^k \to \mathbb{F}_q^r$ of size $O(k+r)$ with $O(k+r)$ uniformly random field elements such that, over the $O(k+r)$ uniformly random field elements, the circuit implements a t-LUOF.

Proof. We first provide the construction, which is very similar to that given in the proof of Theorem 4.3. Fix $0 < \varepsilon, \delta_2 < 1$ with $1 - q^t \cdot \varepsilon + \delta_2 > 1$. Fix $\beta \in \mathbb{N}$ with $\beta \geq t$ large enough so that the following hold. We first choose codes $\mathcal{C}_1, \mathcal{C}_2$ exactly as in Theorem 4.3, which we now recall for the reader's convenience.

Let $\mathcal{C}_1 \subseteq \left(\mathbb{F}_q^\beta\right)^m$ and $\mathcal{C}_2 \subseteq \left(\mathbb{F}_q^\beta\right)^m$ be $\mathbb{F}_q$-additive codes of dimension k and r, respectively, and with distance $1-\varepsilon$ and δ_2, respectively. Such codes are promised to exist by Theorem 4.1 (along with Remark 4.2 in order to reduce the codes' dimensions, if required). Let G_1 and G_2 denote generator matrices for $\mathcal{C}_1$ and $\mathcal{C}_2$, respectively. Additionally, from Theorem 4.1 and Lemma 3.4 it holds that G_1 and $G_2^\top$ can both be implemented by uniform arithmetic circuits of size $O(m) = O(k+r)$; call these circuits C_1 and C_2, respectively.

Lastly, the circuit samples $m \cdot \beta^2 = O(r+k)$ uniformly random symbols which we take to describe m random matrices $F_1, \ldots, F_m \in \mathbb{F}_q^{\beta\times\beta}$. Let $F \in \mathbb{F}_q^{\beta m \times \beta m}$ denote the block-diagonal matrix with

the F_i's as blocks, that is,

$$F = \begin{bmatrix} F_1 & 0 & \cdots & 0 \\ 0 & F_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & F_m \end{bmatrix} .$$

Observe that the map $\left(\mathbb{F}_q^\beta\right)^m \to \left(\mathbb{F}_q^\beta\right)^m$ obtained by mapping $((x^{(1)}, \ldots, x^{(m)}) \mapsto (x^{(1)}F_1, \ldots, x^{(m)}F_m)$ can be implemented by a uniform randomized arithmetic circuit of size $O(m \cdot \beta^2) = O(r + k)$ with the F_i's as above; let B denote this circuit. The final circuit $C_{k,r}$ is then defined as $C_2 \circ B \circ C_1$. That they have size $O(k + r)$ follows from the given construction. The circuit samples $O(r + k)$ uniformly random field elements to implement F.

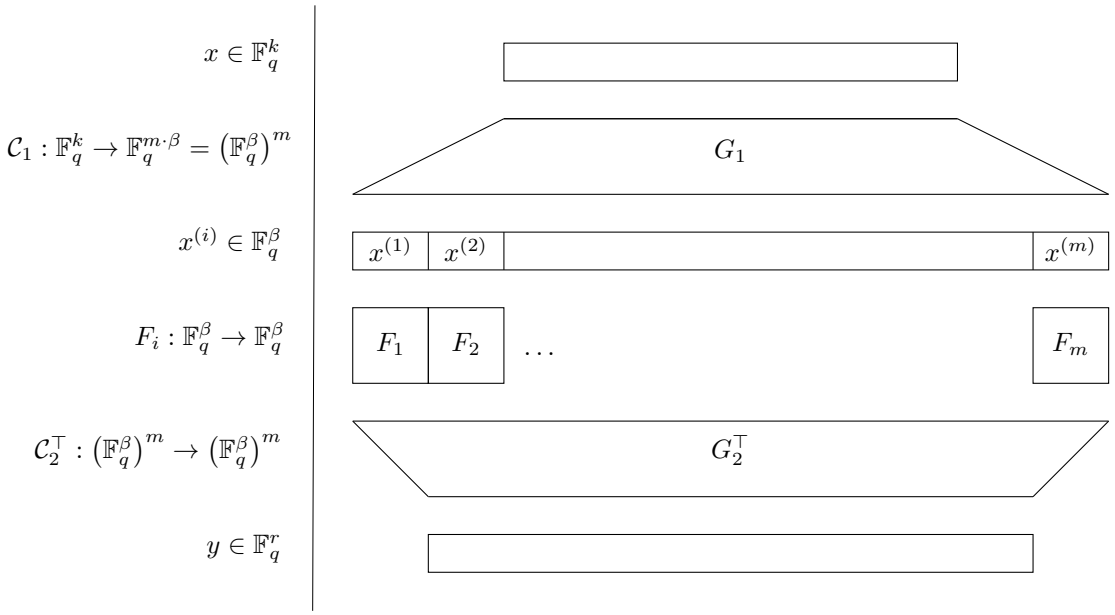


Figure 2: A schematic of the IKOS construction for t-linear uniform output families (t-LUOF). The functions F_i are constant size random matrices which we sample. $\mathcal{C}_1$ is a code with generator matrix G_1 and $\mathcal{C}_2$ is a code with generator matrix G_2.

Let us write $A = G_1 F G_2^\top$ to denote the function defined by the circuit $C_{k,r}$. We refer the reader to Figure 2 which gives a schematic of A. What we want to prove is that, over the random field elements sampled by the circuit to implement F, A is a t-LUOF. Let $x_1, \ldots, x_t \in \mathbb{F}_q^k$ be linearly independent. By Lemma 2.4, it suffices to consider a nonzero $y \in \mathbb{F}_q^{tr}$, and prove that $\langle y, (Ax_1, \ldots, Ax_t)\rangle \sim \mathbb{F}_q$ over the $O(k + r)$ uniformly random field elements. Write $y = (y_1, \ldots, y_t)$ with each $y_i \in \mathbb{F}_q^r$. Then

$$\langle y, (x_1 A, \ldots, x_t A) = \sum_{i=1}^{t} \left\langle y_i, x_i G_1 F G_2^\top \right\rangle = \sum_{i=1}^{t} \langle y_i G_2, x_i G_1 F \rangle \ .$$

Put $d_i := y_i G_2$ and $c_i := x_i G_1$ for all $i \in [t]$. Note that the vectors $c_1, \ldots, c_t \in \mathcal{C}_1$, and that they are all linearly independent. We additionally have that $d_1, \ldots, d_t \in \mathcal{C}_2$, and they are not all 0. Without loss of generality, assume that $d_1 \neq 0$. For each $i \in [t]$, write $d_i = (d_i^{(1)}, \ldots, d_i^{(m)})$ and $c_i = (c_i^{(1)}, \ldots, c_i^{(m)})$ with each $d_i^{(\ell)}, c_i^{(\ell)} \in \mathbb{F}_q^\beta$. Then

$$\sum_{i=1}^{t} \langle y_i G_2, x_i G_1 F \rangle = \sum_{\ell=1}^{m} \sum_{i=1}^{t} \left\langle d_i^{(\ell)}, c_i^{(\ell)} F_\ell \right\rangle$$

We claim there exists an $\ell^* \in [m]$ for which:

(a) $d_1^{(\ell^*)} \neq 0$, and

(b) the vectors $c_1^{(\ell^*)}, \ldots, c_t^{(\ell^*)} \in \mathbb{F}_q^\beta$ are *linearly independent*.

To see this, since $\mathcal{C}_2$ has distance $\geq \delta_2$, condition (a) holds for at least a δ_2 fraction of $\ell \in [m]$. Considering now condition (b), for each $\lambda \in \mathbb{F}_q^t \setminus \{0\}$, we have that $c_\lambda = \sum_{i=1}^t \lambda_i c_i$ is a nonzero codeword (that they are nonzero uses the assumption that the c_i's are linearly independent). Thus, for at least a $1 - \varepsilon$ fraction of $\ell \in [m]$, we have $c_\lambda^{(\ell)} \neq 0$, or in other words, for at most an ε fraction of $\ell \in [m]$ we have $c_\lambda^{(\ell)} = 0$. By a union bound, the fraction of $\ell \in [m]$ for which $c_\lambda^{(\ell)} = 0$ for *some* $\lambda \in \mathbb{F}_q^t \setminus \{0\}$ is at most $q^t \cdot \varepsilon$. Hence, condition (b) holds for at least a $1 - q^t \cdot \varepsilon$ fraction of $\ell \in [m]$. Since we assumed $1 - q^t \cdot \varepsilon + \delta_2 > 1$, we find that the required $\ell^* \in [m]$ exists.

Fix one such $\ell^* \in [m]$. Since the $F_1, \ldots, F_m$ are independent, if we prove that $\sum_{i=1}^t \left\langle d_i^{(\ell^*)}, c_i^{(\ell^*)} F_{\ell^*} \right\rangle \sim \mathbb{F}_q$, then also $\sum_{\ell=1}^m \sum_{i=1}^t \left\langle d_i^{(\ell)}, c_i^{(\ell)} F_\ell \right\rangle \sim \mathbb{F}_q$, as desired. To prove $\sum_{i=1}^t \left\langle d_i^{(\ell^*)}, c_i^{(\ell^*)} F_{\ell^*} \right\rangle \sim \mathbb{F}_q$, suppose without loss of generality that we have $d_1^{(\ell^*)}, \ldots, d_s^{(\ell^*)} \neq 0$ and $d_{s+1}^{(\ell^*)} = \cdots = d_t^{(\ell^*)} = 0$. Then

$$\sum_{i=1}^{t} \left\langle d_i^{(\ell^*)}, c_i^{(\ell^*)} F_{\ell^*} \right\rangle = \sum_{i=1}^{s} \left\langle d_i^{(\ell^*)}, c_i^{(\ell^*)} F_{\ell^*} \right\rangle \ .$$

Now, since the $c_1^{(\ell^*)}, \ldots, c_s^{(\ell^*)}$ are linearly independent, over the randomness of F_{ℓ^*}, the vectors $c_1^{(\ell^*)} F_{\ell^*}, \ldots, c_s^{(\ell^*)} F_{\ell^*}$ are independent and uniform over $\mathbb{F}_q^\beta$. Therefore, applying Lemma 2.4, the inner products $\left\langle d_i^{(\ell^*)}, c_i^{(\ell^*)} F_{\ell^*} \right\rangle$ are independent and uniform for $i = 1, \ldots, s$, so the sum $\sum_{i=1}^s \left\langle d_i^{(\ell^*)}, c_i^{(\ell^*)} F_{\ell^*} \right\rangle$ is uniform over $\mathbb{F}_q$, as desired. □

Remark 4.7. Again, as in Remark 4.4 we can roughly say that since we require $\varepsilon < q^{-t}$ and $\beta > 1/\varepsilon^3$, we at the very least require $\beta > q^{3t}$, showing that the constant in the big-O notation for the circuits in Theorem 4.6 is at least $q^{O(t)}$. In particular, this forces both q and t to be constant to obtain a meaningful bound.

5 Cryptographic Applications

In this section we discuss several cryptographic applications of fast bounded-independence functions, most of which also require fast duals.

5.1 Fast Information-Theoretic MPC

We start by applying asymptotically good fast codes with fast duals towards secure multiparty computation (MPC) in the client-server model. For constant-size functions, our protocol is perfectly secure against almost 1/2 of the n servers, and has total circuit complexity of $O(n)$. This should be contrasted with similar protocols based on algebraic MPC-friendly codes, such as Reed-Solomon codes, whose circuit complexity is (at least) quasi-linear in n.

Theorem 5.1 (Near-optimal perfect MPC for constant-size functions)**.** Let m be a constant number of clients, X and Y finite input and output domains, and $\varepsilon > 0$ an arbitrarily small constant. For every (finite) $f : X^m \to Y^m$ and $\varepsilon > 0$, there is an m-client, n-server protocol for f with the following features.

- The protocol has perfect (information-theoretic) security against a passive adversary corrupting (at most) one client and $t = (1/2 - \varepsilon) \cdot n$ of the n servers.
- The total computational complexity of all parties, measured by Boolean circuit size, is $O(n)$.

The above result is essentially the best one could hope for in the case of constant-size functions. First, the security threshold is nearly optimal, since a similar protocol with security threshold $t > n/2$ would imply information-theoretic oblivious transfer. Second, the circuit complexity is asymptotically optimal, since if a client sends messages to $o(n)$ servers, the adversary can corrupt all of these servers and learn the client's input. (In the perfect security case, this holds even in random access models where we only charge for messages actually sent.)

Protocols in the above client-server model can be used as a building block for other cryptographic applications. For example, the above result can be used to obtain an *oblivious transfer combiner* with similar near-optimality features using the MPC-based approach from [HIKN08].

An OLE protocol. We will start with the simpler special case of (1) $m = 2$ clients, (2) the function $f = \mathsf{OLE}$ that takes one bit x from a receiver client, two bits (y, z) from a sender client and delivers $xy + z$ (over $\mathbb{F}$) to the receiver, and (3) sub-optimal but constant fractional security threshold $t = \Omega(n)$. This case already captures the core difficulty of the problem. We will later outline how to bootstrap from this case to general functions and near-optimal security threshold using standard techniques. In the following $\mathbb{F}$ can be any finite field, but the case $\mathbb{F} = \mathbb{F}_2$ is sufficient for our purposes.

Proposition 5.2. Let $f : \mathbb{F} \times \mathbb{F}^2 \to \mathbb{F}$ be the OLE functionality, which takes $x \in \mathbb{F}$ from a Receiver client and $y, z \in \mathbb{F}$ from a Sender client, and delivers $f(x, (y, z)) = xy + z$ to Receiver. Let $\mathcal{C} \leq \mathbb{F}^{n+1}$ and its dual $\mathcal{C}^\perp \leq \mathbb{F}^{n+1}$ be linear codes with coordinates indexed $0, 1, \ldots, n$. Assume the following:

1. $\mathcal{C}$ has minimum distance $\mathsf{dist} > t + 1$.
2. $\mathcal{C}^\perp$ has minimum distance $\mathsf{dist}^\perp > t + 1$.
3. Both $\mathcal{C}$ and $\mathcal{C}^\perp$ have encoder circuits, E and $E^\perp$, of size s.

Then, there exists a perfectly secure protocol that computes f against any passive adversary corrupting up to t servers and at most one client. Furthermore, the total arithmetic circuit size of all parties is $O(n + s)$.

To prove the above proposition, we start with a lemma that applies the well-known connection between linear error correcting codes and secret sharing [Mas95] while respecting the circuit complexity of the encoder. See Appendix A.1 for a proof.

Lemma 5.3 (Secret sharing from linear codes)**.** Let $\mathcal{C} \leq \mathbb{F}^{n+1}$ be a linear code with coordinates indexed $0, 1, \ldots, n$ and dual distance $\mathsf{dist}^{\perp} \geq 2$. Let $E : \mathbb{F}^k \to \mathbb{F}^{n+1}$ be an arithmetic encoder circuit for $\mathcal{C}$ of size s. Then, the secret sharing scheme that shares a secret $\sigma \in \mathbb{F}$ into n shares by sampling a uniformly random codeword $c \in \mathcal{C}$ subject to $c_0 = \sigma$ and dropping c_0 satisfies the following:

1. Perfect privacy: Any subset of shares $T \subset \{1, \ldots, n\}$ of size $|T| \leq \mathsf{dist}^{\perp} - 2$ perfectly hides σ.
2. Linear complexity: The generation of the shares from σ can be computed by a randomized arithmetic circuit of size at most $s + n$.

We are now ready to describe the protocol establishing Proposition 5.2, using a simple client-server instance of MPC from "multiplicative" secret sharing [CDM00].

Protocol 1 (Information-Theoretic OLE over $\mathbb{F}$)**.**

1. **Setup:** Client 1 (Receiver) holds input $x \in \mathbb{F}$. Client 2 (Sender) holds inputs $y, z \in \mathbb{F}$. There are n servers. A passive adversary may corrupt at most one client and t servers.
2. **Receiver input sharing:**
   (a) Receiver samples a uniformly random codeword $c \in \mathcal{C}$ subject to the constraint $c_0 = x$. This is done using Lemma 5.3.
   (b) Receiver sends share c_i to Server i (for $i = 1, \ldots, n$).
3. **Sender input sharing:**
   (a) Sender samples a uniformly random codeword $d \in \mathcal{C}^{\perp}$ subject to the constraint $d_0 = y$. This too is done using Lemma 5.3.
   (b) Sender samples a uniformly random blinding vector $e = (e_1, \ldots, e_n) \in \mathbb{F}^n$ subject to the strict additive constraint $\sum_{i=1}^{n} e_i = z$.
   (c) Sender sends shares d_i and e_i to Server i (for $i = 1, \ldots, n$).
4. **Server computation:**
   (a) Server i receives c_i from Receiver and (d_i, e_i) from Sender.
   (b) Server i locally computes $w_i = e_i - c_i \cdot d_i$.
   (c) Server i sends w_i back to Receiver.
5. **Output decoding:**
   (a) Receiver computes the final output as $\text{out} = \sum_{i=1}^{n} w_i$.

See Appendix A.2 for analysis of Protocol 1.

From OLE to general constant-size functions. To extend the above protocol from two-party OLE to a protocol with $m = O(1)$ clients for any constant-size function f, we can use the known completeness of OLE (an arithmetic variant of OT) for secure computation of Boolean or arithmetic circuits [GMW87, IPS09]. At a high level, the clients start by additively sharing their inputs among each other. Then, each addition gate is processed by having the clients locally add their shares, and each multiplication gate is processed via $O(m^2)$ invocations of the OLE protocol to additively share each cross-term $a_i b_j$ between the client holding a_i and the client holding b_j. The protocol concludes by having the clients exchange their output shares. When the field $\mathbb{F}$ is of constant size, the Boolean circuit complexity of the protocol is $O(n)$ for any constant-size circuit.

Achieving near-optimal security. The security threshold t of the protocol is close to the smaller minimal distance of the primal code and its dual, namely $\min(\mathsf{dist}, \mathsf{dist}^{\perp})$. One approach for getting it arbitrarily close to $1/2$ is to let $\mathbb{F}$ be an arbitrarily large constant, in which case Theorem 3.1 gives a pair of codes with relative distance $1/2 - \varepsilon$. Another approach is to start with a protocol that has an arbitrarily small (but constant) security threshold, say by using $\mathbb{F} = \mathbb{F}_2$, and apply a general threshold amplification technique based on virtualization [Bra87, ORV94, DIK$^+$08].

Putting everything together. With the above extensions of Proposition 5.2, we can use our constructions of fast dual codes (cf. Theorem 3.1) to obtain the main result of this section captured by Theorem 5.1. This construction is semi-explicit in the sense that there is a PPT algorithm that generates circuits implementing the parties with $negl(n)$ failure probability.

5.2 Fast Encrypted Matrix-Vector Product

Switching to the computational security setting, a recent application of fast dual codes with random-like properties was recently given in the context of computing on encrypted data [BCH$^+$25]. Concretely, an *encrypted matrix-vector product* (EMVP) protocol enables a client to encode a matrix M using a secret code $\mathcal{C}$ and then compute an unbounded number of matrix-vector products Mq_i by encrypting each query vector q_i using the dual code $\mathcal{C}^{\perp}$. Linear algebra attacks are prevented by adding a suitable type of noise. Under a variant of the Learning Subspace with Noise (LSN) assumption [DKL09, CIMR25], the protocol hides both M and q_i from a server who stores the encrypted matrix and processes the encrypted queries.

Using our new families of fast dual codes, we obtain the first candidate EMVP protocols in which both the matrix encoding and the query encoding have asymptotically optimal circuit complexity. The recent construction of fast dual codes from [BR26] is limited to $\mathbb{F}_2$ and has a non-negligible failure probability.

5.3 Fast and Conservative Substitutes for Random Linear Codes

The previous two applications inherently relied on having a fast code coupled with a fast dual. A broader use case, which does not require duality, is providing a fast substitute to a random linear code in the context of applications that rely on the hardness of "noisy linear algebra." Our new construction of fast t-LUOF (Section 4.2) can serve as a more conservative candidate than the previous 1-LUOF construction from [DI14].

In more detail, in most cryptographic applications of noisy linear algebra, one typically considers the minimal distance of the code or its dual as a heuristic measure of hardness. Thus, it is perhaps

not clear why the t-local similarity provided by t-LUOF (for $t \geq 2$) is interesting. To illustrate this, consider code-based constructions that rely on the LPN assumption,[5] where one assumes that the distributions

$$(A, Ae) \quad \text{and} \quad (A, b)$$

are computationally indistinguishable, where e is a random sparse vector and b is a uniformly random vector (in this case, one should think of $A \in \mathbb{F}_q^{k \times r}$ with $r > k$). This use of minimal distance as a proxy for conjectured LPN hardness is common in the recent line of works on pseudorandom correlation generators (PCGs) [BCGI18, BCG+19]. Indeed, minimal distance is sufficient for provably defeating any attack that fits under a natural "linear attack" framework [CRR21].

However, it is arguably a more conservative to use an A that shares more nuanced combinatorial properties with random linear codes, such as list-decodability/-recoverability. This is exactly what the t-LUOF property guarantees.

5.4 MPC-Friendly t-Wise Hash Functions

Finally, when distributing the computation of a function f between two or more parties, either via classical MPC techniques [Yao86, GMW87, BGW88, CCD88] or via fully homomorphic encryption [Gen09], the cost of the protocol depends on both the circuit size and the algebraic degree. For $t = O(1)$, our constructions of fast t-wise independent hash functions from Section 4.1 *simultaneously* optimize both asymptotic size and degree.

There are many algorithmic applications that benefit from t-wise independent hashing for constant $t > 2$. Two well-known examples are the use of 4-wise independence in sketching [AMS99] and 5-wise independence in linear probing [PPR07].

References

[AMS99] Noga Alon, Yossi Matias, and Mario Szegedy. The space complexity of approximating the frequency moments. *Journal of Computer and System Sciences*, 58(1):137–147, 1999.

[BC12] Christina Boura and Anne Canteaut. On the influence of the algebraic degree of F^{-1} on the algebraic degree of $G \circ F$. *IEEE Transactions on Information Theory*, 59(1):691–702, 2012.

[BCG+19] Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, Lisa Kohl, and Peter Scholl. Efficient pseudorandom correlation generators: Silent OT extension and more. In *Annual International Cryptology Conference (CRYPTO)*, pages 489–518. Springer, 2019.

[BCG+23] Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, Lisa Kohl, Nicolas Resch, and Peter Scholl. Oblivious transfer with constant computational overhead. In Carmit Hazay and Martijn Stam, editors, *Advances in Cryptology - EUROCRYPT 2023 - 42nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Lyon, France, April 23-27, 2023, Proceedings, Part I*, volume 14004 of *Lecture Notes in Computer Science*, pages 271–302. Springer, 2023.

[5] What we will give below is often called *dual*-LPN, which is equivalent to the more standard variant.

[BCGI18] Elette Boyle, Geoffroy Couteau, Niv Gilboa, and Yuval Ishai. Compressing vector OLE. In David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang, editors, *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS 2018, Toronto, ON, Canada, October 15-19, 2018*, pages 896–912. ACM, 2018.

[BCH+25] Fabrice Benhamouda, Caicai Chen, Shai Halevi, Yuval Ishai, Hugo Krawczyk, Tamer Mour, Tal Rabin, and Alon Rosen. Encrypted matrix-vector products from secret dual codes. In Chun-Ying Huang, Jyh-Cheng Chen, Shiuh-Pyng Shieh, David Lie, and Véronique Cortier, editors, *Proceedings of the 2025 ACM SIGSAC Conference on Computer and Communications Security, CCS 2025, Taipei, Taiwan, October 13-17, 2025*, pages 394–408. ACM, 2025.

[BGW88] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation. In *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing (STOC)*, pages 1–10. ACM, 1988.

[BIR26] Martijn Brehm, Yuval Ishai, and Nicolas Resch. Fast bounded-independence functions and their duals. In *Information Theoretic Cryptography Conference (ITC '26)*, 2026.

[Bor56] J. L. Bordewijk. Inter-reciprocity applied to electrical networks. *Applied Scientific Research, Section B*, 6(1):1–74, 1956.

[BR26] Martijn Brehm and Nicolas Resch. Linear Time Encodable Binary Code Achieving GV Bound with Linear Time Encodable Dual Achieving GV Bound. In Shubhangi Saraf, editor, *17th Innovations in Theoretical Computer Science Conference (ITCS 2026)*, volume 362 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 28:1–28:19, Dagstuhl, Germany, 2026. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

[Bra87] Gabriel Bracha. An $O(\log n)$ expected rounds randomized Byzantine generals protocol. *Journal of the ACM (JACM)*, 34(4):910–920, 1987.

[CCD88] David Chaum, Claude Crépeau, and Ivan Damgård. Multiparty unconditionally secure protocols. In *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing (STOC)*, pages 11–19. ACM, 1988.

[CDM00] Ronald Cramer, Ivan Damgård, and Ueli M. Maurer. General secure multi-party computation from any linear secret-sharing scheme. In Bart Preneel, editor, *Advances in Cryptology - EUROCRYPT 2000, International Conference on the Theory and Application of Cryptographic Techniques, Bruges, Belgium, May 14-18, 2000, Proceeding*, volume 1807 of *Lecture Notes in Computer Science*, pages 316–334. Springer, 2000.

[CGH+85] Benny Chor, Oded Goldreich, Johan Håstad, Joel Friedman, Steven Rudich, and Roman Smolensky. The bit extraction problem of t-resilient functions (preliminary version). In *26th Annual Symposium on Foundations of Computer Science, Portland, Oregon, USA, 21-23 October 1985*, pages 396–407. IEEE Computer Society, 1985.

[CIMR25] Caicai Chen, Yuval Ishai, Tamer Mour, and Alon Rosen. Secret-key PIR from random linear codes. *Cryptology ePrint Archive*, 2025. To appear in STOC 2026.

[CRR21] Geoffroy Couteau, Peter Rindal, and Srinivasan Raghuraman. Silver: Silent VOLE and oblivious transfer from hardness of decoding structured LDPC codes. In Tal Malkin and Chris Peikert, editors, *Advances in Cryptology - CRYPTO 2021 - 41st Annual International Cryptology Conference, CRYPTO 2021, Virtual Event, August 16-20, 2021, Proceedings, Part III*, volume 12827 of *Lecture Notes in Computer Science*, pages 502–534. Springer, 2021.

[CW79] Larry Carter and Mark N. Wegman. Universal classes of hash functions. *Journal of Computer and System Sciences*, 18(2):143–154, 1979.

[DI14] Erez Druk and Yuval Ishai. Linear-time encodable codes meeting the gilbert-varshamov bound and their cryptographic applications. In *Proceedings of the 5th Conference on Innovations in Theoretical Computer Science*, ITCS '14, page 169182, New York, NY, USA, 2014. Association for Computing Machinery.

[DIK+08] Ivan Damgård, Yuval Ishai, Mikkel Krøigaard, Jesper Buus Nielsen, and Adam Smith. Scalable multiparty computation with nearly optimal work and resilience. In *Advances in Cryptology – CRYPTO 2008*, volume 5157 of *Lecture Notes in Computer Science*, pages 241–261. Springer, 2008.

[DJM98] Dariush Divsalar, Hui Jin, and Robert J McEliece. Coding theorems for" turbo-like" codes. In *Proceedings of the annual Allerton Conference on Communication control and Computing*, volume 36, pages 201–210. University Of Illinois, 1998.

[DKL09] Yevgeniy Dodis, Yael Tauman Kalai, and Shachar Lovett. On cryptography with auxiliary input. In Michael Mitzenmacher, editor, *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, May 31 - June 2, 2009*, pages 621–630. ACM, 2009.

[FLY22] Zhiyuan Fan, Jiatu Li, and Tianqi Yang. The exact complexity of pseudorandom functions and the black-box natural proof barrier for bootstrapping results in computational complexity. In Stefano Leonardi and Anupam Gupta, editors, *STOC '22: 54th Annual ACM SIGACT Symposium on Theory of Computing, Rome, Italy, June 20 - 24, 2022*, pages 962–975. ACM, 2022.

[Gen09] Craig Gentry. Fully homomorphic encryption using ideal lattices. In Michael Mitzenmacher, editor, *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, May 31 - June 2, 2009*, pages 169–178. ACM, 2009.

[GHK11] Venkatesan Guruswami, Johan Håstad, and Swastik Kopparty. On the list-decodability of random linear codes. *IEEE Transactions on Information Theory*, 57(2):718–725, 2011.

[GHK+13] Anna Gál, Kristoffer Arnsfelt Hansen, Michal Koucký, Pavel Pudlák, and Emanuele Viola. Tight bounds on computing error-correcting codes by bounded-depth circuits with arbitrary gates. *IEEE Trans. Inf. Theory*, 59(10):6611–6627, 2013.

[GM22] Venkatesan Guruswami and Jonathan Mosheiff. Punctured low-bias codes behave like random linear codes. In *63rd IEEE Annual Symposium on Foundations of Computer*

Science, FOCS 2022, Denver, CO, USA, October 31 - November 3, 2022, pages 36–45. IEEE, 2022.

[GMW87] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or a completeness theorem for protocols with honest majority. In *Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing (STOC)*, pages 218–229. ACM, 1987.

[HHL19] Hamed Hatami, Pooya Hatami, and Shachar Lovett. Higher-order fourier analysis and applications. *Foundations and Trends® in Theoretical Computer Science*, 13(4):247–448, 2019.

[HIKN08] Danny Harnik, Yuval Ishai, Eyal Kushilevitz, and Jesper Buus Nielsen. Ot-combiners via secure computation. In Ran Canetti, editor, *Theory of Cryptography, Fifth Theory of Cryptography Conference, TCC 2008, New York, USA, March 19-21, 2008*, volume 4948 of *Lecture Notes in Computer Science*, pages 393–411. Springer, 2008.

[IKOS08] Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Cryptography with constant computational overhead. In Cynthia Dwork, editor, *Proceedings of the 40th Annual ACM Symposium on Theory of Computing, Victoria, British Columbia, Canada, May 17-20, 2008*, pages 433–442. ACM, 2008.

[IPS09] Yuval Ishai, Manoj Prabhakaran, and Amit Sahai. Secure arithmetic computation with no honest majority. In Omer Reingold, editor, *Theory of Cryptography, 6th Theory of Cryptography Conference, TCC 2009, San Francisco, CA, USA, March 15-17, 2009. Proceedings*, volume 5444 of *Lecture Notes in Computer Science*, pages 294–314. Springer, 2009.

[Li26] Yuan Li. Secret sharing on superconcentrator, 2026.

[LW20] Ray Li and Mary Wootters. Improved list-decodability of random linear binary codes. *IEEE Transactions on Information Theory*, 67(3):1522–1536, 2020.

[Mas95] James L Massey. Some applications of coding theory in cryptography. In *Codes and Cyphers: Cryptography and Coding IV*, pages 33–47. Forma, 1995.

[MRR+24] Jonathan Mosheiff, Nicolas Resch, Noga Ron-Zewi, Shashwat Silas, and Mary Wootters. Low-density parity-check codes achieve list-decoding capacity. *SIAM J. Comput.*, 53(6):S20–38, 2024.

[MRSY25] Jonathan Mosheiff, Nicolas Resch, Kuo Shang, and Chen Yuan. Randomness-efficient constructions of capacity-achieving list-decodable codes. In *2025 IEEE International Symposium on Information Theory (ISIT)*, pages 1–6, 2025.

[ORV94] Rafail Ostrovsky, Sridhar Rajagopalan, and Umesh V. Vazirani. Simple and efficient leader election in the full information model. In Frank Thomson Leighton and Michael T. Goodrich, editors, *Proceedings of the Twenty-Sixth Annual ACM Symposium on Theory of Computing, 23-25 May 1994, Montréal, Québec, Canada*, pages 234–242. ACM, 1994.

[PPR07] Anna Pagh, Rasmus Pagh, and Milan Ružić. Linear probing with constant independence. In *Proceedings of the thirty-ninth annual ACM symposium on Theory of computing*, pages 318–327, 2007.

[Rao07] Anup Rao. An exposition of bourgain's 2-source extractor. *Electron. Colloquium Comput. Complex.*, TR07-034, 2007.

[Spi96] Daniel A. Spielman. Linear-time encodable and decodable error-correcting codes. *IEEE Transactions on Information Theory*, 42(6):1723–1731, 1996.

[Vad12] Salil P. Vadhan. Pseudorandomness. *Foundations and Trends in Theoretical Computer Science*, 7(1–3):1–336, 2012.

[Yao86] Andrew Chi-Chih Yao. How to generate and exchange secrets. In *27th Annual Symposium on Foundations of Computer Science (FOCS 1986)*, pages 162–167. IEEE, 1986.

A Deferred Proofs

A.1 Proof of Lemma 5.3

Proof. We prove each property separately.

Perfect privacy. Suppose for the sake of contradiction that a set of shares T with $|T| \leq \mathsf{dist}^{\perp} - 2$ leaks information about the secret σ. In a linear secret sharing scheme, this implies that the secret can be reconstructed via a linear combination of the shares in T. Thus, there exist constants $\{a_i\}_{i \in T}$ over $\mathbb{F}$ such that for all valid codewords $c \in \mathcal{C}$:

$$c_0 = \sum_{i \in T} a_i c_i$$

Rearranging, we get the following linear constraint:

$$c_0 - \sum_{i \in T} a_i c_i = 0$$

By definition, any such a constraint corresponds to a vector in the dual code $\mathcal{C}^{\perp}$. Specifically, it defines a dual codeword $h \in \mathcal{C}^{\perp}$ where $h_0 = 1$, $h_i = -a_i$ for $i \in T$, and $h_j = 0$ for all $j \notin T \cup \{0\}$.

Because $h_0 \neq 0$, h is a non-zero dual codeword. Its Hamming weight is at most $1 + |T|$. Given our assumption that $|T| \leq \mathsf{dist}^{\perp} - 2$, the weight of h is bounded by:

$$wt(h) \leq 1 + |T| \leq 1 + (\mathsf{dist}^{\perp} - 2) = \mathsf{dist}^{\perp} - 1$$

This directly contradicts the premise that the dual distance of $\mathcal{C}$ is $\mathsf{dist}^{\perp}$, which requires all non-zero dual codewords to have a weight of at least $\mathsf{dist}^{\perp}$. Therefore, no such linear combination exists, and the shares in T are perfectly independent of σ.

Circuit size. Using column vector notation, encoder circuit E implements a linear map $c = Mm$, where $m \in \mathbb{F}^k$ is a message vector and $M \in \mathbb{F}^{(n+1)\times k}$ is the generator matrix of $\mathcal{C}$. The first coordinate of the resulting codeword is:

$$c_0 = \sum_{j=1}^{k} M_{0,j} m_j$$

Because $\mathsf{dist}^\perp \geq 2$, the vector $(1, 0, \ldots, 0)$ is strictly excluded from $\mathcal{C}^\perp$. Consequently, c_0 is not identically zero for all codewords, meaning there is at least one index j^* where the matrix entry $M_{0,j^*} \neq 0$. Without loss of generality, assume $M_{0,k} \neq 0$.

To enforce the condition $c_0 = \sigma$ and uniformly sample the remaining degrees of freedom, the arithmetic sharing circuit performs the following steps:

1. Takes σ and $k-1$ uniformly random field elements $m_1, \ldots, m_{k-1} \in \mathbb{F}$ as input.
2. Computes the dependent message symbol m_k algebraically:

$$m_k = M_{0,k}^{-1} \left(\sigma - \sum_{j=1}^{k-1} M_{0,j} m_j \right)$$

   Because the coefficients $M_{0,j}$ are fixed constants, computing m_k requires at most $k-1$ field additions/subtractions and 1 scalar multiplication. This step adds exactly k arithmetic gates.
3. Feeds $m = (m_1, \ldots, m_k)$ into the original encoder circuit E, which requires s gates.

The total size of the new sharing circuit is $s + k$. Since $\mathcal{C}$ is a proper subspace of $\mathbb{F}^{n+1}$ (implied by the dual distance strictly greater than 1), its dimension k is bounded by $k \leq n$. Therefore, the total arithmetic circuit size is strictly bounded by $s + n$. □

A.2 Analysis of Protocol 1

We separately analyze correctness, circuit complexity, and security.

Perfect correctness. Receiver computes $\text{out} = \sum_{i=1}^{n} w_i = \sum_{i=1}^{n} (e_i - c_i d_i)$. By linearity, this separates into:

$$\text{out} = \sum_{i=1}^{n} e_i - \sum_{i=1}^{n} c_i d_i$$

Because $c \in \mathcal{C}$ and $d \in \mathcal{C}^\perp$, their inner product over all $n+1$ coordinates is exactly zero: $\sum_{i=0}^{n} c_i d_i = 0$. This implies $\sum_{i=1}^{n} c_i d_i = -c_0 d_0 = -xy$. Furthermore, Sender constructed the blinding vector such that $\sum_{i=1}^{n} e_i = z$. Substituting these values yields:

$$\text{out} = z - (-xy) = xy + z$$

as required.

Circuit complexity. Using Lemma 5.3, sampling $c \in \mathcal{C}$ given $c_0 = x$ using the size-s encoder E requires an arithmetic circuit of size at most $s + n$. Symmetrically, sampling $d \in \mathcal{C}^\perp$ requires size at most $s + n$. Sampling the additive shares e requires generating $n - 1$ random elements and $n - 1$ subtractions, which takes $O(n)$ gates. Each of the n servers performs exactly one multiplication and one subtraction ($O(1)$ per server, $O(n)$ total). The Receiver performs $n - 1$ additions to reconstruct the output ($O(n)$ gates). Summing these costs yields a total arithmetic circuit size of $O(n + s)$.

Perfect security. We show that the adversary's view can be perfectly simulated. Let $T \subset \{1, \ldots, n\}$ be the set of corrupted servers, with $|T| \leq t$. If only the servers are corrupted, security follows immediately from Lemma 5.3. We consider the cases where one client is corrupted, and assume wlog that $|T| = t$.

Case 1: Sender and t servers are corrupted. The real view consists of y, z, the codeword d, the blinding vector e, and the shares $\{c_i\}_{i \in T}$. The privacy of the Receiver's input depends on the dual distance of $\mathcal{C}$, which is $\mathsf{dist}^\perp$. Because $\mathsf{dist}^\perp > t + 1$, we have $|T| \leq t \leq d^\perp - 2$. By Lemma 5.3, the projection of c onto the coordinates perfectly hides x.

Case 2: Receiver and t servers are corrupted. The real view consists of x, the codeword c, the output $xy + z$, the shares $\{d_i, e_i\}_{i \in T}$ from Sender, and the responses $\{w_i\}_{i=1}^n$. The privacy of Sender's input y depends on the dual distance of $\mathcal{C}^\perp$, which is the minimum distance of the primal code $\mathcal{C}$, denoted as dist.

1. **Simulate Sender's shares:** Because $\mathsf{dist} > t + 1$, we have $|T| \leq t \leq \mathsf{dist} - 2$. The true projection of d onto T is uniformly distributed over $\mathbb{F}^{|T|}$. The simulator samples $\{d_i\}_{i \in T}$ uniformly at random. It also samples $\{e_i\}_{i \in T}$ uniformly at random.
2. **Simulate corrupted server responses:** For $i \in T$, the simulator computes $w_i = e_i - c_i d_i$.
3. **Simulate honest server responses:** The final sum must evaluate to the legitimate output $xy + z$. The sum of the uncorrupted responses is deterministically constrained: $\sum_{i \notin T} w_i = (xy + z) - \sum_{i \in T} w_i$. The simulator samples the remaining $n - |T|$ values of w_i uniformly at random subject to this sum.

In the real protocol, the uncorrupted blinding shares $\{e_i\}_{i \notin T}$ are uniformly random field elements subject only to the constraint $\sum_{i \notin T} e_i = z - \sum_{i \in T} e_i$. Because $w_i = e_i - c_i d_i$, the $\{e_i\}_{i \notin T}$ values act as an information-theoretic one-time pad perfectly masking the values of $c_i d_i$. The constraint on the visible responses is:

$$\sum_{i \notin T} w_i = \sum_{i \notin T} (e_i - c_i d_i) = \left(z - \sum_{i \in T} e_i \right) - \left(-xy - \sum_{i \in T} c_i d_i \right) = (xy + z) - \sum_{i \in T} (e_i - c_i d_i) = (xy + z) - \sum_{i \in T} w_i$$

The real variables $\{w_i\}_{i \notin T}$ are uniformly distributed subject exactly to this equation, perfectly matching the simulator. The adversary extracts no additional information about y or z.