BOOLEAN FUNCTIONS FROM HYPERPLANE COVERINGS

a new representation and applications

Benjamin E. Diamond [*]

J.P. Morgan AI Research

benjamin.e.diamond@jpmchase.com

Abstract

We give a new mathematical representation of boolean functions over prime fields. We show that any subset of the discrete unit cube can be exactly covered by affine hyperplanes (over a sufficiently large prime field). Any boolean function whose "on-set" has been covered in this way can be evaluated within arithmetic protocols. We study the number of hyperplanes required to cover concrete boolean functions, and provide various interesting and surprising results. Our results yield randomized polynomial encodings in the sense of Ishai and Kushilevitz (FOCS '00) which are smaller than prior known constructions. For example, we obtain depth-three, quasilinearly-sized arithmetic circuits which randomly encode the *majority* and *parity* functions. We give practical and implemented applications in secure computation. We finally pose the problem of *constructing* compact hyperplane coverings of arbitrary prescribed cube subsets. We present a new computational-algebraic algorithm for this problem, applying a new technique for the evaluation of affine spans over prime fields.

1 Introduction

It is common in cryptography to evaluate arithmetic circuits over finite fields. Shamir's classic secret-sharing scheme, for example, "splits" field elements into parts, upon which arithmetic operations may be jointly evaluated by mutually distrustful parties. This is essentially the strategy taken by the classic protocol of Ben-Or, Goldwasser, and Wigderson [BOGW88] (see also the subsequent works of Gennaro, Rabin, and Rabin [GRR98] and Asharov and Lindell [AL17]). Many of the techniques surveyed Cramer, Damgård, and Nielsen [CDN15] assume this rough structure.

Many "important" functions, however—like arithmetic comparison—are most naturally expressed as boolean functions, which in particular have boolean inputs. It's thus typical to "feed" initially *boolean* values into arithmetic protocols (using the natural inclusion $\{0,1\} \subset \mathbb{F}_p$) and to evaluate arithmetic operations on these. (Alternatively, one may begin with arithmetic inputs, but then "bit-decompose" them, in the sense of Damgård et al. [DFK$^+$06, §1]).

It's thus important to represent initially boolean functions $f : \{0,1\} \rightarrow \{0,1\}$ as arithmetic circuits. Moreover, the *depth* of the arithmetic circuit directly controls the *round complexity* of the resulting protocol (see e.g. [AL17, Thm. 1]). Cramer, Damgård and Nielsen [CDN15, Ex. 3.1] note that any boolean function $f : \{0,1\}^n \rightarrow \{0,1\}$ can be trivially represented by a polynomial whose degree grows polynomially in f's boolean circuit depth. The construction of *constant-depth* representations was initiated by Ishai and Kushilevitz [IK00] [IK02], who observe that, in practice, it's enough to "randomly encode" an initially boolean function into an *array* of multivariate *random* polynomials, which maps boolean inputs to output *distributions.* Ishai and Kushilevitz [IK00] [IK02] give degree-3 representations of arbitrary functions f (and argue that degree-2 representations are generally impossible [IK00, Cor. 5.9]). The *sizes* of the resulting polynomials are measured in various ways. Those of [IK00] grow quadratically in the size of a nondeterministic mod-p branching program evaluating f. Those of [IK02, §3.1] grow as $O(s^{\log_2 3})$ in the size s of a boolean formula

[*]With an appendix by Benjamin Diamond and Jason Long.

evaluating f, though the authors report that—at least for certain functions—a further improvement to $O(s \cdot 2^{\sqrt{\log s}})$ is possible.

In more recent work, Applebaum, Brakerski and Tsabary [ABT18] achieve randomized polynomial encodings of degree-2, which achieves the theoretical optimum. They circumvent [IK00, Cor. 5.9] by relaxing the definition [IK00, §2.1]; the authors introduce a generalization of randomizing polynomials called "multi-party randomized encodings", in which—roughly—each party is allowed to arbitrarily preprocess its randomness (see for example the discussion [ABT18, §1.1]). As [ABT18] shows, this relaxation allows the degree of the construction [IK02, §3.1] to be reduced by 1. The construction of [ABT18] is rather abstract, and is not easily adapted for practical use. Indeed, it requires that an entire multi-player *protocol* be encoded into a boolean circuit, that this circuit be garbled, and that this garbled circuit be represented as a polynomial (roughly as in the construction [IK02, §3.1]). The authors report only polynomial efficiency, and do not explicitly analyze their protocol's performance.

1.1 Our contribution

We present a new, efficient, mathematically natural and remarkably simple arithmetic representation of boolean functions, starting from first principles. Our approach departs radically from prior techniques, and relies on affine-linear algebra over prime fields. (Arbitrary finite fields of odd characteristic work too, but we specialize our notation for convenience). It yields asymptotically smaller representations than do prior approaches. Moreover, it's conceptually very simple, and more favorable for practical implementation and use.

Definition 1.1. We say that a fixed subset $S \subset \{0,1\}^n$ of the discrete unit cube is *disjointly covered* by affine hyperplanes $\{H_i\}_{i=0}^{m-1}$ in $\mathbb{F}_p^n$ if $S = \bigsqcup_{i=0}^{m-1} H_i \cap \{0,1\}^n$.

The hyperplanes $\{H_i\}_{i=0}^{m-1}$, viewed as subsets of the vector space $\mathbb{F}_p^n$, thus respectively intersect the discrete unit cube in such a way that their restrictions disjointly cover the desired subset S, and cover no further elements of $\{0,1\}^n$.

We may as well take as S the "on-set" of some boolean function $f : \{0,1\}^n \to \{0,1\}$, defined as $f^{-1}(1) \subset \{0,1\}^n$, the set of points on which f evaluates to 1.

Theorem 1.2. *A covering $\{H_i\}_{i=0}^{m-1}$ of f's* on-set *yields a random encoding of f.*

Indeed, we can just evaluate the hyperplanes $\{H_i\}_{i=0}^{m-1}$ on any boolean input $\mathbf{x} \in \{0,1\}^n$, which we view now as an element of $\mathbb{F}_p^n$ (we recall that hyperplanes can equally be viewed as affine-linear functionals $\mathbb{F}_p^n \to \mathbb{F}_p$). Our hypothesis on $\{H_i\}_{i=0}^{m-1}$ entails exactly that $f(\mathbf{x}) = 1$ if and only if $H_{i^*}(\mathbf{x}) = 0$ for some (unique) index $i^* \in \{0, \ldots, m-1\}$. Thus, the presence of a 0 in the set of evaluations serves as a "fingerprint" of the fact that $f(\mathbf{x}) = 1$.

We're not done yet, because the output vector $\{H_i(\mathbf{x})\}_{i=0}^{m-1}$ reveals more than just $f(\mathbf{x})$ alone. On the other hand, it's not difficult to appropriately randomize these outputs, using an idea already implicit in [WGC19, Alg. 3]. We may multiply each evaluation $H_i(\mathbf{x})$ by a nonzero random scalar $\alpha_i \leftarrow \mathbb{F}_p^*$, and finally randomly *permute* the resulting values, to obscure the index $i^* \in \{0, \ldots, m-1\}$ if it exists (actually, it's enough merely to circularly shift these outputs by a random offset). If $f(\mathbf{x}) = 1$, then the resulting distribution is supported on the union of the coordinate hyperplanes; otherwise, it's uniform on their complement.

We write $P(\mathbf{x}) := \left(\alpha_i \cdot H_{\rho(i)}(\mathbf{x})\right)_{i=0}^{m-1}$ for the resulting construction. The "complexity" of $P(\mathbf{x})$ is interesting to study, and depends on the number of hyperplanes m required to cover $f^{-1}(1)$. At the very least, we have the following "completeness theorem", at least if p is not too small (see Theorem 3.13):

Theorem 1.3. *If $p > n$, then any $S \subset \{0,1\}^n$ has a covering by $\mathbb{F}_p$-hyperplanes.*

Yet the covering could be exponentially large, *a priori.* On the other hand, many natural functions have small—and non-obvious—hyperplane coverings:

Theorem 1.4. *For n even (say), any prime p of n bits, and $f : \{0,1\}^n \to \{0,1\}$*

- *the arithmetic comparison function on $\frac{n}{2}$-bit unsigned integers,*

- *the parity or majority function on n bits (or in fact any symmetric function),*
- *the OR function, or*
- *the Hamming weight comparator function on length-$\frac{n}{2}$ bitstrings,*

then both $f^{-1}(0)$ and $f^{-1}(1)$ can be covered using $m(n) \in O(n)$ $\mathbb{F}_p$-hyperplanes. These hyperplanes can be expressed as an arithmetic circuit of $O(n)$ total size.

Thus, there are amply many functions for which $P(\mathbf{x})$ can be evaluated in $O(n)$ time on any input $\mathbf{x} \in \{0,1\}^n$ (see also Examples 3.22, 3.25, and 3.33 below). In fact, e initiate the theoretical study of "hyperplane complexity". We introduce the boolean complexity class consisting of functions whose off-sets and on-sets can be covered by only polynomially many hyperplanes (see Definition 3.3 below):

Definition 1.5. A sequence of functions $\{f_n : \{0,1\}^n \to \{0,1\}\}_{n\in\mathbb{N}}$ is *efficiently computable by affine hyperplanes* if, for each $n \in \mathbb{N}$, there exists an n-bit prime p and disjoint coverings $f^{-1}(0) = \bigsqcup_{i=0}^{m-1} H_{0,i} \cap \{0,1\}^n$ and $f^{-1}(1) = \bigsqcup_{i=0}^{m-1} H_{1,i} \cap \{0,1\}^n$ using $\mathbb{F}_p$-hyperplanes, where m moreover grows polynomially in n.

Though we are not able to concretely exhibit a function which is not efficiently computable by hyperplanes, we prove non-constructively—in the spirit of Shannon—that "most" functions require exponentially many hyperplanes to compute (see Theorem 3.4 below).

$P(\mathbf{x})$ isn't strictly a polynomial operation, because its construction involves a permutation. In many plausible applications, this is not a problem:

Example 1.6. $P(\mathbf{x})$ can be used to create a simple two-party computation scheme. The first party El Gamal-encrypts (let's say) its inputs and sends the ciphertexts to the second. The second party evaluates $P(\mathbf{x})$ homomorphically on the first party's encrypted inputs and on its own plaintexts. It re-randomizes and returns the resulting ciphertexts to the first party. Upon decrypting these and checking for a 0, the first party learns the output, which it may report to the second.

This construction is both correct and secure, because of Theorem 1.2; we describe it in detail in Subsection 5.1.

If we want to "plug" $P(\mathbf{x})$ into generic secure computation protocols, then we have to express it as a polynomial. Thankfully, permutations aren't hard to express as polynomial operations (see Lemma 3.39):

Theorem 1.7. *There is a random matrix $R \in \mathbb{F}_p^{m\times m}$ such that $P(\mathbf{x})$ and $R \cdot [H_i(\mathbf{x})]_{i=0}^{m-1}$ give identical distributions on $\mathbb{F}_p^m$ for each boolean input $\mathbf{x} \in \{0,1\}^n$.*

This shows that $P(\mathbf{x})$ is expressible as a *degree-2* polynomial (in its combined secret and random inputs). This already "contradicts" the impossibility result [IK00, Cor. 5.9] and matches the optimal degree attained by [ABT18]. We evade that impossibility result in an arguably simpler way: we simply allow the use of random inputs which are *not* necessarily independent and uniform. Indeed, the impossibility result [IK00, Cor. 5.9] relies essentially on the independence of all scalars sampled by $P(\mathbf{x})$; our R does not have independently random components, and in particular draws from a distribution on $\mathbb{F}_p^{m\times m}$ which is *not* a product of independent uniform distributions. We describe in Remark 3.40 explicitly where the proof of [IK00, Cor. 5.9] fails in our more general setting.

Moreover, this particular polynomial representation is of size $O(m^2)$. This too improves upon prior constructions, at least for certain functions; for example, parity and majority satisfy $m(n) \in O(n)$, and yet can be evaluated by boolean formulas only of size $\Omega(n^2)$ [Weg87, §8.8 Thm. 8.2] (we recall moreover that [IK02, §3] grows at least superlinearly in f's formula's size).

If we're willing to increase the degree to 3, then we attain a still-smaller representation (see Lemma 3.41):

Theorem 1.8. *$P(\mathbf{x})$ can be computed by an $O(m \cdot \log m)$-size arithmetic circuit of depth 3.*

The proof invokes the number-theoretic transform, and again requires nonuniform randomness. This yields the smallest known random polynomial encodings for a large class of functions $f : \{0,1\}^n \to \{0,1\}$ (including, for example, any f whose boolean formula size grows at least as fast as its hyperplane complexity). We give details in Table 1 below.

It appears that there is no general "compiler" from boolean circuits to relatedly-sized hyperplane coverings (though we show that certain particular circuit families admit efficient conversion procedures in Corollaries 3.11 and 3.18 below). When no explicit covering $S = \bigsqcup_{i=0}^{m-1} H_i \cap \{0,1\}^n$ by hyperplanes is known, it is interesting to construct one algorithmically. This is a variant of a classical problem in logic synthesis and hardware design, which seeks to generate compact "sum-of-products" representations (any such representation amounts to a covering of S by *subcubes*).

Theorem 1.9. *Given n, a subset $S \subset \{0,1\}^n$, and a prime $p \geq 2^n$, Algorithm 4 outputs a family of affine hyperplanes $\{H_i\}_{i=0}^{m-1}$ in $\mathbb{F}_p^n$ for which $S = \bigsqcup_{i=0}^{m-1} H_i \cap \{0,1\}^n$, where moreover m is (heuristically) minimal.*

Moreover, Algorithm 4 seeks to find such a configuration for which m is as small as possible.

Our basic idea is to begin with points, which are 0-dimensional affine flats (i.e., affine subspaces). We then try to "grow" each individual affine flat dimension-by-dimension as much as we can. We do this using a recursive backtracking strategy. At each recursive stage, we select a random candidate point not yet included in the flat, and then extend the flat along the new direction. If this latter span "leaks" out of S (or intersects another flat already constructed), we discard the candidate, and try another. In this way, a tree of "flags" is built; we retain those leaves which cover the most points.

The key subroutine of our algorithm involves evaluating the intersection of an affine flat—expressed concretely as an affine basis—with the unit cube. We develop a new routine for this problem, exploiting a clever lemma of Odlyzko [Odl81, Thm. 2]. The idea is that once the matrix whose rows span an affine flat has been Gauss–Jordan-reduced, its possible intersections with $\{0,1\}^n$ can be rapidly read off and tested. We further improve the efficiency of the algorithm using the Gray code. We give details in Subsection 4.1.

Upon its termination, our algorithm only returns *flats* covering S, not hyperplanes. While any affine flat can clearly be extended into a hyperplane, the hard part is to not "pick up" new cube points in the process (this would in general destroy the correctness of the configuration). We develop in Subsection 4.4 below an algorithm for this problem, which requires that $p \geq 2^n$. In Appendix A, we show that this latter requirement can be weakened.

We exhibit a full implementation of our hyperplane-synthesis algorithm in Subsection 4.5 below. Our implementation evaluates random subsets $S \subset \{0,1\}^8$ optimally or near-optimally in well under a second. It also handles higher-dimensional subsets. For example, our algorithm evaluated a random subset $S \subset \{0,1\}^{14}$, of cardinality 8,112, in under 6 minutes, and produced a covering using $m = 546$ hyperplanes. We believe that this tool may be of general interest for cryptographers seeking to evaluate boolean functions by affine-linear means.

We conclude the paper with an application in cryptography, which can be understood as illustrative. Our protocol is practical, and is fully implemented. It is the first use in real life of random encodings, as far as we are aware. Given an arbitrary function $f : \{0,1\}^n \to \{0,1\}$ and a hyperplane covering $\{H_i\}_{i=0}^{m-1}$ in the sense of Definition 1.1, our protocol allows two parties and an untrusted "server" to compute f with security under one static malicious corruption. If the server alone requires the output, then it takes only 2 rounds; if all parties want the output, it takes 3. The computational and communication complexity of the protocol depend on the hyperplane cardinality m, and are small. Our implementation is multi-threaded for all parties and side-channel resistant, and operates over a WAN. Specialized to the boolean comparator function with $m = 32$, it can run the full protocol in less than 100 milliseconds, when parallelism is incorporated (and under 600 when it's not). Our protocol has a similar structure to Wagh, Gupta, and Chandran [WGC19, Alg. 3], and can be viewed as a vast generalization of that protocol, in which arbitrary functions f can be computed, and malicious security is achieved. We give details in Section 5.

We expend a significant amount of effort studying whether the requirement in Theorem 1.9 that p be *more than* n bits can be weakened (cf. Definition 1.5, where $p < 2^n$). That is, we study whether each natural number n admits a prime $p < 2^n$ such that every proper affine flat $K \subset \mathbb{F}_p^n$ has an affine hyperplane $H \subset \mathbb{F}_p^n$ for which $K \cap \{0,1\}^n = H \cap \{0,1\}^n$ (this property holds trivially for primes $p \geq 2^n$, as we argue in Subsection 4.4). This question presents a profound challenge in discrete geometry; we defer it to the appendix, because of its technical complexity. In a series of results (see Corollaries A.4 and A.17 below), we establish an essentially affirmative answer:

Theorem 1.10. *For each large enough n, that a prime p satisfies $p \geq 2^n - \frac{n}{\log \log n} \cdot \sqrt{2^n}$ implies that each proper affine flat $K \subset \mathbb{F}_p^n$ admits an affine hyperplane $H \subset \mathbb{F}_p^n$ for which $K \cap \{0,1\}^n = H \cap \{0,1\}^n$.*

Primes which satisfy the hypothesis of this theorem *and* satisfy $p < 2^n$ are abundant in practice; their existence would be guaranteed unconditionally by certain conjectures related to the Riemann hypothesis (see Corollary A.18 below). It is thus highly likely that the n-bit requirement of Defintion 1.5 is not restrictive.

2 Definitions and Notation

We give an accelerated overview of terminology in this section, focusing on linear algebra and cryptography. By the "natural numbers" (represented by the symbol $\mathbb{N}$) we shall mean the *positive integers*.

2.1 Linear and affine algebra

We write p for an *odd* prime (unless otherwise specified), and $\mathbb{F}_p$ for the finite field of order p (see Cohn [Coh74, §6.3] for basics). Most—though not all—of our results hold over arbitrary fields of odd characteristic; we present our results only for prime fields (for simplicity). We work in the *arithmetic* complexity model, in which each field operation takes constant time (see for example von zur Gathen and Gerhard [vzGG13, §2]).

We refer to Cohn [Coh74, §4] for preliminaries on linear algebra and affine spaces; in particular, see [Coh74, §8]. We also refer to Meyer [Mey00] for basic computational methods in linear algebra. In particular, an *affine flat* $K \subset \mathbb{F}_p^n$ is vector susbspace (containing the origin) of $\mathbb{F}_p^n$, say U, together with an *origin point* $\mathbf{o} \in \mathbb{F}_p^n$. The *dimension* of K is the dimension of U. The flat K *contains* the set $\{\mathbf{o} + \mathbf{x} \mid \mathbf{x} \in U\}$. We say that K *contains the origin* if $\mathbf{o} = \mathbf{0} \in \mathbb{F}_p^n$. Affine flats are exactly the nullsets of affine-linear maps $K : \mathbb{F}_p^n \to \mathbb{F}_p^{n-k}$. We often freely identify these two representations (effectively, the two representations can be interchanged using Lemmas 2.2 and 2.3).

The *affine span* of a set of $k+1$ points $\mathbf{x}_0, \ldots, \mathbf{x}_k$ in $\mathbb{F}_p^n$ is the set of all combinations $\sum_{i=0}^k \alpha_i \cdot \mathbf{x}_i$ for which $\sum_{i=0}^k \alpha_i = 1$. Each such combination can equally be expressed as $\mathbf{x}_0 + \sum_{i=1}^k \alpha_i \cdot (\mathbf{x}_i - \mathbf{x}_0)$. The affine span of $\mathbf{x}_0, \ldots, \mathbf{x}_k$ is equal to the flat with origin point $\mathbf{o} := \mathbf{x}_0$ and $U := \langle \mathbf{x}_1 - \mathbf{x}_0, \ldots, \mathbf{x}_k - \mathbf{x}_0 \rangle$.

By a "hyperplane", we shall mean a flat whose dimension is one less than that of the space, and which in particular need *not* contain the origin. A "subspace" necessarily contains the origin. Each flat K we study satisfies $K \cap \{0,1\}^n \neq \varnothing$. For each positive n, the $n-1$-dimensional *projective space* $\mathbb{P}\mathbb{F}_p^{n-1}$ consists of the set of lines (through the origin) in $\mathbb{F}_p^n$.

The following basic result essentially allows us to replace affine-linear algebra with linear algebra:

Lemma 2.1. *For each $\mathbf{x} \in \{0,1\}^n$, there exists an invertible affine $\mathbb{F}_p$-linear map $o_{\mathbf{x}} : \mathbb{F}_p^n \to \mathbb{F}_p^n$ which maps $\{0,1\}^n$ to itself, and which sends $\mathbf{x}$ to the origin.*

Proof. We write the coordinates of $\mathbf{x}$ as $(x_0, \ldots, x_{n-1})$. The map $o_{\mathbf{x}}$ defined on $\mathbf{y} = (y_0, \ldots, y_{n-1}) \in \mathbb{F}_p^n$ by:

$$o_{\mathbf{x}}(\mathbf{y}) := \left(\begin{cases} 1 - y_i & \text{if } x_i = 1 \\ y_i & \text{if } x_i = 0 \end{cases} \right)_{i=0}^{n-1}$$

clearly satisfies the desired properties. □

We note that, on the unit cube itself, $o_{\mathbf{x}}$ restricts to the XOR-by-$\mathbf{x}$ map.

We take for granted the notion of reduced row-echelon form and the Gauss–Jordan elimination algorithm (see e.g. Meyer [Mey00, p. 48]). We record the following formalization of an elementary technique in linear algebra (see e.g. [Mey00, (4.2.9)]):

Lemma 2.2. *Fix a full-rank, $k \times n$ matrix U in reduced row-echelon form. Write $\{b_0, \ldots, b_{k-1}\}$ and $\{c_0, \ldots, c_{n-k-1}\}$ for U's "pivot" and "free" column indices, respectively, viewed as subsets of $\{0, \ldots, n-1\}$. Define an $(n-k) \times n$ matrix A by setting:*

$$A := \left[\begin{cases} -U_{c_i,h} & \text{if } j \text{ is a pivot, say } b_h, \\ i \stackrel{?}{=} l & \text{if } j \text{ is free, say } c_l. \end{cases} \right]_{i,j=0}^{n-k-1,n-1}$$

Then A's rows give a linearly independent spanning set of U's kernel.

By "dualizing" the above lemma, we obtain the following result:

Lemma 2.3. *Viewed as a linear map, $A : \mathbb{F}_p^n \to \mathbb{F}_p^{n-k}$ annihilates exactly U's row-space.*

Proof. This follows trivially from the implication $\mathbf{0}_{k\times(n-k)} = U \cdot A^T \implies \mathbf{0}_{(n-k)\times k} = A \cdot U^T$. □

It is intuitively obvious that a line which passes through the cube can intersect at most two of its points. Lemma 2.5 below generalizes this fact to k-dimensional subspaces. This result was first stated and proven in a paper of Odlyzko [Odl81, Thm. 2]; the proof we give here is essentially identical.

Lemma 2.4. *Suppose that a $k \times n$ matrix U is row-reduced over $\mathbb{F}_p$. Then an $\mathbb{F}_p$-linear combination $\mathbf{x} = \sum_{i=0}^{k-1} \alpha_i \cdot \mathbf{x}_i$ of U's rows can reside in $\{0,1\}^n$ only if $\alpha_i \in \{0,1\}$ for each $i \in \{0,\ldots,k-1\}$.*

Proof. Writing $\{b_0,\ldots,b_{k-1}\} \subset \{0,\ldots,n-1\}$ for U's pivot column indices, the definition of row-reduction implies that each row $\mathbf{x}_i$, for $i \in \{0,\ldots,k-1\}$, has a component—namely, b_i—for which $U_{i,b_i} = 1$; moreover, $U_{i,b_i} = 1$ is the sole nonzero element in its column. The condition $\mathbf{x} \in \{0,1\}^n$ implies in particular that $\mathbf{x}$'s ${b_i}^{\text{th}}$ component is in $\{0,1\}$, and hence that $\alpha_i \in \{0,1\}$. □

Lemma 2.5. *A k-dimensional affine flat $K \subset \mathbb{F}_p^n$ can intersect $\{0,1\}^n$ in at most 2^k points.*

Proof. After assuming that K contains the origin (by Lemma 2.1), picking an independent spanning set of K, and row-reducing the resulting matrix, we may assume that K consists exactly of the $\mathbb{F}_p$-linear combinations of a row-reduced matrix U's rows. The result immediately follows from Lemma 2.4. □

2.2 Basic security definitions

We give basic security definitions, following Katz and Lindell [KL21]. In experiment-based games involving an adversary $\mathcal{A}$, we occasionally use the notation $\mathsf{out}_{\mathcal{A}}\left(\mathsf{E}_{\mathcal{A}}(\lambda)\right)$ to denote the *output* of $\mathcal{A}$ in the game $\mathsf{E}_{\mathcal{A}}(\lambda)$ (as distinguished from whether $\mathcal{A}$ wins the experiment). We work throughout in the *random oracle* model (see [KL21, §6.5]). Two distribution ensembles $Y_0 = \{Y_0(a,\lambda)\}_{a\in\{0,1\}^*;\lambda\in\mathbb{N}}$ and $Y_1 = \{Y_1(a,\lambda)\}_{a\in\{0,1\}^*;\lambda\in\mathbb{N}}$ are *computationally indistinguishable* (see [KL21, §8.8] and [Lin17, §6.2]) if, for each nonuniform PPT distinguisher D, there is a negligible function μ for which, for each $a \in \{0,1\}^*$ and $\lambda \in \mathbb{N}$,

$$|\Pr[D(Y_0(a,\lambda)) = 1] - \Pr[D(Y_1(a,\lambda)) = 1]| \le \mu(\lambda).$$

in this case, we write $Y_0 \overset{c}{\equiv} Y_1$.

We write U_λ for the uniform distribution on λ-bit strings. We have the classical notion of a pseudorandom generator (see [KL21, §8.30]):

Definition 2.6. Consider a polynomial $l(\lambda)$ for which $l(\lambda) > \lambda$ for each λ, and a deterministic algorithm G which, on any λ-bit input $s \in \{0,1\}^\lambda$, outputs an $l(\lambda)$-bit string $G(s) \in \{0,1\}^{l(\lambda)}$. We say that G is a *pseudorandom generator* if the distributions $\{G(U_\lambda)\}_{\lambda\in\mathbb{N}}$ and $\left\{U_{l(\lambda)}\right\}_{\lambda\in\mathbb{N}}$ are computationally indistinguishable.

We recall the definition of a *group-generation algorithm* $\mathcal{G}$, which, on input 1^λ, outputs a cyclic group $\mathbb{G}$, its prime order p (with bit-length λ), and a generator $g \in \mathbb{G}$ (see [KL21, §9.3.2]). We recall the notions whereby the *discrete logarithm problem is hard relative to* $\mathcal{G}$ (see [KL21, Def. 9.63]) and the *decisional Diffie–Hellman problem is hard relative to* $\mathcal{G}$ (see [KL21, Def. 9.64]).

We define the security of key-exchange protocols Ξ (see [KL21, §11.3]):

Definition 2.7. The *key-exchange experiment* $\mathsf{KE}_{\Xi,\mathcal{A}}(\lambda)$ is defined as:

1. Two parties execute $\Xi(1^\lambda)$, yielding a transcript trans and a key ξ obtained by both parties.
2. A uniform bit $b \in \{0,1\}$ is chosen. If $b = 0$, $\hat{\xi} := \xi$ is assigned; if $b = 1$, $\hat{\xi} \leftarrow \{0,1\}^\lambda$ is set to a random λ-bit string.
3. $\mathcal{A}$ is given trans and $\hat{\xi}$.
4. $\mathcal{A}$ outputs a bit b'. The output of the experiment is defined to be 1 if and only if $b = b'$.

The key-exchange protocol Ξ is said to be *secure in the presence of an eavesdropper* if, for each nonuniform PPT adversary $\mathcal{A}$, there exists a negligible function μ for which $\Pr[\mathsf{KE}_{\Xi,\mathcal{A}}(\lambda) = 1] \leq \frac{1}{2} + \mu(\lambda)$.

Example 2.8. For example, we have the Diffie–Hellman protocol (see e.g. [KL21, Cons. 11.2]) relative to a group-generation algorithm $\mathcal{G}$. If the decisional Diffie–Hellman assumption holds relative to $\mathcal{G}$, then the resulting Diffie–Hellman key exchange is secure in the presence of an eavesdropper.

An *encryption scheme* is a triple of algorithms $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$; given a keypair $(pk, sk) \leftarrow \mathsf{Gen}(1^\lambda)$ and a message m, we have an encryption procedure $A \leftarrow \mathsf{Enc}_{pk}(m; r)$ and a decryption $m := \mathsf{Dec}_{sk}(A)$ (see [KL21, Def. 12.1] for more details). We define the security of encryption schemes, following [KL21, Def. 12.5]:

Definition 2.9. The *multiple encryptions experiment* $\mathsf{PubK}^{\mathsf{LR\text{-}cpa}}_{\Pi,\mathcal{A}}(\lambda)$ is defined as:

1. A keypair $(pk, sk) \leftarrow \mathsf{Gen}(1^\lambda)$ is generated and a uniform bit $b \in \{0, 1\}$ is chosen.
2. The adversary $\mathcal{A}$ is given pk and oracle access to $\mathsf{LR}_{pk,b}(\cdot, \cdot)$.
3. $\mathcal{A}$ outputs a bit $b' \in \{0, 1\}$.
4. The output of the experiment is defined to be 1 if and only if $b = b'$.

We say that $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ has *indistinguishable multiple encryptions* if, for each nonuniform PPT adversary $\mathcal{A}$, there exists a negligible function μ for which $\Pr[\mathsf{PubK}^{\mathsf{LR\text{-}cpa}}_{\Pi,\mathcal{A}}(\lambda) = 1] \leq \frac{1}{2} + \mu(\lambda)$.

An encryption scheme is *homomorphic* if, for each keypair (pk, sk), the resulting message space is a cyclic group of prime order, and the encryption function is a group homomorphism.

Example 2.10. Given a group-generation algorithm $\mathcal{G}$, we have the resulting El Gamal encryption scheme Π (see [KL21, Cons. 12.16]), which is homomorphic. If the decisional Diffie–Hellman problem is hard relative to $\mathcal{G}$, then Π has indistinguishable multiple encryptions (see [KL21, Thm. 12.6 and Thm. 12.18]).

A *commitment scheme* is a pair of probabilistic algorithms $(\mathsf{Gen}, \mathsf{Com})$; given public parameters $\mathsf{params} \leftarrow \text{Gen}(1^\lambda)$ and a message m, we have the *commitment* $A := \mathsf{Com}(\mathsf{params}, m; r)$, as well as a decommitment procedure effected by sending m and r (see [KL21, §6.6.5] for more details). For notational convenience, we often omit params. We often write $A \leftarrow \mathsf{Com}(m)$ to mean $A := \mathsf{Com}(m; r)$ for a uniformly random r.

Definition 2.11. The *commitment binding experiment* $\mathsf{Binding}_{\mathsf{Com},\mathcal{A}}(\lambda)$ is defined as:

1. Parameters $\mathsf{params} \leftarrow \mathsf{Gen}(1^\lambda)$ are generated.
2. $\mathcal{A}$ is given params and outputs (m_0, r_0) and (m_1, r_1).
3. The output of the experiment is defined to be 1 if and only if $m_0 \neq m_1$ and $\mathsf{Com}(m_0; r_0) = \mathsf{Com}(m_1; r_1)$.

We say that Com is *computationally binding* if, for each nonuniform PPT adversary $\mathcal{A}$, there exists a negligible function μ for which $\Pr[\mathsf{Binding}_{\mathsf{Com},\mathcal{A}}(\lambda) = 1] \leq \mu(\lambda)$. If $\mu = 0$, w say that Com is *perfectly binding*.

Definition 2.12. The *commitment hiding experiment* $\mathsf{Hiding}_{\mathsf{Com},\mathcal{A}}(\lambda)$ is defined as:

1. Parameters $\mathsf{params} \leftarrow \mathsf{Gen}(1^\lambda)$ are generated.
2. The adversary $\mathcal{A}$ is given input params. The experimenter chooses a uniform bit $b \in \{0, 1\}$.
3. $\mathcal{A}$ is given access to an oracle $\mathsf{LR}_{\mathsf{params},b}(\cdot, \cdot)$, where $\mathsf{LR}_{\mathsf{params},b}(m_0, m_1)$ returns a random commitment $A \leftarrow \mathsf{Com}(m_b)$.
4. The adversary $\mathcal{A}$ outputs a bit b'. The output of the experiment is 1 if and only if $b = b'$.

We say that Com is *computationally hiding* if, for each nonuniform PPT adversary $\mathcal{A}$, there exists a negligible function μ for which $\Pr[\mathsf{Hiding}_{\mathsf{Com},\mathcal{A}}(\lambda) = 1] \leq \frac{1}{2} + \mu(\lambda)$. If $\mu = 0$, we say that Com is *perfectly hiding.*

A commitment scheme is *homomorphic* if, for each params, its message space is a cyclic group of prime order, and its commitment function is a group homomorphism.

Example 2.13. We recall the Pedersen commitment scheme (see e.g. Hazay and Lindell [HL10, Prot. 6.5.3]) with respect to a group-generation algorithm $\mathcal{G}$. We use the variant in which $(\mathbb{G}, p, g, h) \leftarrow \mathsf{Gen}(1^\lambda)$ outputs a global, fixed "Pedersen base" between whose elements g and h no discrete logarithm relation is known. This can be achieved by assigning to h the output of a random oracle query. The resulting commitment scheme is non-interactive. The Pedersen scheme is perfectly hiding. If the discrete logarithm problem is hard with respect to $\mathcal{G}$, then the scheme is also computationally binding.

2.3 Secure three-party computation

We record security definitions for secure multi-party computation. We mainly follow Lindell [Lin17]. We have the notions of *functionalities* $\mathcal{F}$ and *protocols* Π. Motivated by our particular application, we specialize to the case of *three-party* protocols—involving players P_0, P_1, and P_2—for deterministic, *two-party* functionalities, in which only P_0 and P_1 participate, and moreover all parties receive a single identical output bit v.

In fact, we consider only a single sort of functionality, which we presently describe. Our functionality captures the *commitment-consistent* computation of some fixed boolean function $f : \{0,1\}^n \rightarrow \{0,1\}$. It operates in two stages. In the first, it solicits inputs; in the second, it evaluates f on its inputs and reports the output to all parties. Between the two phases, an arbitrary delay may occur. We describe this functionality now. For notational simplicity, we treat only the case in which both parties have equally-sized inputs.

FUNCTIONALITY 2.14 ($\mathcal{F}$—main functionality)**.**
All parties have a function $f : \{0,1\}^n \rightarrow \{0,1\}$, where n is even. Consider two players, P_0 and P_1, and a server P_2.

- **First phase:** P_0 and P_1 send elements $\mathbf{x}_0$ and $\mathbf{x}_1$ of $\{0,1\}^{n/2}$ to $\mathcal{F}$.
- **Second phase:** $\mathcal{F}$ concatenates $\mathbf{x} := \mathbf{x}_0 \parallel \mathbf{x}_1$ and evaluates $v := f(\mathbf{x})$. $\mathcal{F}$ then sends v to P_0, P_1, and P_2.

For notational convenience, we stipulate permanently in what follows that $\mathbf{x}_2 = \varnothing$.

Definition 2.15. Fix a functionality of the form $\mathcal{F}$, a three-party protocol Π, and a simulator $\mathcal{S}$. Fix a corrupt party $C \in \{0,1,2\}$. Consider the distributions:

- $\mathsf{Real}_\Pi(\lambda, C; \mathbf{x}_0, \mathbf{x}_1)$: Generate a run of Π, with security parameter λ, in which all parties behave honestly, and P_0 and P_1 use the inputs $\mathbf{x}_0$ and $\mathbf{x}_1$. Output P_C's view V_C.
- $\mathsf{Ideal}_{\mathcal{F},\mathcal{S}}(\lambda, C; \mathbf{x}_0, \mathbf{x}_1)$: Compute $v := \mathcal{F}(\mathbf{x}_0, \mathbf{x}_1)$. Output $S(1^\lambda, C, \mathbf{x}_C, v)$.

We say that Π *securely computes* $\mathcal{F}$ *in the presence of one semi-honest corruption* if, for each choice of corrupt party $C \in \{0,1,2\}$, there exists a probabilistic polynomial-time simulator $\mathcal{S}$ such that:

$$\{\mathsf{Real}_\Pi(\lambda, C; \mathbf{x}_0, \mathbf{x}_1)\}_{(\mathbf{x}_\nu)_{\nu\in\{0,1\}};\lambda\in\mathbb{N}} \overset{c}{\equiv} \{\mathsf{Ideal}_{\mathcal{F},\mathcal{S}}(\lambda, C; \mathbf{x}_0, \mathbf{x}_1)\}_{(\mathbf{x}_\nu)_{\nu\in\{0,1\}};\lambda\in\mathbb{N}},$$

where we require that $\mathbf{x}_0$ and $\mathbf{x}_1$ have equal lengths.

Definition 2.16. Fix a functionality of the form $\mathcal{F}$, a three-party protocol Π, a real-world adversary $\mathcal{A}$, and a simulator $\mathcal{S}$. Fix a corrupt party $C \in \{0,1,2\}$. Consider the distributions:

- $\mathsf{Real}_{\Pi,\mathcal{A}}\left(\lambda, C, (\mathbf{x}_\nu)_{\nu\neq C}\right)$: Generate a run of Π, with security parameter λ, in which each honest party P_ν, $\nu \neq C$, uses the input $\mathbf{x}_\nu$ and the messages of P_C are controlled by $\mathcal{A}$. Write v_ν, $\nu \neq C$, for the honest parties' outputs, and V_C for the view of P_C. Output $\left(V_C, (v_\nu)_{\nu\neq C}\right)$.
- $\mathsf{Ideal}_{\mathcal{F},\mathcal{S}}\left(\lambda, C, (\mathbf{x}_\nu)_{\nu\neq C}\right)$: Run $S(1^\lambda, C)$ until it produces an input $\mathbf{x}_C$ (or else outputs $\perp$, in which case $\mathcal{F}$ outputs $\perp$ to all parties and terminates). Compute $v := \mathcal{F}(\mathbf{x}_0, \mathbf{x}_1)$. Give v to $\mathcal{S}$. Writing V_C for the simulated view output by $\mathcal{S}$, output $\left(V_C, (v)_{\nu\neq C}\right)$.

We say that Π *securely computes $\mathcal{F}$ in the presence of one static malicious corruption* if, for each corrupt party $C \in \{0, 1, 2\}$ and real-world nonuniform PPT adversary $\mathcal{A}$ corrupting C, there exists an expected polynomial-time simulator $\mathcal{S}$ corrupting C in the ideal world such that

$$\left\{\mathsf{Real}_{\Pi,\mathcal{A}}\left(\lambda, (\mathbf{x}_\nu)_{\nu \neq C}\right)\right\}_{(\mathbf{x}_\nu)_{\nu \neq C};\lambda\in\mathbb{N}} \overset{c}{\equiv} \left\{\mathsf{Ideal}_{\mathcal{F},\mathcal{S}}\left(\lambda, (\mathbf{x}_\nu)_{\nu \neq C}\right)\right\}_{(\mathbf{x}_\nu)_{\nu \neq C};\lambda\in\mathbb{N}},$$

where we require that all (nonempty) inputs $\mathbf{x}_\nu$, for $\nu \in \{0, 1\} - \{C\}$, have equal lengths.

Remark 2.17. Our formulation Definition 2.16 *includes* the "fairness" property whereby the corrupted party receives its output if and only if the honest parties also receive theirs (see [HL10, §1.1] for further discussion). In order to relax this guarantee, one would, in the definition of $\mathsf{Ideal}_{\mathcal{F},\mathcal{S}}\left(\lambda, (\mathbf{x}_\nu)_{\nu \neq C}\right)$, give $\mathcal{S}$ the option—*after* it receives the result v—to output $\perp$ to $\mathcal{F}$, which in turn would output $\perp$ to all honest parties. The final output of $\mathsf{Ideal}_{\mathcal{F},\mathcal{S}}\left(\lambda, (\mathbf{x}_\nu)_{\nu \neq C}\right)$ would then be $(V_C, (\perp)_{\nu \neq C})$.

Remark 2.18. In the real world, we also allow that the adversary $\mathcal{A}$ be *rushing*, in the sense that, in each round, it receives the other parties' messages before sending its own (see [Lin17, §6.6.2] for discussion).

2.4 Zero-knowledge proofs

We present definitions for Σ-protocols and honest-verifier zero-knowledge proofs, following the monograph of Hazay and Lindell [HL10, §6].

We fix a binary relation $R \subset \{0, 1\}^* \times \{0, 1\}^*$, whose elements (x, w) satisfy $|w| = \mathsf{poly}(|x|)$ for some polynomial poly. If $(x, w) \in R$, we call x a *statement* and w its *witness*.

We recall the general notion of 3-move, public-coin interactive protocols between PPT machines P and V, captured in the general template [HL10, Prot. 6.2.1]. We reproduce this template here:

PROTOCOL 2.19 (General protocol template for relation R)**.**

- **Common input.** The prover P and the verifier V both have x.
- **Private input.** The prover P has a witness w such that $(x, w) \in R$.
- **The protocol.**
  1. The prover P sends an initial message a to the verifier V.
  2. The verifier V sends a random λ-bit string e to P.
  3. P sends a reply z.
  4. V chooses to *accept* or *reject* based only on the data (x, a, e, z).

We write $\langle P(\lambda, x, w), V(\lambda, x)\rangle$ for the transcript of a random such interaction between P and V. We have the formal notion of *Σ-protocols* [HL10, Def. 6.2.2], which we reproduce here:

Definition 2.20. A protocol Π of the form Protocol 2.19 is said to be a *Σ-protocol* for the relation R if the following conditions hold:

- **Completeness.** If P and V follow the protocol on inputs (x, w) and x, respectively, where $(x, w) \in R$, then V always accepts.
- **Special soundness.** There exists a polynomial-time extractor X which, given any x and accepting transcripts (a, e, z) and (a, e', z') on x where $e \neq e'$, outputs a witness w for which $(x, w) \in R$.
- **Honest verifier zero knowledge.** There exists a polynomial-time machine M which, on inputs λ and x, outputs a random transcript (a, e, z) whose distribution equals that of an honest interaction between P and V. That is, the distributions $M(\lambda, x)$ and $\langle P(\lambda, x, w), V(\lambda, x)\rangle$ are identical.

Example 2.21. A simple Schnorr proof (see [KL21, Fig. 13.2] and [HL10, Prot. 6.1.1]) can be used to show that two homomorphic commitments open to the same message. We have the relation:

$$R_{\mathsf{ComEq}} = \{(\mathsf{params}, A_0, A_1; m_0, r_0, m_1, r_1) \mid A_0 = \mathsf{Com}(m_0; r_0) \wedge A_1 = \mathsf{Com}(m_1; r_1) \wedge m_0 = m_1\}.$$

The protocol is essentially a Schnorr proof on the difference between A_0 and A_1:

PROTOCOL 2.22 (ComEq)**.**

- **Common input.** The prover P and the verifier V both have $\mathsf{params} \leftarrow \mathsf{Gen}(1^\lambda)$ and (A_0, A_1).
- **Private input.** The prover P has openings (m_0, r_0, m_1, r_1) such that $A_0 = \mathsf{Com}(m_0; r_0)$ and $A_1 = \mathsf{Com}(m_1; r_1)$, and moreover $m_0 = m_1$.
- **The protocol.**
  1. The prover P commits $a \leftarrow \mathsf{Com}(0; r)$ (for random $r \in \mathbb{F}_p$) and sends a to the verifier.
  2. The verifier V samples $e \leftarrow \mathbb{F}_p$ and sends e to the prover.
  3. P sets $z := (r_1 - r_0) \cdot e + r$ and sends z to the verifier.
  4. V outputs 1 if and only if $\mathsf{Com}(0; z) \overset{?}{=} \left(A_1 \cdot A_0^{-1}\right)^e \cdot a$.

The following theorem is essentially proven in [HL10, §§6.1–6.2]:

Theorem 2.23. *The protocol ComEq is a Σ-protocol for the relation R_{ComEq}.*

We now recall two important protocols introduced by Groth and Kohlweiss [GK15].

Example 2.24. We have "bit-commitment" proofs, which demonstrate knowledge of an opening (m, r) to a public commitment V for which m is a *bit*. More precisely:

$$R_{\mathsf{BitProof}} = \{(\mathsf{params}, A; m, r) \mid A = \mathsf{Com}(m; r) \wedge m \in \{0, 1\}\}.$$

We write BitProof for the protocol [GK15, Fig. 1]. We recall the following result:

Theorem 2.25 (Groth–Kohlweiss [GK15, Thm. 2])**.** *BitProof is a Σ-protocol for the relation R_{BitProof}.*

We have the following slightly more sophisticated version of Definition 2.20:

Definition 2.26. A protocol Π of the form Protocol 2.19 is said to be a *computational Σ-protocol* for the relation R if the following conditions hold:

- **Completeness.** If P and V follow the protocol on inputs (x, w) and x, respectively, where $(x, w) \in R$, then V always accepts.
- **Computational special $\tau(\lambda)$-soundness.** There exists a polynomial-time extractor X such that, for each nonuniform PPT adversary $\mathcal{A}$ outputting statements x and sets of accepting transcripts $(a, e_i, z_i)_{i=0}^{\tau(\lambda)-1}$ on x with distinct challenges e_i, there exists a negligible function $\mu(\lambda)$ such that X outputs a witness w for x with probability at least $1 - \mu(\lambda)$.
- **Computational honest verifier zero knowledge.** There exists a polynomial-time machine M which, given inputs λ and x, outputs a random transcript (a, e, z) on x, for which $\{M(\lambda, x)\}_{(x,w)\in R;\lambda\in\mathbb{N}} \overset{c}{\equiv} \{\langle P(\lambda, x, w), V(x)\rangle\}_{(x,w)\in R;\lambda\in\mathbb{N}}$.

Example 2.27. We recall "one-out-of-many" proofs, which demonstrate knowledge of a secret *element* of a public list of commitments and an opening of that element to 0. More precisely:

$$R_{\mathsf{OneOutOfMany}} = \{(\mathsf{params}, (A_0, \ldots, A_{m-1}); l, r) \mid A_l = \mathsf{Com}(0; r)\},$$

We write OneOutOfMany for the protocol [GK15, Fig. 2]. We recall the following result:

Theorem 2.28 (Groth–Kohlweiss [GK15, Thm. 3]). *Suppose that* Com *is hiding and binding. Then* OneOutOfMany *is a computational Σ-protocol with* $\log m + 1$*-special soundness.*

Because we work in the random oracle model, we only consider instantiations of Protocol 2.19 to which the Fiat–Shamir transform (see [KL21, Cons. 13.9]) has been applied. That is, P submits the initial message a to the random oracle, and obtains the challenge e; the proof consists of (a, e, z). While verifying the proof, V recomputes e from a using a second oracle query. We write $\pi := (a, e, z) \leftarrow P(x, w)$ for a (random) non-interactive proof obtained by the prover on input (x, w), and write $V(\pi, x)$ for the bit indicating whether V accepts on π and x. We will call such protocols *non-interactive Σ-protocols.*

Because we use (non-interactive) Σ-protocols as subroutines of larger secure protocols, we need a stronger security property than that guaranteed by Definitions 2.20 and 2.26. Indeed, it is not enough that the extractor X be able to extract a witness *given* two (or more) accepting transcripts on the same initial message; rather, a simulator $\mathcal{S}$ (in the sense of Definition 2.16) must first *produce* these transcripts from any successful prover P^* (if P^* outputs a successful proof). This issue is discussed extensively in Hazay and Lindell [HL10, §§6.5.2–6.5.3], who ultimately prove—in [HL10, Thm. 6.5.5] and [HL10, Thm. 6.5.6]—that any Σ-protocol in the sense of Definition 2.20 admits a simulator $\mathcal{S}$ in the sense of Definition 2.16.

Additional difficulties arise in our non-interactive setting, in which a malicious prover can make *many* queries to the random oracle. This issue is discussed in Pointcheval and Stern [PS00]. We end this section with a number of results related to [PS00], which discuss witness-extraction in the random oracle model.

Lemma 2.29. *Fix a computational Σ-protocol Π with $\tau(\lambda)$-special-soundness, and an adversary $\mathcal{A}$ making $Q(\lambda)$ queries to the random oracle. If $\mathcal{A}$ outputs a statement x and a successful proof (a, e, z) on x with probability $\varepsilon \geq 7 \cdot Q(\lambda)/2^\lambda$ within time $T(\lambda)$, then there is a second machine $\mathcal{M}$ which runs in strict time $T' = 16 \cdot \tau(\lambda) \cdot T(\lambda) \cdot Q(\lambda)/\varepsilon$, and which outputs a witness w for x with probability at least $\frac{1}{11}$.*

Proof. We adapt [PS00, Lem. 1] to the setting in which $\tau(\lambda)$ total transcripts are needed, as opposed to just 2, and in which the Σ-protocol is only computationally sound. For conciseness, we do not fully replicate the proof of [PS00, Lem. 1], but rather only indicate the places in which ours differs. We begin by describing a machine $\mathcal{A}'$ which extracts accepting transcripts from $\mathcal{A}$. Exactly as in [PS00, Lem. 1], after rerunning $\mathcal{A}$ $2/\varepsilon$ times, $\mathcal{A}'$ may, with probability at least $\frac{1}{5}$, obtains a single successful proof with a "likely query index" and an initial random tape which "often" yields successful proofs on that index (we refer to [PS00, Lem. 1] for precise statements). By replaying $\mathcal{A}$ with identical random oracle queries up to the appropriate query index, $\mathcal{A}'$ may obtain a *further* proof on the same initial message with probability at least $\varepsilon/(14 \cdot Q(\lambda))$. Our proof differs from [PS00, Lem. 1] only in that $\mathcal{A}'$ replays this latter procedure $14 \cdot \tau(\lambda) \cdot Q(\lambda)/\varepsilon$ times (we add the additional factor $\tau(\lambda)$). Because the median of the binomial distribution with success probability $\varepsilon/14 \cdot Q(\lambda)$ and $14 \cdot \tau(\lambda) \cdot Q(\lambda)/\varepsilon$ trials is at least $\tau(\lambda) - 1$, $\mathcal{A}'$ thereby obtains, throughout this process, at least $\tau(\lambda) - 1$ additional signatures on the same initial message with probability at least $\frac{1}{2}$. It follows that with probability at least $\frac{1}{5} \cdot \frac{1}{2} = \frac{1}{10}$, $\mathcal{A}'$ obtains a full accepting tree $(a, e_i, z_i)_{i=0}^{\tau(\lambda)-1}$ on x after $2/\varepsilon + 4 \cdot \tau(\lambda) \cdot Q(\lambda)/\varepsilon \leq 16 \cdot \tau(\lambda) \cdot Q(\lambda)/\varepsilon$ runs of $\mathcal{A}$.

Finally, $\mathcal{M}$ may first run $\mathcal{A}'$ on $\mathcal{A}$. Because $\mathcal{A}'$ is strictly polynomial-time, it satisfies hypothesis of Definition 2.26's soundness condition. We conclude that, conditioned on $\mathcal{A}'$'s success, X outputs a witness w for x with overwhelming probability; $\mathcal{M}$'s success probability thus clearly exceeds $\frac{1}{11}$ for large enough λ. □

Lemma 2.30. *Fix a Σ-protocol Π. Then given an arbitrary prover $\mathcal{A}$, there is an expected polynomial-time simulator $\mathcal{S}$ which generates statements x and proofs (a, e, z) whose distribution exactly matches that output by of $\mathcal{A}$, and which, if (a, e, z) is accepting, also outputs a witness w for x with overwhelming probability.*

Proof. We essentially combine the ideas of [PS00, Lem. 1] and [HL10, Thm. 6.5.6]. $\mathcal{S}$ begins by running $\mathcal{A}$ in a "straight-line" manner and obtaining x and (a, e, z). If the proof is not accepting, $\mathcal{S}$ terminates immediately. Otherwise, $\mathcal{S}$ begins an "extraction" phase. As in [HL10, Thm. 6.5.6], $\mathcal{S}$ begins by estimating $\mathcal{A}$'s success probability $\varepsilon(x)$. To do this, $\mathcal{S}$ repeatedly runs $\mathcal{A}$ with fresh coins until it obtains a total of $12 \cdot \lambda$ accepting proofs, after a total of N total attempts, say; $\mathcal{A}$ then sets the estimate $\tilde{\varepsilon}(x) := 12 \cdot \lambda/N$. If $\tilde{\varepsilon}(x) < 14 \cdot Q(\lambda)/2^\lambda$, $\mathcal{S}$ aborts. Otherwise, $\mathcal{S}$ runs the machine $\mathcal{M}$ of [PS00, Thm. 1] on $\mathcal{A}$, with the caveat that $\mathcal{S}$ aborts if its *total* runtime exceeds 2^λ.

We first study the expected runtime of $\mathcal{S}$. With probability at most 2^λ, $\tilde{\varepsilon}(x) \notin \left[\frac{2}{3} \cdot \varepsilon(x), 2 \cdot \varepsilon(x)\right]$ (this is exactly as in [HL10, Thm. 6.5.6]). In this setting, $\mathcal{S}$ runs for at most 2^λ steps in any case, so that at most constant additional expected runtime is accrued. We thus assume that $\tilde{\varepsilon}(x) \in \left[\frac{2}{3} \cdot \varepsilon(x), 2 \cdot \varepsilon(x)\right]$. If $\mathcal{S}$ aborts upon learning $\tilde{\varepsilon}(x)$, then $\varepsilon(x) \leq \frac{3}{2} \cdot \tilde{\varepsilon}(x) < 21 \cdot Q(\lambda)/2^\lambda$, which is negligible; in this setting $\mathcal{S}$ thus only enters the extraction phase *in the first place* with negligible probability, and so loses nothing by aborting. Otherwise, $\varepsilon(x) \geq \frac{1}{2} \cdot \tilde{\varepsilon}(x) \geq 7 \cdot Q(\lambda)/2^\lambda$, so that the hypothesis of [PS00, Thm. 1] holds, and in particular $\mathcal{S}$ obtains (a, e', z') on x after an expected $84480 \cdot Q(\lambda) \cdot T(\lambda)/\varepsilon$ steps, where $T(\lambda)$ is $\mathcal{A}$'s runtime. $\mathcal{S}$ thus runs for expected time:

$$1 + \varepsilon(x) \cdot \frac{84480 \cdot Q(\lambda) \cdot T(\lambda)}{\varepsilon(x)} = 1 + 84480 \cdot Q(\lambda) \cdot T(\lambda),$$

which is polynomial in λ.

Finally, among those runs in which $\mathcal{A}$ outputs a successful proof, $\mathcal{S}$ fails to output a witness only if $\tilde{\varepsilon}(x) \notin \left[\frac{2}{3} \cdot \varepsilon(x), 2 \cdot \varepsilon(x)\right]$, if $\varepsilon(x) < 21 \cdot Q(\lambda)/2^\lambda$, or else if $\mathcal{S}$ runs for a total of 2^λ steps. Each of these events happens with only negligible probability. □

3 Theoretical Introduction

In this section, we introduce the theory of hyperplane coverings. This section plays the dual role of situating our paradigm within existing literature and, simultaneously, of establishing many of its fundamental facts. Indeed, departing from convention, we simultaneously survey prior literature and introduce new material—much of which is theoretically challenging—in this section. We also include a handful of open problems and conjectures, laying the ground for future work.

3.1 Initial definitions and remarks

We begin with basic definitions.

Definition 3.1. Given an odd prime p, we say that a family of affine hyperplanes $\{H_i\}_{i=0}^{m-1}$ in $\mathbb{F}_p^n$ *disjointly cover* a subset $S \subset \{0,1\}^n$ if $S = \bigsqcup_{i=0}^{m-1} H_i \cap \{0,1\}^n$.

Example 3.2. For $n = 3$, we consider the subset $S := \{(0,0,0), (1,1,0), (1,0,1), (0,1,1)\} \subset \{0,1\}^3$. S can be covered by 2 hyperplanes over any odd prime field $\mathbb{F}_p$. Indeed, any three points of S can be covered by a single hyperplane; its fourth can be covered by a second. We illustrate this covering below:

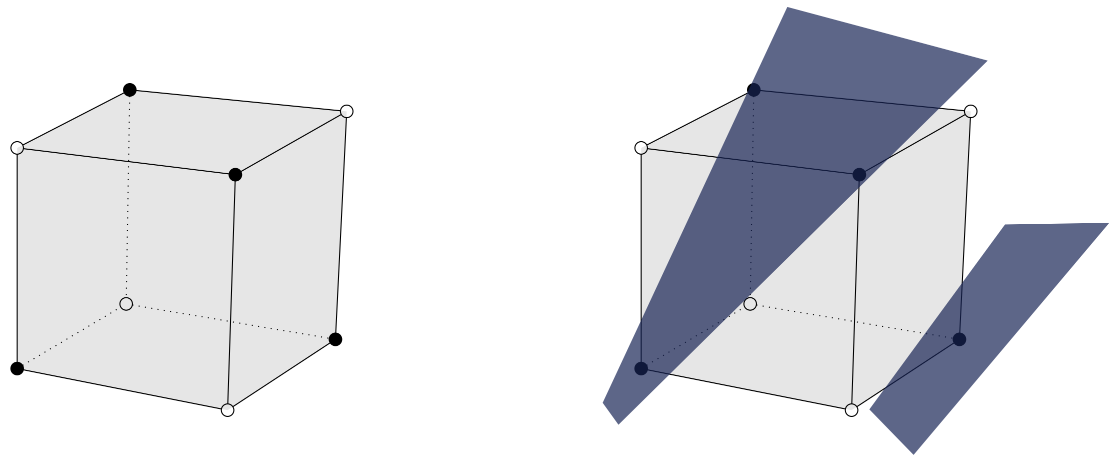

Figure 1: A depiction of an example set $S \subset \{0,1\}^3$, as well as a covering of it by hyperplanes.

We will argue in Example 3.16 below that *one* hyperplane cannot suffice to cover S.

Upon representing each hyperplane "dually" $H_i : \mathbb{F}_p^n \to \mathbb{F}_p$ as an affine-linear functional, we see that the condition of Definition 3.1 implies that degree-m multivariate polynomial $F := \prod_{i=0}^{m-1} H_i(x_0, \ldots, x_{m-1})$

evaluates to 0 exactly on $S \subset \{0,1\}^n$; that is, for each $\mathbf{x} \in \{0,1\}^n$, $\mathbf{x} \in S$ if and only if $F(\mathbf{x}) = 0$ (F can act arbitrarily on elements $\mathbf{x} \in \mathbb{F}_p^n - \{0,1\}^n$). This viewpoint yields the following schematic depiction of F:

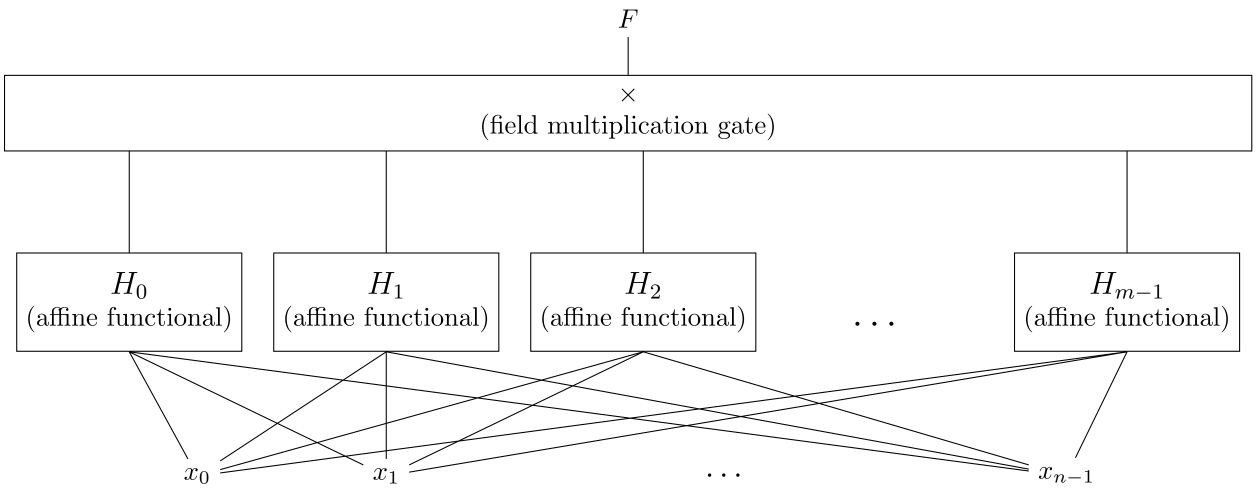
F
$\times$
(field multiplication gate)
H_0 (affine functional)
H_1 (affine functional)
H_2 (affine functional)
$\dots$
H_{m-1} (affine functional)
x_0
x_1
$\dots$
x_{n-1}

Figure 2: An arithmetic-circuit-based depiction of a hyperplane covering.

In practice, we allow that common affine sub-expressions from *within* the "affine gates" H_i be shared arbitrarily. In practical examples of interest, this can reduce the amortized cost of evaluating each *individual* affine gate from $O(n)$ to $O(1)$ (see e.g. Example 3.22). Inspired by arithmetic circuits, we will occasionally call polynomials F of this form *affine circuits*.

We are interested in disjoint hyperplane coverings $S = \bigsqcup_{i=0}^{m-1} H_i \cap \{0,1\}^n$ for which m is small. In fact, we define the following "complexity class", consisting of functions whose on-sets and off-sets can both be covered by polynomially many hyperplanes:

Definition 3.3. A sequence of functions $\{f_n : \{0,1\}^n \to \{0,1\}\}_{n \in \mathbb{N}}$ is *efficiently computable by affine hyperplanes* if, for each $n \in \mathbb{N}$, there exists an n-bit prime p and disjoint coverings $f^{-1}(0) = \bigsqcup_{i=0}^{m-1} H_{0,i} \cap \{0,1\}^n$ and $f^{-1}(1) = \bigsqcup_{i=0}^{m-1} H_{1,i} \cap \{0,1\}^n$ using $\mathbb{F}_p$-hyperplanes, where m moreover grows polynomially in n.

Many natural functions are efficiently computable by hyperplanes (see Subsection 3.2 below for a thorough treatment). On the other hand, the following result in the spirit of Shannon [Weg87, §4] shows that the vast majority of functions $f_n : \{0,1\}^n \to \{0,1\}$ require exponentially many affine hyperplanes to compute. Following [Weg87, §1.2], we write B_n for the set of all subsets $S \subset \{0,1\}^n$; the magnitude of B_n is 2^{2^n}.

Theorem 3.4. *Fix an unbounded function $\omega(1)$. For each $n \in \mathbb{N}$ and each prime p of n bits, at least $|B_n| \cdot (1 - 2^{-\frac{1}{2}\cdot\omega(1)})$ among B_n's elements fail to be coverable using any fewer than $\frac{2^n - \omega(1)}{n^2}$ $\mathbb{F}_p$-hyperplanes.*

Proof. Following the general strategy of Wegener [Weg87, §4.2], we bound from above the total number of distinct subsets $S \subset \{0,1\}^n$ which can possibly be covered using m hyperplanes. For a fixed p, the total number of distinct affine hyperplanes $H \subset \mathbb{F}_p^n$ is given by the Gaussian binomial coefficient expression $p \cdot \begin{bmatrix} n \\ n-1 \end{bmatrix}_p$ (see Cameron [Cam95, Prop. 2.9]), which we bound as follows:

$$p \cdot \begin{bmatrix} n \\ n-1 \end{bmatrix}_p = p \cdot \prod_{i=0}^{n-2} \frac{p^n - p^i}{p^{n-1} - p^i} \le p \cdot \left(\frac{p}{p-1} \right) \cdot \prod_{i=0}^{n-2} \frac{p^n}{p^{n-1}} = \frac{p^{n+1}}{p-1}.$$

Using the inequality $p < 2^n$ imposed by Definition 3.3, we see that the number of subsets of $\{0,1\}^n$ coverable by m hyperplanes is bounded from above by:

$$S(n,m) := \left(\frac{(2^n)^{n+1}}{2^n - 1} \right)^m = \left(\frac{2^{n^2+n}}{2^n - 1} \right)^m.$$

For fixed $\omega(1)$ as in the hypothesis of the theorem, we see that for any $m(n) < \frac{2^n - \omega(1)}{n^2}$,

$$\log S(n, m(n)) < \left(n^2 + \frac{1}{2^n}\right) \cdot \left(\frac{2^n - \omega(1)}{n^2}\right) = O(1) + 2^n - \omega(1) < 2^n - \frac{1}{2} \cdot \omega(1),$$

where in the first inequality we use $\log\left(\frac{2^{n^2+n}}{2^n-1}\right) = n^2 + n - \log(2^n - 1) \leq n^2 + n - \left(\log(2^n) - \frac{1}{2^n}\right)$, itself a consequence of $\log(2^n) - \log(2^n - 1) < \frac{1}{2^n}$. The final inequality above holds for sufficiently large n.

We thus see $m(n)$ hyperplanes or fewer eventually fail to exhaust the elements of any subset $B_n^* \subset B_n$ for which $2^n - \frac{1}{2} \cdot \omega(1) \leq \log |B_n^*|$. In particular, we may as well choose as B_n^* the set consisting of those $2^{2^n - \frac{1}{2} \cdot \omega(1)}$ subsets of B_n which require the fewest hyperplanes to cover. But B_n^* represents a vanishing proportion of B_n, as:

$$\frac{|B_n^*|}{|B_n|} = \frac{2^{2^n - \frac{1}{2}\cdot\omega(1)}}{2^{2^n}} = 2^{-\frac{1}{2}\cdot\omega(1)}.$$

This completes the proof. ☐

Question 3.5. *Can we exhibit a concrete function family $\{f_n : \{0,1\}^n \to \{0,1\}\}_{n \in \mathbb{N}}$ each of whose elements requires exponentially many hyperplanes to compute over any n-bit prime?*

Corollary 3.13 below shows that 2^n hyperplanes suffice to cover any set $S \subset \{0,1\}^n$. The following conjecture would establish the "Shannon effect" (see [Weg87, §4, Def. 1.3]) for Definition 3.3:

Conjecture 3.6. *Any subset $S \subset \{0,1\}^n$ can be covered by $\frac{2^n}{n^2}$ disjoint $\mathbb{F}_p$-hyperplanes for an n-bit prime p.*

We record the additional open questions:

Question 3.7. *Would relaxing the requirement that the coverings $\{H_{0,i}\}_{i=0}^{m-1}$ and $\{H_{1,i}\}_{i=0}^{m-1}$ in Definition 3.3 be disjoint strictly enlarge the resulting complexity class?*

Question 3.8. *Would allowing that p have polynomially bits in Definition 3.3 strictly enlarge the resulting complexity class?*

Question 3.9. *Would requiring that p have fewer than n bits in Definition 3.3 strictly shrink the resulting complexity class?*

3.2 A perspective from logic synthesis

Finding minimal-cardinality coverings of subsets $S \subset \{0,1\}^n$ by hyperplane intersections $H_i \cap \{0,1\}^n$ represents, in practice, a non-trivial "logic synthesis" problem, which admits important analogies to classical problems.

A highly classical problem seeks to cover sets $S \subset \{0,1\}^n$ using minimally many *subcubes*, namely, subsets of the form $C = \left\{(x_0, \ldots, x_{n-1}) \in \{0,1\}^n \;\middle|\; x_{c_0} = y_{c_0}, \ldots, x_{c_{n-k-1}} = y_{c_{n-k-1}}\right\}$, where $\{c_0, \ldots, c_{n-k-1}\} \subset \{0, \ldots, n-1\}$ is a subsequence and $y_{c_0}, \ldots, y_{c_{n-k-1}}$ are binary constants. Equivalently, a k-dimensional subcube is the solution set of an AND of $n-k$ literals and negated literals. The original algorithm for this problem is that of Quine and McCluskey [MJ56]. The "Espresso II" logic minimizer of Brayton, Hachtel, McMullen and Sangiovanni-Vincentelli [BHMSV84] vastly advanced this field, and introduced many new techniques.

The following result relates our paradigm to that of classical logic minimization:

Lemma 3.10. *Suppose that $p > n$. Then for each proper subcube $C \subset \{0,1\}^n$, there exists an affine hyperplane $H : \mathbb{F}_p^n \to \mathbb{F}_p$ for which $C = H \cap \{0,1\}^n$.*

Proof. We write $C = \left\{(x_0, \ldots, x_{n-1}) \in \{0,1\}^n \;\middle|\; x_{c_0} = y_{c_0}, \ldots, x_{c_{n-k-1}} = y_{c_{n-k-1}}\right\}$. By Lemma 2.1, it suffices to assume that C contains the origin, and hence that each $y_{c_i} = 0$. The affine map $H : (x_0, \ldots, x_{n-1}) \mapsto \sum_{i=0}^{n-k-1} x_{c_i}$ suffices to define $C \subset \{0,1\}^n$. Indeed, on each element $\mathbf{x} \in \{0,1\}^n$, $H(\mathbf{x})$ is a sum consisting of $n-k$ terms—each in $\{0,1\}$—at least one of which is nonzero if and only if $\mathbf{x} \notin C$. By hypothesis on p, each nonzero such sum necessarily fails to overflow, and thus remains unequal to 0 even modulo p. ☐

Corollary 3.11. *If $S \subset \{0,1\}^n$ has a disjoint sum-of-products expression with m summands, then, for any $p > n$, S can be disjointly covered by m affine hyperplanes over $\mathbb{F}_p$.*

Example 3.12. The comparison algorithm of Wagh, Gupta, and Chandran [WGC19, Alg. 3] exploits this basic property of subcubes. Indeed, [WGC19, Alg. 3] observes—though using different terminology—that the function which compares an n-bit input integer (written in binary) to a *fixed* n-bit integer ("hardcoded" in the function) evaluates to true exactly on a union of n disjoint subcubes of $\{0,1\}^n$. (This can be seen directly by specializing one of the two inputs of the circuit of Fig. 3 below.) The protocol [WGC19, Alg. 3], in particular, takes $n = 64$ and $p = 67$, and essentially applies Corollary 3.11.

The following consequence of Corollary 3.11 can be viewed as a "completeness theorem" for hyperplane coverings:

Corollary 3.13. *Given any subset $S \subset \{0,1\}^n$ and any prime $p > n$, S can be disjointly covered by at most 2^n affine hyperplanes over $\mathbb{F}_p$.*

Proof. S's trivial "minterm representation" expresses it as the disjoint union of its at-most 2^n points, each of which is a subcube; the result follows from Corollary 3.11. $\square$

The following example shows that Lemma 3.10's requirement that $p > n$ can't be *completely* dropped:

Example 3.14. We fix $p = 2$. Any hyperplane $H \subset \mathbb{F}_2^n$ satisfies $H \cap \{0,1\}^n = 2^{n-1}$. This shows that only $n-1$-dimensional subcubes can be represented as hyperplane intersections $C = H \cap \{0,1\}^n$ in $\mathbb{F}_2^n$.

It is wide-open to close the gap between p *odd* and $p > n$:

Question 3.15. *Can the hypothesis $p > n$ of Lemma 3.10 be weakened? If so, by how much?*

A more general logic synthesis paradigm was initiated by Luccio and Pagli [LP99], who studied coverings of subsets $S \subset \{0,1\}^n$ by *pseudocubes*, and proposed appropriate generalizations of the Quine–McCluskey algorithm. Though a number of equivalent characterizations exist, a k-dimensional pseudocube is—roughly—a subset of the cube evaluated by an AND of $n-k$ XOR expressions, each of which in turn may involve arbitrarily many of the variables $x_0, \ldots, x_{n-1}$. We refer to Luccio and Pagli [LP99] for further details.

The study of "sum-of-pseudoproducts" expressions was furthered by Ciriani [Cir01], culminating in the work [Cir03], which identified pseudocubes $C \subset \{0,1\}^n$ as exactly the affine subspaces of $\mathbb{F}_2^n$. Ciriani [Cir03] also characterized each pseudocube $C \subset \{0,1\}^n$ by the maximal arity attained across its XOR-factors; in particular, for $j \in \{1, \ldots, n\}$, *j-pseudocube* is one each of whose XOR-factors is of arity at most j (a 1-pseudocube is just a subcube).

When $p > 2$, a general j-pseudocube $C \subset \{0,1\}^n$ *cannot* be expressed as a hyperplane intersection $C = H \cap \{0,1\}^n$, as the following example shows:

Example 3.16. The set $S \subset \{0,1\}^3$ considered in Example 3.2 above is in fact a 2-dimensional 3-pseudocube, given by the arity-3 "canonical expression" [Cir01, Def. 1] $x_0 \oplus x_1 \oplus \overline{x_2}$. Fix an arbitrary odd prime $p > 2$. We argue that S can not be expressed as the intersection of a *single* $\mathbb{F}_p$-hyperplane with the cube. Indeed, any hyperplane $H : \mathbb{F}_p^3 \rightarrow \mathbb{F}_p$ containing S would also contain the difference $(1,1,0) - (1,0,1) - (0,1,1) = (0,0,-2)$, hence also $(0,0,1)$ (we use here that $2 \neq 0$). But $(0,0,1) \notin S$, so that $S \subsetneq H \cap \{0,1\}^n$. (A similar argument shows that H would also contain $(1,0,0)$ and $(0,1,0)$.)

The following result shows that 2-pseudocubes *are* necessarily representable as hyperplane intersections:

Lemma 3.17. *Suppose that $p \geq 2^{n-1}$. Then for each proper 2-pseudocube $C \subset \{0,1\}^n$, there exists an affine hyperplane $H : \mathbb{F}_p^n \rightarrow \mathbb{F}_p$ for which $C = H \cap \{0,1\}^n$.*

Proof. If C's dimension k is 0, then C is a subcube, and we may use Lemma 3.10. We thus assume $k > 0$.

We prove the lemma constructively, by defining an appropriate functional. By hypothesis, C consists of those points $\mathbf{x} \in \{0,1\}^n$ satisfying an AND of $n-k$ XOR factors, each of whose arity is at most 2. To simplify the proof, we assume by Lemma 2.1 that C contains the origin, and hence that each XOR-factor takes the form $x_{c_{l,0}} \oplus \overline{x_{c_{l,1}}}$, for indices $\{c_{l,0}, c_{l,1}\} \subset \{0, \ldots, n-1\}$ (or else $\overline{x_{c_l}}$ if the arity is 1). We first claim that for each such factor, there exists an affine functional H_l which annihilates exactly those $\mathbf{x} \in \{0,1\}^n$

satisfying the factor; indeed, $H_l : (x_0, \ldots, x_{n-1}) \mapsto x_{c_{l,0}} - x_{c_{l,1}}$ suffices (we may use $H_l : (x_0, \ldots, x_{n-1}) \mapsto x_{c_l}$ in the arity-1 case). Moreover, H_l clearly takes values in $\{-1, 0, 1\}$.

The result follows from writing $H(\mathbf{x}) := \sum_{i=0}^{n-k-1} 2^i \cdot H_l(\mathbf{x})$. It is clear that $H(\mathbf{x}) = 0$ for each $\mathbf{x}$ which satisfies all of the expression's XOR factors. Conversely, we claim that $H(\mathbf{x}) \neq 0 \pmod p$ for *any* unsatisfying argument $\mathbf{x} \in \{0,1\}^n$. Indeed, by assumption on $\mathbf{x}$, $H_l(\mathbf{x}) \neq 0$ for at least some XOR-factor $l \in \{0, \ldots, n-k-1\}$; we write i^* for the *first* such factor. Fixing an $i \in \{i^*+1, \ldots, n-k-1\}$, we assume by induction that $\sum_{j=0}^{i-1} 2^j \cdot H_j(\mathbf{x})$ is a nonzero element of the integer range $\{-2^i - 1, \ldots, 2^i - 1\}$. We claim that, regardless of $H_i(\mathbf{x})$, $\sum_{j=0}^{i} 2^j \cdot H_j(\mathbf{x})$ is a nonzero element of the integer range $\{-2^{i+1} - 1, \ldots, 2^{i+1} - 1\}$. Indeed, the top term $2^i \cdot H_l(\mathbf{x})$ is either -2^i, 0, or 2^i; adding it to a nonzero element of $\{-2^i - 1, \ldots, 2^i - 1\}$ cannot yield the sum 0. Finally, $\sum_{j=0}^{i} 2^j \cdot H_j(\mathbf{x})$ cannot exceed $(2^i - 1) + 2^i = 2^{i+1} - 1$ in absolute value.

For any unsatisfying argument $\mathbf{x} \in \{0,1\}^n$, the sum $\sum_{i=0}^{n-k-1} 2^i \cdot H_l(\mathbf{x})$ is thus a nonzero element of the integer range $\{-2^{n-k} - 1, \ldots, 2^{n-k} - 1\}$. By hypothesis on p, this nonzero integer is also nonzero modulo p. We conclude that $\mathbf{x}$ is not contained in the hyperplane H, and H satisfies the conclusion of the lemma. □

Corollary 3.18. *If $S \subset \{0,1\}^n$ has a disjoint sum-of-2-pseudoproducts expression with m summands, then, for any $p \geq 2^{n-1}$, S can be disjointly covered by m affine hyperplanes over $\mathbb{F}_p$.*

The following example shows that Lemma 3.17's hypothesis $p \geq 2^{n-1}$ is necessary, at least when $n = 3$:

Example 3.19. The subset $C = \{(0,0,0), (1,1,0)\} \subset \mathbb{F}_3^3$ is a 2-pseduocube, given by the expression $(x_0 \oplus \overline{x_1}) \wedge \overline{x_2}$. We argue that there is no hyperplane (i.e., plane) $H \subset \mathbb{F}_3^3$ for which $C = H \cap \{0,1\}^3$. Any such plane would be generated over $\mathbb{F}_3$ by $(1,1,0)$ and some second element $\mathbf{x} \in \mathbb{F}_3^3 - \{0,1\}^3$ for which $2 \cdot \mathbf{x} \notin C$. It can be checked manually that each such $\mathbf{x}$ satisfies either $2 \cdot \mathbf{x} \in \{0,1\}^3 - C$ or $\mathbf{x} + (1,1,0) \in \{0,1\}^3 - C$.

Question 3.20. *Can the hypothesis $p \geq 2^{n-1}$ of Lemma 3.17 be weakened for general n? If so, by how much?*

Question 3.21. *Is the hypothesis $j = 2$ of Lemma 3.17 always necessary? That is, can there exist any j-pseudocube $C \subset \{0,1\}^n$, with $j > 2$, for which $C = H \cap \{0,1\}^n$ for some $\mathbb{F}_p$-hyperplane H with p odd?*

Example 3.22. We consider the function $f : \{0,1\}^n \to \{0,1\}$ (for even n) which compares the integers represented in binary by its argument's respective halves; that is, $f : (x_0, \ldots, x_{n-1}) \mapsto \sum_{i=0}^{n/2-1} 2^i \cdot x_i \leq \sum_{i=n/2}^{n-1} 2^{i-n/2} \cdot x_i$. Each f is represented by the well-known boolean comparator circuit of Figure 3.

Importantly, unlike the function of Example 3.12—one of whose inputs was permanently specialized—the on-set evaluated by f_n is *not* a union of subcubes. Indeed, each of this circuit's OR gate's inputs contains AND as well as arity-2 XOR gates; each of its wires thus defines exactly a *2-pseudocube.*

This circuit's ultimate OR gate has $\frac{n}{2} + 1$ inputs. Applying Corollary 3.18, we see that $f_n^{-1}(1)$ can be exactly annihilated by $\mathbb{F}_p$-hyperplanes $H_0, \ldots, H_{n/2}$ (provided that $p \geq 2^{n/2-1}$). Evaluating each of these on an input $\mathbf{x} \in \{0,1\}^n$ would thus take $\Omega(n^2)$ time, naïvely.

Because the above circuit shares many common subexpressions, the values $H_i(\mathbf{x})$ for $i \in \{0, \ldots, \frac{n}{2}\}$ can be evaluated in $O(n)$ *total* time under a slightly more careful scheme, which we presently describe. For each $i \in \{0, \ldots, \frac{n}{2} - 1\}$, the AND gate attached to the inputs x_{n-1-i} and $x_{n/2-1-i}$ can be evaluated using the expression $1 + x_{n/2-1-i} - x_{n-1-i}$. The XNOR gate attached to x_{n-1-i} and $x_{n/2-1-i}$ can be evaluated using the expression $2^{2+i} \cdot \left(x_{n/2-1-i} - x_{n-1-i}\right)$ (we observe that the common subexpression $x_{n/2-1} - x_{n-1-i}$ can already be reused here, though the resulting savings are not asymptotic). Each remaining AND gate can be evaluated simply by adding its two input wires. An argument similar to that of Lemma 3.17 shows that—because we use powers of 2—these sums cannot spuriously yield 0. Putting these facts together, we obtain the expressions:

$$H_i(x_0, \ldots, x_{n-1}) = \left(1 + x_{n/2-1-i} - x_{n-1-i}\right) + \sum_{j<i} 2^{j+2} \cdot \left(x_{n/2-1-j} - x_{n-1-j}\right).$$

Each functional H_i's sum be evaluated in constant time, by incrementally adding one summand to that of H_{i-1}. This completes the example.

The following example of Ciriani [Cir03, p. 1311] shows an exponential separation between the sizes of sum-of-products expressions and sum-of-j-pseudoproducts, even when $j = 2$:

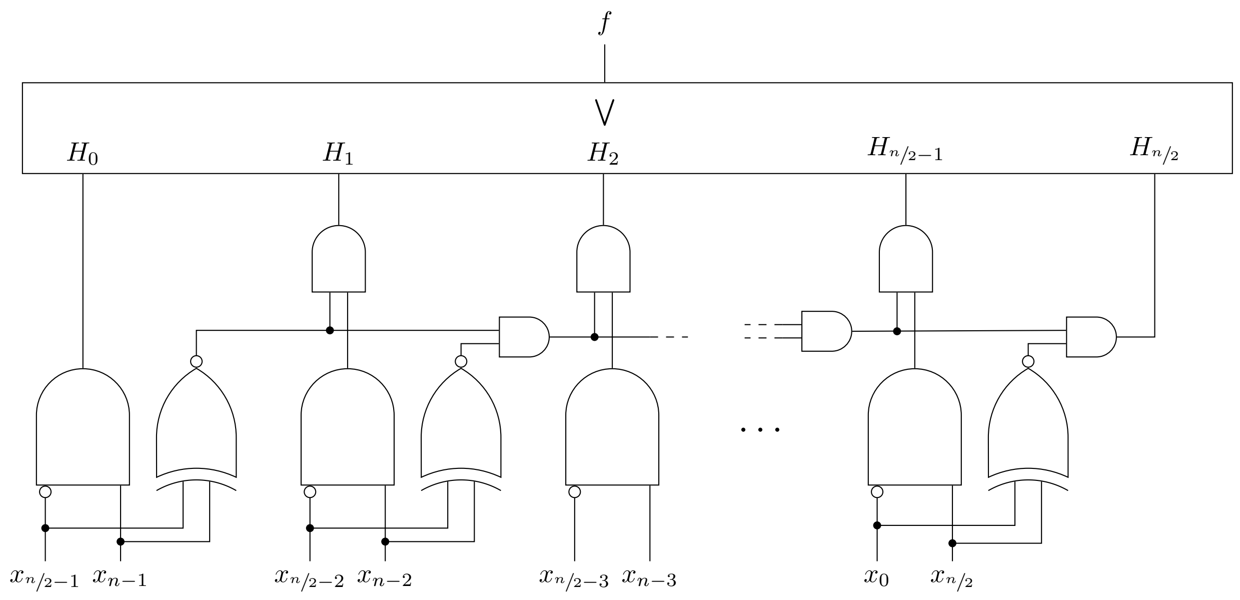

Figure 3: A standard boolean circuit evaluating the arithmetic comparison function f.

Example 3.23 (Ciriani [Cir03])**.** The function $f_n : (x_0, \ldots, x_{n-1}) \mapsto (x_0 \oplus x_1) \wedge (x_2 \oplus x_3) \wedge \cdots \wedge (x_{n-2} \oplus x_{n-1})$ (for even n) is obviously coverable by a single 2-pseudocube. On the other hand, the smallest sum-of-products expression for $f_n^{-1}(1)$ requires $2^{n/2}$ subcubes (each containing a single point) [Cir03, p. 1311].

Together with Lemma 3.17, the above example already shows that hyperplane representations can be exponentially more compact than sum-of-products expressions.

We recall the notion of *symmetric* functions $f \in S_n$ (see e.g. [Weg87, §3.4, Def. 4.1]). As it turns out, symmetric functions are easily coverable by hyperplanes:

Lemma 3.24. *Fix a symmetric function $f : \{0,1\}^n \to \{0,1\}$, and let $p > n$. Then the sets $f^{-1}(0)$ and $f^{-1}(1)$ can each be covered by $O(n)$ $\mathbb{F}_p$-hyperplanes. These hyperplanes can be evaluated in $O(n)$ total time.*

Proof. The cube $\{0,1\}^n$ has an obvious stratification by Hamming weight; that is, we write $S_{n,i} := \left\{(x_1, \ldots, x_n) \in \{0,1\}^n \;\middle|\; \sum_{j=0}^{n-1} x_j = i\right\}$ for each $i \in \{0, \ldots, n\}$. It is evident by inspection that each $S_{n,i}$ is exactly equal to $H_i \cap \{0,1\}^n$ for an appropriate $\mathbb{F}_p$-hyperplane H_i (here we use the assumption $p > n$, to rule out overflows). By hypothesis on f, both $f^{-1}(0)$ and $f^{-1}(1)$ are disjoint unions of appropriate sets $S_{n,i}$.

We finally note that the hyperplanes $\{H_i(\mathbf{x})\}_{i=0}^n$ share a common expression $\sum_{j=0}^{n-1} x_j$, which takes $O(n)$ time to evaluate and need be evaluated only once. The hyperplanes H_i differ only in their constant terms, which can be added individually to this common subexpression. □

Example 3.25. For even n, we consider the *majority* function $f : (x_0, \ldots, x_{n-1}) \mapsto \left(\sum_{i=0}^{n-1} x_i\right) \geq \frac{n}{2}$. The majority function f is of course symmetric; by Lemma 3.24 above, if $p > n$ then $f^{-1}(0)$ and $f^{-1}(1)$ both admit coverings by linearly many hyperplanes (in fact, by $\frac{n}{2}$ and $\frac{n}{2} + 1$, respectively).

The following lemma shows that majority does *not* admit a compact sum-of-pseudoproducts expression, giving a further exponential separation between sum-of-pseudoproducts and hyperplane representations:

Lemma 3.26. *Fix as $f : \{0,1\}^n \to \{0,1\}$ the majority function of Example 3.25. Then any covering of $f^{-1}(0)$ or of $f^{-1}(1)$ by pseudocubes must use $\Omega(2^{n/2})$ pseudocubes.*

Proof. We prove the lemma only for $f^{-1}(1)$; the proof for $f^{-1}(0)$ is similar (though slightly trickier).

We argue first that $f^{-1}(1)$ cannot contain *any* pseudocube of dimension strictly bigger than $\frac{n}{2}$. Indeed, any such pseudocube $C \subset \{0,1\}^n$ would have (at least) $\frac{n}{2}+1$ "canonical columns" $\{b_0, \ldots, b_{n/2}\} \subset \{0, \ldots, n-$

1}, each of whose possible combinations of values obtained in some element of C (see e.g. Ciriani [Cir01, §2]). In particular, there would exist an element $(x_0, \ldots, x_{n-1}) \in C$ for which $x_{b_i} = 0$ for each $i \in \{0, \ldots, \frac{n}{2}\}$. This would imply that $\sum_{i=0}^{n-1} x_i < (n - \frac{n}{2}) = \frac{n}{2}$, so that $(x_0, \ldots, x_{n-1}) \notin f^{-1}(1)$ and $C \not\subset f^{-1}(1)$.

Thus any covering of $f^{-1}(1)$ by pseudocubes must employ only pseudocubes of dimension at most $\frac{n}{2}$, each of which contains at most $2^{n/2}$ points. On the other hand, we claim that the set $f^{-1}(1)$ contains at least 2^{n-1} points. Indeed, $S_{n,i}$ clearly has cardinality $\binom{n}{i}$ for each $i \in \{0, \ldots, n\}$; it follows that $f^{-1}(1)$ has cardinality $\sum_{i=n/2}^{n} S_{n,i} = \sum_{i=n/2}^{n} \binom{n}{i}$, the sum of the last $n/2 + 1$ entries of Pascal's triangle's n^{th} row (i.e., including the central binomial coefficient $\binom{n}{n/2}$). We can bound this quantity from below as:

$$\sum_{i=n/2}^{n} \binom{n}{i} = \frac{2^n - \binom{n}{n/2}}{2} + \binom{n}{n/2} \geq 2^{n-1},$$

using the fact that the n^{th} row of Pascal's triangle sums to 2^n, and moreover is symmetric. We thus see that to cover $f^{-1}(1)$ using pseudocubes—each of which covers at most $2^{n/2}$ points—at least $\frac{2^{n-1}}{2^{n/2}} = 2^{n/2-1}$ pseudocubes would be required. □

The separation result of Lemma 3.26 indicates that the power of hyperplane representations is not due *merely* to the advantage of pseudocubes over subcubes (cf. Example 3.23); indeed hyperplane representations can be exponentially *more* compact still than sum-of-j-pseudoproducts expressions. This is true even over logarithmically sized primes p, when j is allowed to be arbitrary, when the hyperplanes are required to be disjoint, and when the pseudocubes are *not* required to be disjoint.

It is also interesting to consider *boolean formulas* (with fan-out 1 and where only the "De Morgan basis" $\{\wedge, \vee, \neg\}$ is allowed; see [Weg87, §8.8]). The following example shows a further asymptotic separation, this time between between hyperplane complexity and boolean formulas:

Example 3.27. We again set as $f : \{0,1\}^n \to \{0,1\}$ the majority function on n bits. It follows from an elegant result of Krapchenko [Weg87, §8.8 Thm. 8.1]) that any boolean formula for f must have size at least $s(n) \in \Omega(n^2)$ (see also [Weg87, p. 247]). For self-containedness, we outline the details here. Indeed (in the notation of Lemma 3.24), it's enough to set $A := S_{n,n/2-1} \subset f^{-1}(0)$ and $B := S_{n,n/2} \subset f^{-1}(1)$ in [Weg87, Def. 8.2]. Evaluating the resulting Krapchenko measure [Weg87, Def. 8.2], we obtain:

$$K_{AB} := \frac{H(A,B)^2}{|A| \cdot |B|} = \frac{\left(\binom{n}{n/2} \cdot \frac{n}{2}\right)^2}{\binom{n}{n/2-1} \cdot \binom{n}{n/2}} = \frac{n/2 + 1}{n/2} \cdot \left(\frac{n}{2}\right)^2 = \Omega(n^2).$$

The lower bound $s(n) \in \Omega(n^2)$ finally follows from [Weg87, §8.8 Thm. 8.1]. Example 3.25 on the other hand shows that $f^{-1}(1)$ can be covered by just $O(n)$ hyperplanes, achieving the separation.

We finally note that *further* asymptotic separations between hyperplanes and formulas can be established, even when the full fan-in 2 boolean basis B_2 (i.e., including the gates $\oplus$ and $\equiv$) is permitted in the latter:

Example 3.28. Wegener's [Weg87, §8.8 Ex. 9] shows that there exist symmetric functions $f : \{0,1\}^n \to \{0,1\}$ for which any formula over the full boolean basis B_2 is of size $L(f) \in \Omega(n \cdot \log\log n)$ (in fact, "most" symmetric functions $f \in S_n$ satisfy this property). Lemma 3.24 thus again yields a separation.

Example 3.29. Another natural example arises from the *Hamming weight comparator* function $f : (x_0, \ldots, x_{n-1}) \mapsto \sum_{i=0}^{n/2-1} x_i \leq \sum_{i=n/2}^{n-1} x_i$, for n even (this function compares its argument's two halves' Hamming weights). This function f differs from the majority function of Example 3.25 by precomposition with the affine-linear bijection $(x_0, \ldots, x_{n-1}) \mapsto (x_0, \ldots, x_{n/2-1}, 1 - x_{n/2}, \ldots, 1 - x_{n-1})$. We thus conclude directly that $f^{-1}(0)$ and $f^{-1}(1)$ again admit coverings by $\frac{n}{2}$ and $\frac{n}{2} + 1$ hyperplanes.

3.3 Relation to the work of Alon and Füredi

In an aptly titled paper, Alon and Füredi [AF93] study a problem related to ours. They show that any hyperplane configuration $H_0, \ldots, H_{m-1}$ with $m \leq n$ which doesn't cover the whole cube must "miss" at least 2^{n-m} points. Their results allow us study the role of m in Definition 3.3.

Lemma 3.30. *Fix a k-dimensional subcube $C \subset \{0,1\}^n$. The complement $\{0,1\}^n - C$ can be covered by $n-k$ disjoint subcubes.*

Proof. We prove the lemma constructively. Applying Lemma 2.1, we assume that C contains the origin, and hence that $C = \left\{(x_0, \ldots, x_{n-1}) \in \{0,1\}^n \,\middle|\, x_{c_0} = 0, \ldots, x_{c_{n-k-1}} = 0\right\}$ for some subsequence $\{c_0, \ldots, c_{n-k-1}\} \subset \{0, \ldots, n-1\}$. For each $i \in \{0, \ldots, n-k-1\}$, we define the subcube $C_i = \left\{(x_0, \ldots, x_{n-1}) \in \{0,1\}^n \,\middle|\, \bigwedge_{j<i} x_{c_j} = 0 \wedge x_{c_i} = 1\right\}$. It can be checked explicitly that the cubes $\{C_i\}_{i=0}^{n-k-1}$ are disjoint and cover $\{0,1\}^n - C$. □

Theorem 3.31 (Alon–Füredi [AF93]). *Given some fixed function $g(n) : \mathbb{N} \to \mathbb{N}$ for which $0 \le g(n) < n$ for each n, define $f_n : \{0,1\}^n \to \{0,1\}$ by $f_n : (x_0, \ldots, x_{n-1}) \mapsto \bigvee_{i \le g(n)} x_i$. Let p be arbitrary. The on-set $f_n^{-1}(1)$ cannot be exactly covered by fewer than $g(n)$ $\mathbb{F}_p$-affine hyperplanes.*

Proof. This is essentially a restatement and specialization of [AF93, Thm. 4]. □

Example 3.32. Setting $g(n) = 0$ in the above yields the "projection" family $f_n : (x_0, \ldots, x_{n-1}) \mapsto x_0$. For each $n \in \mathbb{N}$, f_n's off-sets and on-sets are each coverable by a single hyperplane. This shows that the covering size m can be constant, even when $\{f_n\}_{n \in \mathbb{N}}$ is required to contain infinitely many nonconstant maps.

Example 3.33. Setting $g(n) = n-1$ yields $f_n : \{0,1\}^n \to \{0,1\}$ by $f_n : (x_0, \ldots, x_{n-1}) \mapsto \bigvee_{i=0}^{n-1} x_i$ (i.e., $f_n(\mathbf{x}) = 1$ unless $\mathbf{x}$ is the origin). Let $p > n$. By Lemma 3.10, $f^{-1}(0)$ can be covered by a single hyperplane; Lemma 3.30 additionally shows that $f^{-1}(1)$ can be covered by n disjoint $\mathbb{F}_p$-affine hyperplanes. Finally, Theorem 3.31 shows that no fewer than n hyperplanes can cover $f^{-1}(1)$ (this certainly remains true if one moreover restricts to disjoint collections), so that this configuration is *optimal.* We now argue that the hyperplanes $\{H_i\}_{i=0}^{n-1}$ covering $f^{-1}(1)$ can be evaluated in $O(n)$ *total* time, using a clever expression-sharing scheme. Indeed, combining Lemmas 3.10 and 3.30 above, we see that hyperplanes covering $f^{-1}(1)$ can be written explicitly in the form $H_i(x_0, \ldots, x_{n-1}) := 1 + \sum_{j<i} x_j - x_i$, for $i \in \{0, \ldots, n-1\}$. It suffices to argue that, for each $i > 0$, $H_i(\mathbf{x})$ can be evaluated in constant additional time, given $H_{i-1}(\mathbf{x})$. Indeed, it suffices to add the quantity $2 \cdot x_{i-1} - x_i$; adding this quantity clearly takes constant additional time.

We record the following corollary of Example 3.33, which essentially re-expresses [AF93, Thm. 4]:

Corollary 3.34. *Requiring that $m < n$ in Definition 3.3 would strictly shrink the resulting complexity class.*

In fact, the above argument, together with Theorem 3.31, essentially yields an infinite "hierarchy" of complexity classes—with strict inclusions—obtained by further restricting m in Definition 3.3.

3.4 Relation to the work of Ishai and Kushilevitz

In this subsection, we show that hyperplane coverings in the sense of Definition 3.1 can be used to construct randomized polynomial encodings, in the sense of Ishai and Kushilveitz [IK00, IK02]. We also survey general applications in secure computation. We show, roughly, that our setting generalizes [IK00, §2.1] by allowing *non-uniform* randomness, and that this allows us to attain a number of efficiency improvements.

We first record here a slight variant of [IK00, Def. 1], which we moreover specialize to the "perfect" case:

Definition 3.35 (Ishai–Kushilevitz). Given a function $f : \{0,1\}^n \to \{0,1\}$, a parameterized family of random variables $P(\mathbf{x})$ each with values in $\mathbb{F}_p^m$ is said to *randomize* f if the following two conditions hold:

- **Perfect privacy.** There exists a polynomial-time simulator $\mathcal{S}$ such that, for each $\mathbf{x} \in \{0,1\}^n$, the distributions $\mathcal{S}(f(\mathbf{x}))$ and $P(\mathbf{x})$ are identical.
- **Perfect correctness.** There exists a polynomial-time reconstruction algorithm X such that, for each $\mathbf{x} \in \{0,1\}^n$, $\Pr[X(P(\mathbf{x})) = f(\mathbf{x})] = 1$.

We note that we do not require here that $P(\mathbf{x})$ be given as a fixed vector of polynomials with additional *uniform* random inputs (cf. [IK00, §2.1]).

We now fix a function $f : \{0,1\}^n \to \{0,1\}$, a prime p, and a family of hyperplanes $\{H_i\}_{i=0}^{m-1}$ disjointly covering $f^{-1}(1)$ (in the sense of Definition 3.1), which we view as affine functionals. We take the additional step of sampling m random, nonzero scalars $(\alpha_i)_{i=0}^{m-1}$ in $\mathbb{F}_p^*$, as well as a random *permutation* $\rho \leftarrow \mathbf{S}_m$. (This idea is already implicit in [WGC19, Alg. 3].) Actually, it is enough that ρ be sampled uniformly from the subgroup $\langle(0, 1, \ldots, m-1)\rangle \subset \mathbf{S}_m$ consisting of *circular shift* permutations (this order-m subgroup is much smaller than $\mathbf{S}_m$, and requires fewer random coins to sample from). We now consider, for each input $\mathbf{x} = (x_0, \ldots, x_{n-1}) \in \{0,1\}^n$, the *distribution*—over random choice of $(\alpha_i)_{i=0}^{m-1}$ and ρ—of the vector $P(\mathbf{x}) := \left(\alpha_i \cdot H_{\rho(i)}(\mathbf{x})\right)_{i=0}^{m-1}$. That is, we evaluate $H_i(\mathbf{x})$ for each $i \in \{0, \ldots, m-1\}$, permute the resulting evaluations using ρ, and then multiply each permuted output by a random nonzero scalar (these latter two steps can also be reversed).

Remarkably, the resulting construction perfectly randomizes f:

Theorem 3.36. *The construction $P(\mathbf{x}) := \left(\alpha_i \cdot H_{\rho(i)}(\mathbf{x})\right)_{i=0}^{m-1}$ randomly encodes f as in Definition 3.35.*

Proof. We describe a simulator $\mathcal{S}$ satisfying the criterion of Definition 3.35. $\mathcal{S}$ first samples m nonzero scalars $(y_i)_{i=0}^{m-1}$ in $\mathbb{F}_p^*$. If and only if its input $v = 1$, $\mathcal{S}$ samples a random index $i^* \in \{0, \ldots, m-1\}$ and overwrites $y_{i^*} := 0$. We now argue that this simulator generates the correct distribution. Directly by Definition 3.1, we see that, for each $\mathbf{x} \in \{0,1\}^n$, $f(\mathbf{x}) = 1$ if and only if $H_i(\mathbf{x}) = 0$ for one and only one index $i \in \{0, \ldots, m-1\}$; otherwise, $H_i(\mathbf{x}) \neq 0$ for each $i \in \{0, \ldots, m-1\}$. It follows by its construction that $P(\mathbf{x})$ exactly matches $\mathcal{S}(0)$ or $\mathcal{S}(1)$, accordingly as $f(\mathbf{x})$ equals 0 or 1.

The reconstructor X may, on any sample $\mathbf{y} = (y_0, \ldots, y_{m-1}) \leftarrow P(\mathbf{x})$, return 1 if and only if one of the components $y_i = 0$. □

In order to talk about the *complexity* of $P(\mathbf{x})$, we need to express it as a polynomial. We begin with the following definition:

Definition 3.37. A *randomizing polynomial* of f is a randomized encoding $P(\mathbf{x})$ in the sense of Definition 3.35 which, moreover, is expressed as a vector of m polynomial functions of $\mathbf{x} \in \{0,1\}^n$ and additional random inputs. In addition:

- P's *degree* is its degree as a polynomial, in its combined boolean and random inputs.
- P's *output complexity* is the dimension of the space in which the outputs $P(\mathbf{x})$ are distributed.
- P's *randomness complexity* is the dimension of the space from which its random inputs are drawn.
- P's *evaluation complexity* is the size of the arithmetic circuit representing P.

Remark 3.38. Our notions of "degree" and "output complexity" match those of [IK00, §2.1]. Our notion of "randomness complexity" specializes to that of [IK00, §2.1] in case P's random inputs are independent uniform scalars, but does not require that they be so. Finally, our notion of "evaluation complexity" appears new. Indeed, the works [IK00, IK02] do not consider the complexity of *evaluating* $P(\mathbf{x})$, but rather just its numbers of random input and output components.

Theorem 3.36 can be expressed as a randomizing polynomial of low degree:

Lemma 3.39. *The construction $P(\mathbf{x})$ of Theorem 3.36 can be expressed as a randomizing polynomial of degree 2, output complexity m, randomness complexity m^2, and evaluation complexity $O(m^2)$.*

Proof. We prove the lemma by an explicit construction. We argue that $P(\mathbf{x})$ can be attained by multiplying the initial vector $(H_i(\mathbf{x}))_{i=0}^{m-1}$ of deterministic linear polynomials by a certain random $m \times m$ *matrix* $R \in \mathbb{F}_p^{m \times m}$, whose distribution we presently describe. Indeed, R can be constructed by first sampling a uniform "offset" index $r^* \in \{0, \ldots, m-1\}$, and then setting, for each i and j in $\{0, \ldots, m-1\}$, $r_{i,j} \leftarrow \mathbb{F}_p^*$ if and only if $j - i \equiv r^* \pmod{m}$ and $r_{i,j} := 0$ otherwise. It is clear that $P(\mathbf{x}) = R \cdot [H_i(\mathbf{x})]_{i=0}^{m-1}$. This implies the statement of the lemma. □

Remark 3.40. Lemma 3.39 appears to contradict the impossibility result [IK00, Cor. 5.9], which states that only certain *trivial* functions can be evaluated by degree-2 randomizing polynomials. We explain this apparent contradiction. The result [IK00, Cor. 5.9] relies crucially on $P(\mathbf{x})$'s random scalars being uniform and independent, as [IK00, Def. 2.1] requires; on the other hand, our random matrix R above is *not* uniform over $\mathbb{F}_p^{m \times m}$, and nor are its components independent. Our construction thus does not satisfy the hypothesis of [IK00, Cor. 5.9]. In fact, we now point out explicitly where the proof of [IK00, Cor. 5.9] fails in our more general setting. Following the proof of [IK00, Lem. 5.5], we obtain a *single* "weakly randomizing" polynomial

$$p(\mathbf{x}) = r_0 \cdot H_0(\mathbf{x}) + \cdots + r_{m-1} \cdot H_{m-1}(\mathbf{x}),$$

whose distribution depends only on $f(\mathbf{x})$. In our setting, however, the scalars $(r_i)_{i=0}^{m-1}$ are no longer uniform and independent. For example, choosing a weighting scheme as in [IK00, Fact 5.2] with exactly one nonzero weight $w_{i^*} = 1$, we obtain a $p(\mathbf{x})$ as above for which a *random* r_i is nonzero and uniform, whereas the rest are zero. It is now simply false that any $\mathbf{x} \in \{0,1\}^n$ for which $H_i(\mathbf{x}) \neq 0$ for some i (in fact, all $\mathbf{x} \in \{0,1\}^n$ satisfy this property) induces a uniform distribution on $\mathbb{F}_p$. Rather, the distribution of $p(\mathbf{x})$ depends exactly on whether $H_i(\mathbf{x}) \neq 0$ for *all* $i \in \{0, \ldots, m-1\}$ or not; in these respective cases, $p(\mathbf{x})$ is either uniform on $\mathbb{F}_p^*$ or else is uniform on $\mathbb{F}_p^*$ with probability $\frac{m-1}{m}$ and 0 with probability $\frac{1}{m}$. In particular, case 1. of [IK00, Lem. 5.5]'s assertion that any $p(\mathbf{x})$ for which each $\mathbf{x} \in \{0,1\}^n$ satisfies $H_i(\mathbf{x}) \neq 0$ for some i tests a linear condition is false (those $\mathbf{x}$ satisfying $H_i(\mathbf{x}) \neq 0$ for some i may nonetheless represent distinct distributions).

We now observe that the randomness and evaluation complexity of Theorem 3.36 can be *further* reduced, at the cost of an increase in degree. We refer Nussbaumer [Nus82] for preliminaries on Fourier-theoretic techniques. We have the notions of *circular convolution* (see Nussbaumer [Nus82, (8.1)]) and the *number-theoretic transform* (see [Nus82, (8.2)]). We assume that $m \mid (p-1)$, so that a primitive m^{th} root of unity—say, g—exists modulo p, and the number-theoretic transform makes sense (see [Nus82, Thm. 8.2]). We have the convolution theorem [Nus82, (4.150) and Thm. 8.1], which shows that number-theoretic transforms can be used to evaluate convolutions. We finally recall the radix-2 decimation in time Cooley–Tukey algorithm [Nus82, §4.2.1]; we assume moreover assume that m is a power of 2, so that this algorithm can be applied.

Lemma 3.41. *The construction $P(\mathbf{x})$ of Theorem 3.36 can be expressed as a randomizing polynomial of degree 3, output complexity m, randomness complexity $2 \cdot m$, and evaluation complexity $O(m \cdot \log m)$.*

Proof. This result relies on the following representation of the construction $P(\mathbf{x})$ above:

$$P(\mathbf{x}) = \left[\ \alpha_i\ \right]_{i=0}^{m-1} \cdot \left[\ H_{i+r^*}(\mathbf{x})\ \right]_{i=0}^{m-1},$$

where each $\alpha_i \in \mathbb{F}_p^*$ is random, and $r^* \in \{0, \ldots, m-1\}$ is a *random* offset (and all indices are interpreted modulo m). The key idea is that though circular shifting requires a matrix multiplication in the time domain, it collapses to a *componentwise* multiplication operation in the frequency domain; this is precisely the "time shifting" property of the number-theoretic transform [Nus82, (4.11)]. That is, we have the representation:

$$P(\mathbf{x}) = \left[\ \alpha_i\ \right]_{i=0}^{m-1} \odot \mathrm{NTT}^{-1}\left(\left[\ g^{i \cdot r^*}\ \right]_{i=0}^{m-1} \odot \mathrm{NTT}\left(\left[\ H_i(\mathbf{x})\ \right]_{i=0}^{m-1}\right)\right),$$

where $\odot$ represents the element-wise "Hadamard product" of length-m vectors, and $r^* \leftarrow \{0, \ldots, m-1\}$ is again random. In other words, $P(\mathbf{x})$ can be calculated by first applying the number-theoretic transform to $\left[\ H_i(\mathbf{x})\ \right]_{i=0}^{m-1}$, multiplying the result element-wise by a *random column* $\left[\ g^{i \cdot r^*}\ \right]_{i=0}^{m-1}$ of the NTT matrix [Nus82, (5.8.3)], taking the inverse number-theoretic transform of the resulting quantity, and finally re-randomizing the output component-wise.

The resulting construction is clearly polynomial. Because the NTT and inverse NTT are linear maps (with constant coefficients), this construction is of degree 3 in its combined secret and random inputs. Its random inputs consist of the random vector $\left[\ \alpha_i\ \right]_{i=0}^{m-1}$ (independent and uniform over $\mathbb{F}_p^*$) and the random vector $\left[\ g^{i \cdot r^*}\ \right]_{i=0}^{m-1}$ (a random column of the NTT matrix); these comprise $2 \cdot m$ total components. Its output complexity is clearly m.

It remains to argue that the NTT and inverse NTT can be evaluated in $O(m \cdot \log m)$ time. This is essentially the classic Cooley–Tukey algorithm [Nus82, §4.2.1]. To make this point particularly clear, we express the number-theoretic transform explicitly as a matrix factorization. Indeed, it is implicit in the description [Nus82, §4.3.2] that the NTT matrix can be factored as follows:

$$\mathrm{NTT} = \left[\ g^{i \cdot j}\ \right]_{i,j=0}^{m-1} = \underbrace{\left[\ A_{\log m}\ \right] \cdot \left[\ A_{\log m - 1}\ \right] \cdot \cdots \cdot \left[\ A_1\ \right]}_{\log m \text{ matrices}} \cdot \left[\ B\ \right],$$

where B is a fixed $m \times m$ *bit reversal permutation* matrix, and the matrices A_k, $k \in \{1, \ldots, \log m\}$ encode certain "butterfly operations" (details are given in [Nus82, §4.3.2]). The inverse NTT is identical, except that an extra multiplicative factor of $m^{-1} \pmod{p}$ is applied.

Each of the matrices A_k, $k \in \{1, \ldots, \log m\}$, and B has only $O(m)$ nonzero components, and can be evaluated in $O(m)$ total time. Each Hadamard product takes only $O(m)$ time to evaluate. The total complexity is thus $O(m \cdot \log m)$. This completes the proof. □

We are not currently able to obtain a representation of $P(\mathbf{x})$ as in Theorem 3.36 which is simultaneously of degree 2 and of evaluation complexity $O(m \cdot \log m)$, though it may be possible.

Question 3.42. *Can $P(\mathbf{x})$ be expressed as a randomizing polynomial of degree 2 and of evaluation complexity $O(m \cdot \log m)$?*

We now compare the efficiency of our constructions to those of [IK00, Thm. 3.5] and [IK02, §3]. We emphasize first that the underlying computational models are distinct; in particular, our constructions are measured as functions of f's *hyperplane complexity* m, whereas those of [IK00, Thm. 3.5] and [IK02, §3] are measured in terms of f's nondeterministic mod-p branching program complexity l and f's boolean formula complexity s, respectively. The nondeterministic mod-p branching program complexity—defined in [IK00, Def. 2.5]—is a somewhat atypical metric of computation, which is challenging to assess on concrete functions f ([IK00] gives no examples). We suspect that $m(n) \in O(l(n))$ for many or most functions f of interest. Finally, we argue in fact that $m(n) \in o(s(n))$ for many natural functions f (s here refers to boolean formula complexity). An explicit example is given in Example 3.27 above. This already exhibits an explicit asymptotic separation between $m(n)$ and $s(n)$, in at least one particular natural function of interest.

We demonstrate further asymptotic improvements, expressed in Table 1 below. Some of our improvements depend fundamentally on our use of non-uniform randomness, as is made clear by Remark 3.40 above. However, our improvements don't come "for free" merely from our use of non-uniformity alone. Indeed, we argue that our constructions improve upon those of [IK00, Thm. 3.5] and [IK02, Thm. 1] *even* when nonuniform randomness is permitted in the latter protocols. In fact, [IK02, §4.1] remarks explicitly that an improved variant of [IK00, Thm. 3.5] could be attained were nonuniform randomness allowed; in this hypothetical variant of [IK00, Thm. 3.5], the randomizing polynomial samples uniformly random *nonsingular* matrices $R \in \mathbb{F}_p^{l \times l}$ (a random such matrix cannot be sampled by uniformly sampling *individual* components, as [IK02, §4.1] observes). We call this hypothetical variant "[IK00, Thm. 3.5] With Nonuniformity", and argue that our constructions beat it as well.

We note finally the notion of "distinguishing error". While our reconstructor X succeeds with probability 1, that of [IK00, Thm. 3.5] succeeds with probability only 0.08 (see [IK00, Lem. 3.4]) and so would need to be repeated many times in order to achieve higher distinguishing probabilities. For example, even to achieve a distinguishing probability of $\frac{1}{2}$, at least $\log_{1-0.08}(\frac{1}{2}) \approx 8.31$ (i.e., 9) copies of [IK00, Thm. 3.5]'s construction would have to be concatenated; 56 copies would be required to achieved 0.01-correctness. Though these factors are constant in n, they would likely be significant in practice. Because all constructions considered are perfectly private, we do not include a column for privacy error.

Table 1: Efficiency of our constructions and those of Ishai and Kushilevitz.

Construction	Degree	Out. Comp.	Rand. Comp.	Eval. Comp.	Dist. Err.
[IK00, Thm. 3.5]	3	$\Theta(l^2)$	$\Theta(l^2)$	$\Theta(l^3)$	0.92
[IK00, Thm. 3.5] W/ NonU.	3	$\Theta(l^2)$	$\Theta(l^2)$	$\Theta(l^3)$	0
[IK02, §3]	3	$\Theta(s^{\log_2 3})$	$\Theta(s^{\log_2 3})$	$\Theta(s^{\log_2 3})$	0
[ABT18]	2	$\mathsf{poly}(s)$	$\mathsf{poly}(s)$	$\mathsf{poly}(s)$	0
Lemma 3.39	2	m	m^2	$\Theta(m^2)$	0
Lemma 3.41	3	m	$2 \cdot m$	$\Theta(m \cdot \log m)$	0

We now discuss the applicability of our constructions to secure computation. We note that the generic application [IK00, Thm. 4.1] to secure computation (cf. also [ABT18, Prop. 1]) cannot go through as written in our more general setting, because it relies on the parties' ability to independently generate samples from a uniform distribution. Rather, in order for something like [IK00, Thm. 4.1] to hold in our setting, the parties would need *correlated* randomness. The role of correlated randomness in secure computation is discussed extensively in Ishai, Kushilevitz, Meldgaard, Orlandi, and Paskin-Cherniavsky [IKM$^+$13]. Our constructions can thus be viewed as evidence of the power of correlated randomness.

We refer to [IKM$^+$13, §2] for definitions pertaining to the correlated-randomness model, also called the *preprocessing model.* Applying Lemma 3.39, we obtain the following analogue of [IK00, Cor. 4.4] and [ABT18, Thm. 1]:

Corollary 3.43. *Suppose that $f : \{0,1\}^n \to \{0,1\}$ can be computed by $m(n)$ hyperplanes. Then in the preprocessing model, f can be passively-securely computed by k parties with perfect correctness and perfect $\lfloor \frac{k-1}{2} \rfloor$-privacy in two rounds, using $O(k \cdot m^2)$ correlated random bits and communication $O(k \cdot (n + k \cdot m))$.*

Proof (sketch). The proof is a close variant of [IK00, Lem. 4.3], in which as many as $\lfloor \frac{k-1}{2} \rfloor$ corrupt parties are allowed. The proof is identical, except uses the fact that $P(\mathbf{x})$ can be computed by a polynomial of degree 2. Each player may begin by secret-sharing its inputs under Shamir's scheme (see [KL21, §15.3.1]) to the k parties with security threshold $\lfloor \frac{k-1}{2} \rfloor$; we also assume that the parties are given component-wise shares of a random matrix R distributed exactly as in Lemma 3.39. We now apply [IK00, Thm. 4.1]. Each player may evaluate the construction of Lemma 3.39 locally. Because that construction is of degree 2, the resulting intermediate polynomial is of degree less than k, and can still be reconstructed. The players may use the private reconstruction procedure given in [IK00, p. 304]. $\square$

This corollary compares favorably with [IK00, Cor. 4.4], in that it improves the privacy threshold from $\lfloor \frac{n-1}{3} \rfloor$ to $\lfloor \frac{n-1}{2} \rfloor$, and yet still uses only 2 rounds; it's also comparable to the main result [ABT18, Thm. 1]. We stress on that Corollary 3.43 requires the use of *correlated* randomness.

Corollary 3.43 also appears to compare favorably with certain results of [IKM$^+$13], which explicitly studies the preprocessing model. For example, we refer to [IKM$^+$13, Thm. 1], which requires *exponentially many* correlated random bits to achieve an analogous outcome (see also [IKM$^+$13, Table 1]). We note that Corollary 3.43 does not contradict the impossibility result [IKM$^+$13, Thm. 14]. For one, f is not a "sender-receiver" functionality; in addition, Corollary 3.43 requires polynomially many random bits *only* when m is polynomial in n; this does *not* hold for all functions f, as would be required to meet the hypothesis of [IKM$^+$13, Thm. 14].

4 The Hyperplane Synthesis Algorithm

In this section, we give our main *constructive* algorithm, which generates, given an arbitrary subset S of the cube (expressed as a list of minterms), a disjoint hyperplane covering of S. Roughly speaking:

Theorem 4.1. *Fix a natural number n, and suppose that $p \geq 2^n$. Given a subset $S \subset \{0,1\}^n$, Algorithm 4 below correctly outputs a family of affine hyperplanes $H_0, \ldots, H_{m-1}$ in $\mathbb{F}_p^n$ for which $S = \bigsqcup_{i=0}^{m-1} H_i \cap \{0,1\}^n$.*

That is, the respective intersections of the hyperplanes $H_0, \ldots, H_{m-1}$ with $\{0,1\}^n$ disjointly cover S, and cover no further elements of the cube. In fact, our algorithm attempts to find a such configuration for which m is minimal, and moreover to do so efficiently.

Because the size of the input—that is, a minterm-representation of S—can be as large as 2^n, it is impossible *a priori* to achieve an algorithm whose runtime is polynomial in n. We therefore, following the tradition of (for example) Espresso II [BHMSV84], seek to construct algorithms which are *heuristically* efficient, and which yield good results in practice. Likewise, it seems essentially impossible to construct *necessarily* minimal configurations $H_0, \ldots, H_{m-1}$ with any reasonable efficiency; thus, the configurations our algorithm outputs are only heuristically minimal (though they often wind up exactly minimal in practice). We give concrete benchmarks—exhibiting both runtime and output quality—below, in Subsection 4.5.

Our primary strategy involves "recursive backtracking". Our algorithm builds one flat at a time; it constructs each individual such flat dimension-by-dimension. Given a "flat in progress" and a new candidate basis vector, our algorithm computes the affine span of the extended basis, and checks whether this span remains contained in S (and also disjoint from all flats previously constructed). For each such partially constructed flat, a handful of linearly independent such extension vectors are considered. In this way, a tree structure—each path through which corresponds to some particular sequence of extensions—is built; the leaf which covers the most points is retained. This entire search process is then repeated iteratively; in each iteration, additional flats are added to each configuration. Iteration proceeds until some configuration manages to exhaust all of S. We describe our algorithm throughout a handful of subsections.

4.1 Computing a span

We begin with a concrete and efficient method by which to evaluate the affine span of $k+1$ cube elements, and in particular that portion of their span which intersects $\{0,1\}^n$. Lemma 2.4 above shows that, in order to compute $K \cap \{0,1\}^n$ (where K is represented by a row-reduced matrix U), one must only check those combinations $\sum_{i=0}^{k-1} \alpha_i \cdot \mathbf{x}_i$ of U's rows for which each $\alpha_i \in \{0,1\}$. In fact, we can further improve the efficiency of this computation using the Gray code (see e.g. Knuth [Knu11, §7.2.1.1]), as we now demonstrate. In what follows, we express K as a $k \times n$ matrix U, which we assume moreover is row-reduced over $\mathbb{F}_p$.

Algorithm 1 ComputeSpan

```
Require: A row-reduced $k \times n$ matrix $U$ over $\mathbb{F}_p$.
Ensure: The intersection $C$ of $U$'s row-space with $\{0,1\}^n$.
1: $C := \varnothing$
2: Initialize $\mathbf{x} := \mathbf{0}$.
3: for $(\alpha_0, \ldots, \alpha_{k-1}) = \boldsymbol{\alpha} \in \{0,1\}^k$ in Gray-code order: do
4:     Incrementally adjust $\mathbf{x}$ so that $\mathbf{x} = \sum_{i=0}^{k-1} \alpha_i \cdot \mathbf{x}_i$.        ▷ $\mathbf{x}_i$ here refers to $U$'s $i^{\text{th}}$ row.
5:     if $\mathbf{x} \in \{0,1\}^n$ then $C \cup= \{\mathbf{x}\}$
6: return $C$
```

Lemma 4.2. *Algorithm 1 terminates in $O(n \cdot 2^k)$ time.*

Proof. Because the coefficient combinations $\boldsymbol{\alpha} = (\alpha_0, \ldots, \alpha_{k-1}) \in \{0,1\}^k$ are considered in Gray code order, upon each successive iteration, exactly one length-n basis vector must be added or subtracted from the "running combination" $\mathbf{x}$ in order to determine the appropriate sum $\mathbf{x} = \sum_{i=0}^{k-1} \alpha_i \cdot \mathbf{x}_i$. □

The analogue of Algorithm 1 in which the Gray code were not used would take $O(k \cdot n \cdot 2^k)$ time. In any case, both methods significantly beat the naïve approach whereby K^T is row-reduced and—for each $\mathbf{x} \in \{0,1\}^n$—the augmented system $\left[K^T \mid x\right]$ is checked for consistency; this would take $\Omega(2^n)$ time.

4.2 Fundamental datastructures

Slightly abusing notation, we write $\{0,1\}^n$ for the *datatype* consisting of cube elements. (Practically, we represent these as n-bit integers, using a little-endian conversion; in practice, we have $n \leq 64$.) We also write $\mathbb{F}_p^n$ for the datatype consisting of n-tuples of $\mathbb{F}_p$-elements. Through the natural inclusion, $\{0,1\}^n \subset \mathbb{F}_p^n$.

To avoid performing duplicate work, we use a transposition table construction in our tree search, together with the Zobrist hashing method [Zob70] (typically used in computer game-playing programs). To each element $\mathbf{x} \in \{0,1\}^n$, we associate a random 64-bit key $Z[\mathbf{x}]$. In fact, two *distinct* configurations may nonetheless have the same union; to prevent unwarranted table false-positives in such settings, we actually refresh the entire Zobrist table Z_i between each iteration i of the main outer loop (see Algorithm 2 below).

Thus, to a configuration C, we associate the Zobrist key:

$$C.z := \bigoplus_{K_i \in C.F} \left(\bigoplus_{\mathbf{x} \in K_i \cap \{0,1\}^n} Z_i[\mathbf{x}] \right).$$

Because the flats we consider in this subsection's algorithm are *generated* by cube elements, they are characterized uniquely by their respective intersections with the cube, and this approach is correct.

We now describe our an "affine flat" datastructure, which encodes a partially constructed flat, and also supports "extension" algorithms.

```
class FLAT
    ${0,1}^n$ o                                  ▷ The origin point of the flat $K$ this object represents.
    SET $\langle\{0,1\}^n\rangle$ $C$                        ▷ Tracks the intersection $K \cap \{0,1\}^n$.
    VECTOR $\langle\mathbb{F}_p^n\rangle$ $U$                ▷ A row-reduced, $k \times n$ matrix generating $K$ with respect to $\mathbf{o}$.
    INT $z$                                      ▷ The Zobrist hash key of this configuration.
    function FLAT fresh($\{0,1\}^n$ $\mathbf{x}$)               ▷ Creates a fresh 0-flat containing just $\mathbf{x}$.
        return a fresh FLAT $K$ for which $K.\mathbf{o} := \mathbf{x}$, $K.S := \{\mathbf{x}\}$, $K.U := []$, and $K.z = Z_i[\mathbf{x}]$.
    function VOID extend($\{0,1\}^n$ $\mathbf{x}$)              ▷ Extends self using the new point $\mathbf{x}$.
        $U$ += $\mathbf{o} \oplus \mathbf{x}$                        ▷ Append the "relativized" point $\mathbf{o} \oplus \mathbf{x}$ to the flat's basis.
        Row-reduce $U$                           ▷ By induction, $U$'s first $k-1$ rows are *already* row-reduced.
        $C := \{\mathbf{o} \oplus \mathbf{x} \mid \mathbf{x} \in \mathsf{ComputeSpan}(U)\}$   ▷ Stash the "re-relativized" output of Algorithm 1 above.
        $z := \bigoplus_{\mathbf{x} \in C} Z[\mathbf{x}]$   ▷ Update this flat's Zobrist key. In practice, this can be done "incrementally".
end class
```

Throughout our algorithm, we recursively explore various configurations of disjoint affine flats. To keep track of the data associated with each such configuration, we define the following datastructure:

```
class CONFIGURATION
    SET $\langle\{0,1\}^n\rangle$ $R$                ▷ Set tracking remaining points *uncovered* by the configuration.
    VECTOR $\langle$FLAT$\rangle$ $F$                    ▷ A list of the flats constituting this configuration.
    SET $\langle$CONFIGURATION$\rangle$ $H$          ▷ A list of "child" configurations, each featuring an extension.
    INT $z$                                  ▷ The Zobrist hash key of this configuration.
    function VOID extend(INT $i$, $\{0,1\}^n$ $\mathbf{x}$)          ▷ Extends self's $i^{\text{th}}$ flat, using $\mathbf{x}$.
        Copy-initialize a new CONFIGURATION $c$ from self with $c.H := []$
        if $F$.length $= i$ then $c.F$ += fresh($\mathbf{x}$)     ▷ This is the configuration's first time in the $i^{\text{th}}$ cycle.
        else $c.F$.back().extend($\mathbf{x}$)               ▷ The $i^{\text{th}}$ flat already exists; extend it.
        if $c.F$.back()$.C \not\subseteq R$ then return           ▷ New flat "leaked" outside $R$; abort early.
        $c.R$ −= $c.F$.back()$.C$                ▷ Remove newly covered points from set of remaining points.
        $c.z := \bigoplus_{K \in c.F} K.z$   ▷ Update this configuration's Zobrist key (done incrementally, in practice).
        $H$ += $c$                 ▷ New child was successful; append it to this configuration's list of children.
end class
```

In order to facilitate the discovery of good solutions which nonetheless appear suboptimal initially, we use a "leaderboard" construction, which caches a fixed number—say, l—of high-performing configurations in between iterations of the main search routine. More specifically, we define an ORDEREDSET $\langle$CONFIGURATION, $l\rangle$ datastructure L, which maintains at most l CONFIGURATION handles C (in practice we use pointers), sorted in ascending order by the number of remaining uncovered points $C.R$.size(). When an element is added to L—and if L.size() $\geq l$—the new element is compared to L's worst element. If the new element beats the leaderboard's worst element, then the new element is inserted and the worst one is expunged; otherwise, no action is taken.

In our algorithm, we repeatedly explore a *tree* of configurations, each time modifying only each configuration's i^{th} flat throughout. When each round of exploration concludes (and in fact, in real time, as we discuss below), we prune from the tree each element none of whose descendants are on the leaderboard. We then increment i, clear the leaderboard, and perform a new recursive search. This process is repeated until *some* configuration in the tree manages to exhaust all the initial input subset S. This paradigm has the effect of retaining as leaves exactly those configurations whose first $i-1$ flats "have performed well", and which stand to be propelled into the lead upon the conclusion of the i^{th} search cycle.

The leaderboard construction makes our algorithm somewhat more complicated to describe. Indeed, it makes the CONFIGURATION datastructure necessary; if the leaderboard were not used, then flats could be considered in isolation, and the state refreshed after each tree search. (Nonetheless, setting $l = 1$ has the effect of collapsing our algorithm back to this behavior.) In practice, the leaderboard construction significantly improves our algorithm's output quality, and imposes only a modest performance burden.

4.3 The flat-finding algorithm

We now explicitly describe our flat synthesis algorithm. We fix an integral *branching factor*, say b. We emphasize that the flats output by Algorithm 2 below have *arbitrary* dimensions $k \in \{0, \ldots, n-1\}$, and are not necessarily hyperplanes. We provide an algorithm for flat extension in the next subsection.

Algorithm 2 FlatFinder(SET $\langle\{0,1\}^n\rangle$ S)

```
 1: Initialize MAP ⟨$\{0,1\}^n \to$ INT⟩ $Z$   ▷ Global random mapping between cube elements and Zobrist keys.
 2: Initialize SET ⟨INT⟩ $T$   ▷ Transposition table, whose elements the keys of visited configurations.
 3: Initialize CONFIGURATION $C$ with $C.R := S$   ▷ $C$ here is the root configuration.
 4: Initialize ORDEREDSET ⟨CONFIGURATION, $l$⟩ $L$   ▷ Leaderboard, containing $l$ configurations.
 5: Initialize BOOL $s :=$ false   ▷ The variable $s$ tracks whether an exhausting configuration has been found.
 6: function VOID RecursiveSearch(INT $i$, CONFIGURATION $C$)
 7:     if $C.z \in T$ then return   ▷ Configuration's Zobrist key is present in the transposition table; abort.
 8:     if $C.H = \varnothing$ then   ▷ Node is a leaf; begin process of randomly generating children.
 9:         $X :=$ Sample$(C.R, b)$   ▷ Sample $b$ distinct extension candidates from the *remaining* set $C.R$.
10:         for $\{0,1\}^n$ $\mathbf{x} \in X$ do $C$.extend$(i, \mathbf{x})$   ▷ Populate $C$'s children by attempting extensions.
11:     for $c$ in $C.H$ do
12:         if not $s$ then RecursiveSearch$(i, c)$   ▷ Terminate early if $s$ becomes true.
13:     if $C.H = \varnothing$ then   ▷ This branch will execute only if all attempted extensions in line 10 failed.
14:         $L \cup= C$   ▷ As this node is a leaf, we consider it for inclusion in the leaderboard.
15:         if $C.R = \varnothing$ then $s :=$ true   ▷ Check whether our search is done; if so, mark $s$ for termination.
16:     $T \cup= C.z$   ▷ Add $C$'s Zobrist key to the hashtable.
17: for INT $i \in \{0, 1, 2, \ldots\}$ do
18:     Freshly re-assign $Z$ to a random mapping MAP ⟨$\{0,1\}^n \to$ INT⟩.
19:     RecursiveSearch$(i, C)$
20:     Prune from the tree $C$ each CONFIGURATION whose subtree contains no leaderboard elements.
21:     Clear the transposition table $T$.
22:     if $s =$ true then break
23:     Clear the leaderboard $L$.
24: Output CONFIGURATION $L$.best().
```

Remark 4.3. To minimize our algorithm's memory footprint, we actually prune from the tree *in real time* configurations none of whose descendants reside in the leaderboard. Because the "bookkeeping" involved in this process is fairly intricate, we suppress it from our account of our algorithm.

Remark 4.4. In practice, we employ an additional optimization whereby a set Y of "bad" points—that is, of cube-elements $\mathbf{x}$ which cause line 10 above to fail—is retained as a class member in each CONFIGURATION C and is inherited by C's children. The idea is that a point $\mathbf{x}$ which causes C to "leak" outside of R will necessarily have the same effect upon each of C's descendants, and can be excluded from consideration when subsequent candidates $\mathbf{x}$ for extension are considered; i.e., line 9 can instead execute $X :=$ Sample$(C.R - Y, b)$.

Remark 4.5. The aspect of our algorithm whereby an "already-incorporated" set C, a "plausible" set R, and a "problematic" set Y are simultaneously retained and recursively modified makes it analogous in certain respects to the Bron–Kerbosch max-clique-finding algorithm (see e.g. Cazals and Karande [CK08, 2.1]).

4.4 The flat extension algorithm

In this section, we give an algorithmic procedure which extends flats into hyperplanes. This extension capability is important in practice, as Algorithm 2 above generally delivers k-flats K for which $k < n - 1$.

Our algorithm proceeds in the following way; we assume for now that $p \geq 2^n$. Assuming by Lemma 2.1 that K contains the origin, the quotient map $\mathbb{F}_p^n \to \mathbb{F}_p^n / K$—representable concretely by a matrix $A : \mathbb{F}_p^n \to \mathbb{F}_p^{n-k}$ in the sense of Lemma 2.3—necessarily sends each element of $\{0,1\}^n - K$ to a nonzero element. To define a hyperplane $H : \mathbb{F}_p^n \to \mathbb{F}_p$ satisfying the hypothesis of the question, it suffices to construct a chain of one-dimensional quotients

$$H : \mathbb{F}_p^n \xrightarrow{A} \mathbb{F}_p^{n-k} \xrightarrow{A_{n-k-1}} \mathbb{F}_p^{n-k-1} \xrightarrow{A_{n-k-2}} \cdots \xrightarrow{A_2} \mathbb{F}_p^2 \xrightarrow{A_1} \mathbb{F}_p,$$

where, for each $l \in \{1, \ldots, n-k-1\}$, the map A_{n-k-l} sends each element of the image $(A_{n-k-l-1} \circ \cdots \circ A_{n-k-1} \circ A)(\{0,1\}^n - K)$ to a nonzero element. By induction, this in turn amount to selecting a line in each space $\mathbb{F}_p^{n-k-l+1}$—or equivalently, an element of $\mathbb{P}\mathbb{F}_p^{n-k-l}$—which avoids this image. The image has at most $2^n - 1$ elements. On the other hand, $\left|\mathbb{P}\mathbb{F}_p^{n-k-l}\right| = \sum_{j=0}^{n-k-l} p^j$. By hypothesis on p, this latter quantity strictly exceeds $2^n - 1$ for each l, even in the worst case $l = n-k-1$.

We note that this approach remains correct (with minor modifications) even under certain weaker assumptions on p, as we argue below in Appendix A (see e.g. Theorems A.4 and A.17 below). As the exposition becomes significantly more complex under these weaker assumptions, and we restrict to the case $p \geq 2^n$ in this subsection.

For each projective space $\mathbb{P}\mathbb{F}_p^{n-k-l}$, we write $U_0 \subset \mathbb{P}\mathbb{F}_p^{n-k-l}$ for the *affine coordinate chart* consisting of those elements $(v_0 : \ldots : v_{n-k-l}) \in \mathbb{P}\mathbb{F}_p^{n-k-l}$ for which $v_0 \neq 0$. Each such element has a unique representation in the affine space $U_0 \cong \mathbb{F}_p^{n-k-l}$, namely $\left(\frac{v_1}{v_0}, \ldots, \frac{v_{n-k-l}}{v_0}\right)$ (see for example Hartshorne [Har77, p. 10]).

Algorithm 3 FlatExtender

```
Require: An odd prime $p$ such that $p \geq 2^n$. An $k \times n$ $\mathbb{F}_p$-matrix $U$ spanning $K$, where $k < n-2$.
Ensure: A functional $H : \mathbb{F}_p^n \to \mathbb{F}_p$ such that $K \cap \{0,1\}^n = H \cap \{0,1\}^n$.
 1: Using Lemma 2.3, construct a full-rank $(n-k) \times n$ matrix $A$ annihilating $U$.
 2: Initialize $\mathrm{im}_0 := A(\{0,1\}^n - K)$        ▷ Concretely, $\mathrm{im}_0$ is a $(n-k) \times |\{0,1\}^n - K|$ $\mathbb{F}_p$-matrix.
 3: Initialize $H_0 := A$
 4: for $l \in \{1, \ldots, n-k-1\}$ do
 5:     Express the elements of $U_0 \cap \mathrm{im}_{l-1} \subset \mathbb{P}\mathbb{F}_p^{n-k-l}$ in affine coordinates, and store them in a hashtable.
 6:     do randomly sample $(u_1, \ldots, u_{n-k-l}) \leftarrow U_0 \cong \mathbb{F}_p^{n-k-l}$ until $(u_1, \ldots, u_{n-k-l}) \notin U_0 \cap \mathrm{im}_{i-1}$
 7:     Construct a $(n-k-l) \times (n-k-l+1)$ matrix $A_{n-k-l}$ annihilating $(1, u_1, \ldots, u_{n-k-l})$
 8:     Carry forward $\mathrm{im}_l := A_{n-k-l}(\mathrm{im}_{l-1})$   ▷ Concretely, $\mathrm{im}_l$ is an $(n-k-l) \times |\{0,1\}^n - K|$ $\mathbb{F}_p$-matrix.
 9:     Set $H_l := A_{n-k-l} \cdot H_{l-1}$        ▷ $H_l$ is an $(n-k-l) \times n$ matrix.
10: return $H := H_{n-k-1}$
```

Lemma 4.6. *Algorithm 3 terminates in finite time almost surely. If $p \geq (1+\varepsilon) \cdot (2^n - 1)$ for some fixed constant ε, then Algorithm 3 terminates in expected $O(n^2 \cdot 2^n)$ time.*

Proof. The initial multiplication $\mathrm{im}_0 := A(\{0,1\}^n - K)$ takes $(n-k) \cdot n \cdot |\{0,1\}^n - K| \in O(n^2 \cdot 2^n)$ multiplications. Each iteration of Algorithm 3's main loop loop entails multiplying the $(n-k-l) \times (n-k-l+1)$ matrix A_{n-k-l} by the matrices im_{l-1} and H_{l-1}, which respectively have $|\{0,1\}^n - K|$ and n columns. Moreover, A_{n-k-1} consists of exactly one non-identity column, followed by the size-$(n-k-l)$ identity matrix. Both of these multiplications can thus be performed in $O(n \cdot 2^n)$ time. Beyond this, the inner **do–until** loop must sample an element $(u_1, \ldots, u_{n-k-l}) \in \mathbb{F}_p^{n-k-l}$ which "misses" $U_0 \cap \mathrm{im}_{l-1}$. The set $U_0 \cap \mathrm{im}_{l-1}$ contains at most $|\{0,1\}^n - K| \leq 2^n - 1$ elements. On the other hand, by hypothesis on p, $\left|\mathbb{F}_p^{n-k-l}\right| \geq p > 2^n - 1 \geq |\{0,1\}^n - K|$. Each iteration of the inner loop thus succeeds with positive probability. In fact, under the additional hypothesis that $p \geq (1+\varepsilon) \cdot 2^n$, each iteration of the inner loop succeeds with probability at least:

$$\frac{p^{n-k-l} - |U_0 \cap \mathrm{im}_l - 1|}{p^{n-k-l}} \geq \frac{p^{n-k-l} - (2^n - 1)}{p^{n-k-l}} \geq \frac{p - (2^n - 1)}{p} = 1 - \frac{2^n - 1}{p} \geq 1 - \frac{2^n - 1}{(1+\varepsilon) \cdot (2^n - 1)} = \frac{\epsilon}{1+\varepsilon}.$$

For each iteration of the outer loop, the inner loop thus succeeds after an expected $\frac{1+\varepsilon}{\varepsilon}$ iterations, and thus imposes at most a constant (in n) expected multiplicative overhead. This completes the proof. ☐

We conclude this section with the following "summary" algorithm:

Algorithm 4 HyperplaneFinder

```
Require: An arbitrary set $S \subset \{0,1\}^n$ and a prime $p \geq 2^n$.
Ensure: Hyperplanes $H_0, \ldots, H_{m-1}$ over $\mathbb{F}_p$ such that $S = \bigsqcup_{i=0}^{m-1} H_i \cap \{0,1\}^n$.
1: Obtain CONFIGURATION $C :=$ FlatFinder($S$).          ▷ See Algorithm 2.
2: return {FlatExtender($K$) | $K \in C.F$}             ▷ See Algorithm 3.
```

4.5 Implementation

In this subsection, we describe our implementation of the hyperplane-synthesing Algorithm 4. Our implementation is written in C++, and uses only the C++ Standard Library. We specialize p to the NIST P-256 prime $p = 2^{256} - 2^{224} + 2^{192} + 2^{96} - 1$ [Inf13, D.2.3]. Our algorithm requires that output quality and runtime be traded off. In particular, the branching factor b and the leaderboard size l are tunable parameters. We run our benchmarks using a handful of different parameter choices, in order to demonstrate the nature of the available tradeoffs.

We first focus on the case $n = 8$, and execute our algorithm on *random* subsets $S \subset \{0,1\}^8$. Each such subset can be represented uniquely as a 256-bit string. For each among a handful of arbitrarily chosen such strings—selected as "nothing-up-my-sleeve" hashes—we execute our algorithm 100 times, recording, in each execution, the time taken by the algorithm and the cardinality m of the covering obtained.

We also show that our algorithm can empirically "learn" the covering families already exactly described in Examples 3.22, 3.25, and 3.33 above. To this end, we add table entries for those functions, and run analogous benchmarks for them. Our benchmarks show that Algorithm 4 recovers the "right" covering (i.e., using 5 hyperplanes for the first two and 8 for the last) most or all of the time, at least for larger l and b (setting both to 3 essentially suffices, with diminishing returns for higher values).

In each cell, we report *average* values over 100 runs. All benchmarks are run on an AWS instance of `c5.4xlarge` type. Time values are all in milliseconds.

Algorithm 4 spends the vast majority of its time executing FlatFinder (Algorithm 2). In fact, each individual call to FlatExtender (Algorithm 3) just takes a couple milliseconds, with *lower*-dimensional flats more expensive to extend; a 0-dimensional flat, for example, takes as much as 16 milliseconds to extend (we don't implement the shortcut implicit in Lemma 3.10).

The flats K_i output by Algorithm 2 are typically relatively high-dimensional, with most between 5 and 7 dimensions in the case $n = 8$. Occasionally a single 0-dimensional flat also appears.

Table 2: Performance of algorithm (in both time and output quality) over various parameter choices.

		$b = 2$, $l = 2$		$b = 3$, $l = 3$		$b = 4$, $l = 4$	
Choice of $S \subset \{0,1\}^8$	$\lvert S\rvert$	Time	Hyp.s	Time	Hyp.s	Time	Hyp.s
$S :=$ SHA-256(`0x00`) = `6e340b9c...17afa01d`	127	99	17.29	550	13.64	3,651	12.17
$S :=$ SHA-256(`0x01`) = `4bf5122f...7785459a`	130	104	17.35	621	13.66	4,414	12.13
$S :=$ SHA-256(`0x02`) = `dbc1b4c9...6457d986`	128	102	17.11	634	13.57	4,270	12.06
$S :=$ SHA-256(`0x03`) = `084fed08...adff29c5`	126	98	16.95	589	13.39	3,916	11.86
$S :=$ SHA-256(`0x04`) = `e52d9c50...81c89e71`	119	91	16.51	527	12.91	3,613	11.61
$S :=$ SHA-256(`0x05`) = `e77b9a9a...5ab743db`	139	112	18.17	722	14.09	5,136	12.57
$S :=$ SHA-256(`0x06`) = `67586e98...08c5ecf6`	133	106	17.64	698	13.97	4,596	12.33
$S :=$ SHA-256(`0x07`) = `ca358758...005ee879`	143	112	18.32	731	14.44	4,964	12.80
Example 3.22: $\sum_{i=0}^{3} 2^i \cdot x_i \leq \sum_{l=4}^{7} 2^{i-4} \cdot x_i$	136	61	10.78	225	5.27	1,806	5.06
Example 3.25: $\left(\sum_{i=0}^{7} x_i\right) \geq 4$	163	48	5.94	613	5.00	5,514	5.00
Example 3.33: $\bigvee_{i=0}^{n-1} x_i$	255	134	8.00	2,268	8.00	21,505	8.00

Our algorithm of course can also handle larger subsets, though the runtimes get large. In the following benchmarks, we fix the parameterization $b = 2$ and $l = 1$. In each benchmark, we run our algorithm on a *single* random subset $S \subset \{0,1\}^n$. We report the time taken, and the number of hyperplanes used. All time values here are in *seconds*.

Table 3: Performance of algorithm (in both time and output quality) on higher-dimensional sets.

Dimension n	Cardinality $\lvert S\rvert$ of random subset $S \subset \{0,1\}^n$	Time Taken	Hyperplanes Used
8	124	0.057	15
9	256	0.271	31
10	495	1.142	55
11	1,034	3.874	95
12	2,088	17.698	173
13	4,111	75.420	305
14	8,112	348.748	546

5 Applications in Cryptography

In this section, we provide cryptographic applications of our "computation by hyperplanes" paradigm. We give both a semi-honestly secure two-party protocol and a maliciously secure three-party protocol (involving an untrusted third party without input). In both settings, an arbitrary boolean function $f : \{0,1\}^n \to \{0,1\}$ is evaluated, given a pre-computed hyperplane covering of $f^{-1}(1)$ (and also of $f^{-1}(0)$, in the malicious case). The covering's cardinality controls the efficiency of the protocol.

Our semi-honest protocol is sketched in Example 1.6 above, and is relatively straightforward; we describe it in detail in Protocol 5.1 below. Our malicious protocol is given in Protocol 5.5 below. The latter protocol, in fact, can be viewed as a vast generalization of an algorithm of Wagh, Gupta, and Chandran [WGC19, Alg. 3], as we now explain. Their protocol—though they don't express it in these terms—essentially expresses the "fixed-threshold comparator" function $f : \{0,1\}^n \to \{0,1\}$ as an on-set $S := f^{-1}(1) \subset \{0,1\}^n$, which, moreover, is covered by disjoint $\mathbb{F}_p$-hyperplanes over a fixed prime p (we discuss this further in Example 3.12 above). The two participating parties evaluate these hyperplanes jointly on a shared input $\mathbf{x} \in \{0,1\}^n$ (secret-sharing is a linear operation). The parties finally permute and re-randomize their output shares, using common randomness, before sending these to a third party, who, upon reconstructing their values, learns only whether a 0 is present, and hence whether $\mathbf{x} \in f^{-1}(1)$. These hyperplanes thus actually "randomize" the fixed-threshold comparator f, in the sense of Definition 3.35.

Our generalization of [WGC19, Alg. 3] introduces the use of *arbitrary* functionalities $f : \{0,1\}^n \to \{0,1\}$; these can be, in general, much more complex than that evaluated by [WGC19, Alg. 3]. Indeed—as we observe in Example 3.12 above—the fixed-threshold comparator $f : \{0,1\}^n \to \{0,1\}$ evaluated by [WGC19, Alg. 3] has an on-set consisting of a union of subcubes, and thus represents one of the simplest function-families computable by affine hyperplanes. Examples 3.23, 3.22, and 3.25 above demonstrate how significantly the power of hyperplanes can, in general, exceed that of subcubes, *even* when only polynomially (indeed, linearly) many hyperplanes are allowed (using arbitrarily many hyperplanes, one may of course evaluate arbitrary functions, as Corollary 3.13 demonstrates).

Separately, we show how to more fully exploit the affine-linearity of hyperplane representations, by adding *malicious* security to the above setting. Indeed, just as secret-sharing can be viewed as an $\mathbb{F}_p$-linear operation, so can commitment under an $\mathbb{F}_p$-homomorphic scheme (such as the Pedersen scheme). We thus observe that two parties may evaluate their affine circuit "in parallel" on secret-shares *and* on commitments, and so achieve maclicious security under one corruption.

Our use of commitments conveys additional advantages, beyond those associated with malicious security. Indeed, a significant literature (see e.g. Groth and Kohlweiss [GK15], Bünz, et al. [BBB+18]) has demonstrated the versatility and efficiency of zero-knowledge proof protocols which target languages concerning *commitments*, and which, in particular, assert that certain commitments' messages satisfy certain properties. Indeed, each party may couple its evaluation of an affine circuit, under our protocol, with a proof demonstrating that its input arguments belong to some particular language. As a natural example, each party may demonstrate to the other that the input wires it uses in some execution of the protocol match values committed to in a *prior* commitment. We call this feature *commitment-consistency*; it is not easy to achieve in standard protocols.

The only prior protocol which (possibly) achieves homomorphic commitment-consistency is, to our knowledge, that of Frederiksen, Pinkas, and Yanai [FPY18]. At a high level, our approach shares with that of [FPY18] the idea whereby the use of inconsistent or incorrect shares must ultimately be detected upon opening. As that protocol targets arithmetic circuits, it of course stands to gain equally from our boolean function representation paradigm. On the other hand, it requires a non-constant number of rounds (proportional to the number of multiplication gates in the circuit). Our protocol thus improves upon theirs by using only constantly many rounds (as well as through our general boolean-representation paradigm). We note that [FPY18] uses Beaver multiplication triples to evaluate field-multiplication gates; we instead express intrinsically *boolean* functions as randomizing polynomials (see Subsection 3.4 above), and handle non-linearity by evaluating (disjoint) *unions* of hyperplanes, as opposed to individual ones. Finally, our protocol is *implemented*, and is plausibly much more efficient. We thus give the first *constant-round*, homomorphic commitment consistent protocol for general two-party computation.

5.1 Semi-honest protocol

We now give our semi-honest protocol. We assume that all parties have agreed upon a group-generation algorithm $\mathcal{G}$ and a corresponding homomorphic encryption scheme $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ (for example, the El Gamal scheme of Example 2.10 suffices).

PROTOCOL 5.1 (Semi-honest protocol)**.**
All parties have a function $f : \{0,1\}^n \to \{0,1\}$, where n is even.

- **Setup:** All parties run $(\mathbb{G}, p, g) \leftarrow \mathcal{G}(1^\lambda)$. All parties agree on a disjoint covering $f^{-1}(1) = \bigsqcup_{i=0}^{m-1} H_i \cap \{0,1\}^n$ using $\mathbb{F}_p$-hyperplanes.

P_0 and P_1 hold elements $\mathbf{x}_0$ and $\mathbf{x}_1$ of $\{0,1\}^{n/2}$.

- **First phase:** Empty.
- **Second phase:** P_0 and P_1 proceed as follows.
  1. P_0 runs $(pk, sk) \leftarrow \mathsf{Gen}(1^\lambda)$. For each $i \in \{0, \ldots, \frac{n}{2} - 1\}$, P_0 computes $A_i \leftarrow \mathsf{Enc}_{pk}(x_{0,i})$, where $\mathbf{x}_0 = (x_{0,0}, \ldots, x_{0,n/2-1})$. P_0 sends pk and the array $\{A_i\}_{i=0}^{n/2-1}$ to P_1.
  2. P_1 evaluates the hyperplanes $(H_i)_{i=0}^{m-1}$ on $\{A_i\}_{i=0}^{n/2-1}$ and on $\mathbf{x}_1 = (x_{1,0}, \ldots, x_{1,n/2-1})$ homomorphically; that is, P_1 runs:
$$(D_i)_{i=0}^{m-1} := \left(H_i(A_0, \ldots, A_{n/2-1}, x_{1,0}, \ldots, x_{1,n/2-1})\right)_{i=0}^{m-1}. \tag{1}$$
     P_1 generates random scalars $(\alpha_i)_{i=0}^{m-1}$ in $\mathbb{F}_p^*$ and a random circular shift permutation $\rho \in \langle(0, 1, \ldots, m-1)\rangle \subset \mathbf{S}_m$. P_1 then overwrites $D_i := \alpha_i \cdot D_{\rho(i)}$ for each $i \in \{0, \ldots, m-1\}$. P_1 re-randomizes each D_i by adding a random encryption of 0; that is, P_1 overwrites $D_i := D_i \cdot \mathsf{Enc}_{pk}(0)$. P_1 finally sends the overwritten output array $(D_i)_{i=0}^{m-1}$ to P_0.
  3. P_0 decrypts $y_i := \mathsf{Dec}_{sk}(D_i)$ for each $i \in \{0, \ldots, m-1\}$. P_0 runs the extractor X of Theorem 3.36 on the resulting plaintexts $(y_i)_{i=0}^{m-1}$; that is, P_0 checks whether a 0 is present in the array $(y_i)_{i=0}^{m-1}$. P_0 returns whatever X does, and also reports the result to P_1.

Theorem 5.2. *If* $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *has indistinguishable multiple encryptions, then Protocol 5.1 securely computes Functionality 2.14 in the presence of one semi-honest corruption.*

Proof. The correctness of the protocol is self-evident, and follows from the correctness property of Theorem 3.36. We define a simulator $\mathcal{S}$ which satisfies the properties required by Definition 2.15. We fix a functionality $\mathcal{F}$ in the sense of Functionality 2.14, with function $f : \{0,1\}^n \to \{0,1\}$, say, and a corrupt party $C \in \{0,1\}$.

On input $(1^\lambda, C, \mathbf{x}_C, v)$, $\mathcal{S}$ first runs the setup procedure of Protocol 5.1, and so obtains a prime p and a covering $f^{-1}(1) = \bigsqcup_{i=0}^{m-1} H_i \cap \{0,1\}^n$. We now treat separately the cases $C = 0$ and $C = 1$.

We suppose first that $C = 0$. $\mathcal{S}$ runs the randomizing polynomial simulator of Theorem 3.36 above, and so obtains m field elements $(y_i)_{i=0}^{m-1}$, exactly one of which is 0 if and only if $v = 1$. $\mathcal{S}$ then generates a random keypair $(pk, sk) \leftarrow \mathsf{Gen}(1^\lambda)$ and encrypts $D_i \leftarrow \mathsf{Enc}_{pk}(y_i)$ for each $i \in \{0, \ldots, m-1\}$. $\mathcal{S}$ outputs $(D_i)_{i=0}^{m-1}$. We now argue that the distributions $\mathsf{Real}_\Pi(\lambda, 0; \mathbf{x}_0, \mathbf{x}_1)$ and $\mathsf{Ideal}_{\mathcal{F},\mathcal{S}}(\lambda, 0; \mathbf{x}_0, \mathbf{x}_1)$ are identical. This follows from the correctness of the simulator of Theorem 3.36. Indeed, in both distributions, the ciphertexts $(D_i)_{i=0}^{m-1}$ have messages distributed exactly as in D_v and independent uniformly random randomnesses.

We suppose now that $C = 1$. $\mathcal{S}$ generates a random keypair $(pk, sk) \leftarrow \mathsf{Gen}(1^\lambda)$. $\mathcal{S}$ then simulates the initial ciphertexts $(A_i)_{i=0}^{n/2-1}$ received by P_1 as random encryptions of 0; that is, $\mathcal{S}$ sets $A_i \leftarrow \mathsf{Enc}_{pk}(0)$ for each $i \in \{0, \ldots \frac{n}{2} - 1\}$. We argue now that the distributions $\mathsf{Real}_\Pi(\lambda, 1; \mathbf{x}_0, \mathbf{x}_1)$ and $\mathsf{Ideal}_{\mathcal{F},\mathcal{S}}(\lambda, 1; \mathbf{x}_0, \mathbf{x}_1)$ are computationally indistinguishable. We abbreviate these distributions in the following way:

D_0: Corresponds to $\mathsf{Real}_\Pi(\lambda, 1; \mathbf{x}_0, \mathbf{x}_1)$, i.e., the view V_1.

D_1: Corresponds to $\mathsf{Ideal}_{\mathcal{F},\mathcal{S}}(\lambda, 1; \mathbf{x}_0, \mathbf{x}_1)$, i.e., the output $S(1^\lambda, 1, \mathbf{x}_1, v)$.

We suppose by contradiction that there were a distinguisher D and a polynomial $p(\lambda)$ for which, for a sequence of triples $(\lambda, \mathbf{x}_0, \mathbf{x}_1)$ in which infinitely many distinct values λ are represented, $|\Pr[D(\mathsf{D}_0(\lambda, \mathbf{x}_0, \mathbf{x}_1)) = 1] - \Pr[D(\mathsf{D}_1(\lambda, \mathbf{x}_0, \mathbf{x}_1)) = 1]| \geq \frac{1}{p(\lambda)}$. Without loss of generality—and after possibly flipping D's output bit—we may assume that $\Pr[D(\mathsf{D}_1(\lambda, \mathbf{x}_0, \mathbf{x}_1)) = 1] - \Pr[D(\mathsf{D}_0(\lambda, \mathbf{x}_0, \mathbf{x}_1)) = 1] \geq \frac{1}{p(\lambda)}$ for infinitely many values λ and appropriate $(\mathbf{x}_0, \mathbf{x}_1)$. We define a nonuniform adversary $\mathcal{A}$ attacking the multiple encryptions experiment $\mathsf{PubK}^{\mathsf{LR\text{-}cpa}}_{\Pi,\mathcal{A}}$ as follows. Using the advice $(\mathbf{x}_0, \mathbf{x}_1)$, and given input pk and access to $\mathsf{LR}_{pk,b}(\cdot,\cdot)$, $\mathcal{A}$ generates $A_i \leftarrow \mathsf{LR}_{pk,b}(x_{0,i}, 0)$ for each $i \in \{0, \ldots, \frac{n}{2} - 1\}$, and so obtains a view V_0 given by $(A_i)_{i=0}^{n/2-1}$. $\mathcal{A}$ then runs D on V_0, and outputs whatever D outputs. If the experimenter's hidden bit $b = 0$, then V_0 is distributed exactly as in $\mathsf{D}_0(\lambda, \mathbf{x}_0, \mathbf{x}_1)$. If the experimenter's hidden bit $b = 1$, then V_0 is distributed exactly as in $\mathsf{D}_1(\lambda, \mathbf{x}_0, \mathbf{x}_1)$. We thus have:

$$\begin{aligned}
\Pr[\mathsf{PubK}^{\mathsf{LR\text{-}cpa}}_{\Pi,\mathcal{A}}(\lambda) = 1] &= \frac{1}{2} \cdot \Pr[\mathsf{out}_\mathcal{A}\left(\mathsf{PubK}^{\mathsf{LR\text{-}cpa}}_{\Pi,\mathcal{A}}(\lambda)\right) = 0 \mid b = 0] + \frac{1}{2} \cdot \Pr[\mathsf{out}_\mathcal{A}\left(\mathsf{PubK}^{\mathsf{LR\text{-}cpa}}_{\Pi,\mathcal{A}}(\lambda)\right) = 1 \mid b = 1] \\
&= \frac{1}{2} \cdot \left(1 - \Pr_{V_0 \leftarrow \mathsf{D}_0(\lambda, \mathbf{x}_0, \mathbf{x}_1)}[D(V_0) = 1] + \Pr_{V_0 \leftarrow \mathsf{D}_1(\lambda, \mathbf{x}_0, \mathbf{x}_1)}[D(V_0) = 1]\right) \\
&= \frac{1}{2} + \frac{1}{2} \cdot \left(\Pr_{V_0 \leftarrow \mathsf{D}_1(\lambda, \mathbf{x}_0, \mathbf{x}_1)}[D(V_0) = 1] - \Pr_{V_0 \leftarrow \mathsf{D}_0(\lambda, \mathbf{x}_0, \mathbf{x}_1)}[D(V_0) = 1]\right).
\end{aligned}$$

Our assumption on D would now show that $\Pr[\mathsf{PubK}^{\mathsf{LR\text{-}cpa}}_{\Pi,\mathcal{A}}(\lambda) = 1] - \frac{1}{2} \geq \frac{1}{2 \cdot p(\lambda)}$ for infinitely many λ, which would contradict the security of $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$. This completes the proof of the case $C = 1$. □

Remark 5.3. We note that commitment-consistency comes "for free" in the semi-honest setting; in this setting, we simply trust that both P_0 and P_1 use consistent secret inputs.

Theorem 5.4. *Suppose that $\{f_n\}_{n \in \mathbb{N}}$ is efficiently computable by disjoint hyperplanes, and in particular admits coverings $f_n^{-1}(1) = \bigsqcup_{i=0}^{m-1} H_i \cap \{0,1\}^n$ for $m = \mathsf{poly}(n)$. Then Protocol 5.1 above evaluates f_n in $O(n \cdot m)$ time, using $O(n + m)$ communication, and in three rounds.*

Proof. It takes P_0 $O(n)$ time to encrypt its inputs, and P_1 $O(n \cdot m)$ total time to evaluate the hyperplanes in (1). The final ciphertexts take P_0 $O(m)$ time to decrypt. P_0 must send $O(n)$ bits' worth of ciphertexts to P_1; P_1 must respond with $O(m)$ bits' worth of output ciphertexts. The protocol obviously takes only 3 rounds (in fact, only 2 if P_1 doesn't need the output). □

5.2 Maliciously secure protocol

We now give our malicious protocol for Functionality 2.14. The rough idea is that P_0 and P_1 run the semi-honest Protocol 5.1 simultaneously on *shares* which belong to them and on *commitments* to shares which belong to the other party. That is, each player performs the other player's calculation "through their commitments" and "checks the other party's work". Finally, each party sends both its raw outputs and its commitments to P_2, who checks that each party's outputs match the other party's commitments to those outputs. If at least one among the players P_0 and P_1 is honest, then any deviation from the protocol will be caught by P_2 during this step.

Each player uses efficient zero-knowledge proofs to convince the other that its inputs are *actually bits* and moreover correspond to its initial commitment.

Finally, P_2 must convince P_0 and P_1 that it reconstructed the output correctly. To do this, P_2 generates a *one-out-of-many proof* demonstrating that the appropriate array of commitments contains a commitment to 0. Interestingly, this pre-existing zero-knowledge protocol exactly suits the randomizing polynomial reconstructor of Theorem 3.36 (for which the existence of a 0 exactly reflects the boolean output value). This proof reveals nothing about the raw reconstructed output values, and in particular about where the 0 resides (which would leak information to P_0 and P_1).

We assume that, globally, all parties have agreed on a pseudorandom generator G, a group-generation algorithm $\mathcal{G}$, a homomorphic commitment scheme $(\mathsf{Gen}, \mathsf{Com})$ (for example, the Pedersen scheme of Example 2.13), and a key-exchange protocol Ξ (for example, the Diffie–Hellman protocol of Example 2.8).

PROTOCOL 5.5 (Maliciously secure protocol)**.**
All parties have a function $f : \{0,1\}^n \to \{0,1\}$, where n is even.

- **Setup:** All parties run $(\mathbb{G}, p, g) \leftarrow \mathcal{G}(1^\lambda)$ and generate commitment parameters $\mathsf{params} \leftarrow \mathsf{Gen}(1^\lambda)$. All parties agree on disjoint coverings $f^{-1}(0) = \bigsqcup_{i=0}^{m_0-1} H_{0,i} \cap \{0,1\}^n$ and $f^{-1}(1) = \bigsqcup_{i=0}^{m_1-1} H_{1,i} \cap \{0,1\}^n$ using $\mathbb{F}_p$-hyperplanes (for notational convenience, we assume that $m_0 = m_1$ and write m for the common quantity).

P_0 and P_1 hold elements $\mathbf{x}_0$ and $\mathbf{x}_1$ of $\{0,1\}^{n/2}$.

- **First phase:** For each $\nu \in \{0,1\}$, P_ν commits $A_\nu \leftarrow \mathsf{Com}\left(\sum_{i=0}^{n/2-1} 2^i \cdot x_{\nu,i}\right)$, where $\mathbf{x}_\nu = \left(x_{\nu,0}, \ldots, x_{\nu,n/2-1}\right)$. P_ν sends A_ν to $P_{1-\nu}$.
- **Second phase:** P_0 and P_1 run Ξ, and so obtain a shared key ξ. Using ξ and G, P_0 and P_1 generate shared random values $(\alpha_{j,i})_{j,i=0}^{1,m-1}$ in $\mathbb{F}_p^*$, two shared random circular shift permutations ρ_0 and ρ_1 in $\langle(0,1,\ldots,m-1)\rangle \subset \mathbf{S}_m$, and shared uniform values $(\beta_{j,i,\nu})_{j,i,\nu=0}^{1,n/2-1,1}$ in $\mathbb{F}_p$. Each party P_ν, $\nu \in \{0,1\}$, then proceeds as follows:
  1. For each $i \in \{0,\ldots,\frac{n}{2}-1\}$, P_ν computes a random secret-sharing $x_{\nu,i} = \langle x_{\nu,i}\rangle_0 + \langle x_{\nu,i}\rangle_1$ in $\mathbb{F}_p$, where $\mathbf{x}_\nu = \left(x_{\nu,0}, \ldots, x_{\nu,n/2-1}\right)$. For $i \in \{0,\ldots,\frac{n}{2}-1\}$ and $j \in \{0,1\}$, P_ν commits $A_{\nu,i,j} := \mathsf{Com}(\langle x_{\nu,i}\rangle_j\, ; r_{\nu,i,j})$. P_i sends the full array $(A_{\nu,i,j})_{i,j=0}^{n/2-1,1}$ to $P_{1-\nu}$. For each $i \in \{0,\ldots,\frac{n}{2}-1\}$, P_i opens $A_{\nu,i,1-\nu}$ by directly sending $\langle x_{\nu,i}\rangle_{1-\nu}$ and $r_{\nu,i,1-\nu}$ to $P_{1-\nu}$. P_ν also computes $\pi_\nu \leftarrow \mathsf{ComEq.Prove}\left(A_\nu, \prod_{i=0}^{n/2-1} \left(A_{\nu,i,0} \cdot A_{\nu,i,1}\right)^{2^i}\right)$, as well as $\pi_{\nu,i} \leftarrow \mathsf{BitProof.Prove}\left(A_{\nu,i,0} \cdot A_{\nu,i,1}\right)$ for each $i \in \{0,\ldots,\frac{n}{2}-1\}$. P_ν sends π_ν and $(\pi_{\nu,i})_{i=0}^{n/2-1}$ to $P_{1-\nu}$.
  2. Symmetrically, P_ν checks that the openings $\langle x_{1-\nu,i}\rangle_\nu$ and $r_{1-\nu,i,\nu}$ indeed open $A_{1-\nu,i,\nu}$, for $i \in \{0,\ldots,\frac{n}{2}-1\}$. P_ν checks $\mathsf{ComEq.Verify}\left(\pi_{1-\nu}, A_{1-\nu}, \prod_{i=0}^{n/2-1} \left(A_{1-\nu,i,0} \cdot A_{1-\nu,i,1}\right)^{2^i}\right)$, as well as $\mathsf{BitProof.Verify}\left(\pi_{1-\nu,i}, A_{1-\nu,i,0} \cdot A_{1-\nu,i,1}\right)$ for each $i \in \{0,\ldots,\frac{n}{2}-1\}$. If any verifications fail, P_ν aborts.
  3. P_ν evaluates the hyperplanes $(H_{j,i})_{j,i=0}^{1,m-1}$, in parallel, on the shares $\langle x_{j,i}\rangle_\nu$, the randomnesses $r_{j,i,\nu}$, and the commitments to the *other party*'s shares $A_{j,i,1-\nu}$. That is, P_ν runs:

$$\left(\langle y_{j,i}\rangle_\nu\right)_{j,i=0}^{1,m-1} := \left(H_{j,i}\left(\langle x_{0,0}\rangle_\nu, \ldots, \langle x_{0,n/2-1}\rangle_\nu, \langle x_{1,0}\rangle_\nu, \ldots, \langle x_{1,n/2-1}\rangle_\nu\right)\right)_{j,i=0}^{1,m-1}, \quad (2)$$

$$\left(s_{j,i,\nu}\right)_{j,i=0}^{1,m-1} := \left(H_{j,i}\left(r_{0,0,\nu}, \ldots, r_{0,n/2-1,\nu}, r_{1,0,\nu}, \ldots, r_{1,n/2-1,\nu}\right)\right)_{j,i=0}^{1,m-1}, \quad (3)$$

$$\left(D_{j,i,1-\nu}\right)_{j,i=0}^{1,m-1} := \left(H_{j,i}\left(A_{0,0,1-\nu},\ldots,A_{0,n/2-1,1-\nu},A_{1,0,1-\nu},\ldots,A_{1,n/2-1,1-\nu}\right)\right)_{j,i=0}^{1,m-1}. \quad (4)$$

P_ν overwrites $\langle y_{j,i}\rangle_\nu := \alpha_{j,i}\cdot\langle y_{j,\rho_j(i)}\rangle_\nu$, $s_{j,i,\nu} := \beta_{j,i,\nu} + \alpha_{j,i}\cdot s_{j,\rho_j(i),\nu}$, and $D_{j,i,1-\nu} := \mathsf{Com}(0;\beta_{j,i,1-\nu})\cdot\left(D_{j,\rho_j(i),1-\nu}\right)^{\alpha_{j,i}}$ for each $j\in\{0,1\}$ and $i\in\{0,\ldots,m-1\}$. P_i sends the output shares $\left(\langle y_{j,i}\rangle_\nu\right)_{j,i=0}^{1,m-1}$, the randomnesses $(s_{j,i,\nu})_{j,i=0}^{1,m-1}$, and the commitments $\left(D_{j,i,1-\nu}\right)_{j,i=0}^{1,m-1}$ to P_2.

After receiving all bits of information, P_2 proceeds as follows:

4. For each $\nu\in\{0,1\}$, P_2 checks that the openings $\left(\langle y_{j,i}\rangle_\nu\right)_{j,i=0}^{1,m-1}$ and $(s_{j,i,\nu})_{j,i=0}^{1,m-1}$ indeed open the commitments $\left(D_{j,i,\nu}\right)_{j,i=0}^{1,m-1}$ sent to it by the opposite party. If any checks fail, P_2 aborts.
5. P_2 reconstructs $y_{j,i} := \langle y_{j,i}\rangle_0 + \langle y_{j,i}\rangle_1$ and $s_{j,i} := s_{j,i,0}+s_{j,i,1}$ for each $j\in\{0,1\}$ and $i\in\{0,\ldots,m-1\}$. P_2 runs the extractor X of Theorem 3.36 on both outputs $(y_{0,i})_{i=0}^{m-1}$ and $(y_{1,i})_{i=0}^{m-1}$. If these values are not *distinct*—that is, if there is not a unique value $j\in\{0,1\}$ for which $y_{j,i}=0$ for exactly one $i\in\{0,\ldots,m-1\}$—P_2 aborts. Otherwise, P_2 writes v for this value. Finally, P_2 sets $D_{v,i} := \mathsf{Com}(y_{v,i},s_{v,i})$ for each $i\in\{0,\ldots,m-1\}$ and computes $\pi\leftarrow\mathsf{OneOutOfMany.Prove}\left((D_{v,i})_{i=0}^{m-1}\right)$. P_2 sends v and π to P_0 and P_1.
6. Each party P_ν locally computes $D_{v,i} := \mathsf{Com}(\langle y_{v,i}\rangle_\nu; s_{v,i,\nu})\cdot D_{v,i,1-\nu}$, for each $i\in\{0,\ldots,m-1\}$, and then verifies $\mathsf{OneOutOfMany.Verify}\left(\pi,(D_{v,i})_{i=0}^{m-1}\right)$. If it passes, then P_ν outputs v.

Theorem 5.6. *If Ξ is secure in the presence of an eavesdropper, G is a pseudorandom generator, and* $(\mathsf{Gen},\mathsf{Com})$ *is both hiding and binding, then Protocol 5.5 securely computes Functionality 2.14 in the presence of one static malicious corruption (with fairness if P_0 or P_1 is corrupted and without fairness if P_2 is).*

Proof. We define a simulator $\mathcal{S}$ satisfying the properties of Definition 2.16. We fix a functionality $\mathcal{F}$ (see Functionality 2.14), with function $f:\{0,1\}^n\to\{0,1\}$, say. We let the corrupt party $C\in\{0,1,2\}$ be fixed, and fix a real-world adversary $\mathcal{A}$ corrupting C.

On input $(1^\lambda, C)$, $\mathcal{S}$ first runs the setup procedure of Protocol 5.5, and so obtains group parameters $(\mathbb{G},p,g)$ and commitment parameters params, as well as coverings $f^{-1}(0)=\bigsqcup_{i=0}^{m-1}H_{0,i}\cap\{0,1\}^n$ and $f^{-1}(1)=\bigsqcup_{i=0}^{m-1}H_{1,i}\cap\{0,1\}^n$. We now treat separately the cases $C\in\{0,1\}$ and $C=2$.

We suppose first that $C\in\{0,1\}$. $\mathcal{S}$ operates as follows:

1. In the first stage, $\mathcal{S}$ simulates the initial commitment A_{1-C} sent by P_{1-C} as a random commitment to 0, and receives from $\mathcal{A}$ a commitment A_C.

2. $\mathcal{S}$ simulates the openings $\left(\langle x_{1-C,i}\rangle_C, r_{1-C,i,C}\right)_{i=0}^{n/2-1}$ received from P_{1-C} as random $\mathbb{F}_p$-elements. $\mathcal{S}$ constructs the commitments $(A_{1-C,i,C})_{i=0}^{n/2-1}$ directly from these openings, and simulates the remaining commitments $(A_{1-C,i,1-C})_{i=0}^{n/2-1}$ as random commitments to 0. Using the simulator M guaranteed to exist by Theorems 2.23 and 2.25 (see Definition 2.20), $\mathcal{S}$ simulates the proofs π_{1-C} and $(\pi_{1-C,i})_{i=0}^{n/2-1}$ received from P_{1-C}.

3. If $A_{C,i,1-C}\neq\mathsf{Com}(\langle x_{C,i}\rangle_{1-C}; r_{C,i,1-C})$ holds for any $i\in\{0,\ldots,\frac{n}{2}-1\}$, then $\mathcal{S}$ sends $\perp$ to $\mathcal{F}$ and halts. Likewise, if $\mathsf{ComEq}.V\left(\pi_C, A_C, \prod_{i=0}^{n/2-1}(A_{C,i,0}\cdot A_{C,i,1})^{2^i}\right)$ fails, or if any of the checks $\mathsf{BitProof}.V\left(\pi_{C,i}, A_{C,i,0}\cdot A_{C,i,1}\right)$, for $i\in\{0,\ldots,\frac{n}{2}-1\}$, fails, then $\mathcal{S}$ outputs $\perp$ and aborts.

4. If all of decommitments and proofs pass, then $\mathcal{S}$ runs the machine guaranteed to exist by Lemma 2.30 (see also Theorems 2.23 and 2.25) on $\mathcal{A}$ (we may as well view the parallel protocols ComEq and $\mathsf{BitProof}$ as a single Σ-protocol; see [HL10, §6.4]). Unless the machine outputs $\perp$, $\mathcal{S}$ obtains a witness $(x_{C,i}, r_{C,i})$ for $A_{C,i,0}\cdot A_{C,i,1}$, for each $i\in\{0,\ldots,\frac{n}{2}-1\}$—where each $x_{C,i}\in\{0,1\}$—and in particular obtains the input string $\mathbf{x}_C := \left(x_{C,0},\ldots,x_{C,n/2-1}\right)$ (though does *not* yet give it to $\mathcal{F}$). By subtracting

from the newly acquired openings $(x_{C,i}, r_{C,i})_{i=0}^{n/2-1}$ the initial sent openings $\left(\langle x_{C,i}\rangle_{1-C}, r_{C,i,1-C}\right)_{i=0}^{n/2-1}$, $\mathcal{S}$ obtains further openings $\left(\langle x_{C,i}\rangle_C, r_{C,i,C}\right)_{i=0}^{n/2-1}$ (say) of the remaining commitments $(A_{C,i,C})_{i=0}^{n/2-1}$.

5. When $\mathcal{A}$ outputs $\left(\langle y_{j,i}\rangle_C, s_{j,i,C}\right)_{j,i=0}^{1,m-1}$ and $(D_{j,i,1-C})_{j,i=0}^{1,m-1}$ in 3., $\mathcal{S}$ recomputes the quantities $(D_{j,i,1-C})_{j,i=0}^{1,m-1}$ from the public commitments $(A_{j,i,1-C})_{i=0}^{n/2-1}$ (using (4) above), and aborts unless $\mathcal{A}$'s output commitments $(D_{j,i,1-C})_{j,i=0}^{1,m-1}$ match $\mathcal{S}$'s reconstructions. Separately, $\mathcal{S}$ freshly computes the quantities $(D_{j,i,C})_{j,i=0}^{1,m-1}$ from the public commitments $(A_{j,i,C})_{j,i=0}^{1,n/2-1}$ (as P_{1-C} would in 4), and aborts unless the $\mathcal{A}$'s output openings $\left(\langle y_{j,i}\rangle_C, s_{j,i,C}\right)_{j,i=0}^{1,m-1}$ decommit to $\mathcal{S}$'s reconstructions.

6. $\mathcal{S}$ finally gives $\mathbf{x}_C$ to $\mathcal{F}$, and obtains $v := \mathcal{F}(\mathbf{x}_0, \mathbf{x}_1)$ in return. $\mathcal{S}$ independently computes the local quantities $D_{v,i} := \mathsf{Com}(\langle y_{v,i}\rangle_C; s_{v,i,C}) \cdot D_{v,i,1-C}$ for $i \in \{0, \ldots, \frac{n}{2} - 1\}$. Using the simulator M for the protocol $\mathsf{OneOutOfMany}$, $\mathcal{S}$ simulates the received proof on the statement $(D_{v,i})_{i=0}^{n/2-1}$. $\mathcal{S}$ outputs all simulated quantities, together with v.

By Lemma 2.30, $\mathcal{S}$ runs in expected polynomial time. We now claim that the distributions $\mathsf{Real}_{\Pi,\mathcal{A}}\left(\lambda, C, (\mathbf{x}_\nu)_{\nu\neq C}\right)$ and $\mathsf{Ideal}_{\mathcal{F},\mathcal{S}}\left(\lambda, C, (\mathbf{x}_\nu)_{\nu\neq C}\right)$ are indistinguishable. We describe a sequence of distributions which interpolates between them:

D_0: Corresponds to $\mathsf{Real}_{\Pi,\mathcal{A}}\left(\lambda, C, (\mathbf{x}_\nu)_{\nu\neq C}\right)$, i.e., the pair $(V_C, (v_\nu)_{\nu\neq C})$.

D_1: Same as D_0, except P_2's final check at step 4. is replaced by that made by $\mathcal{S}$ in step 5. above. That is, P_2 is given all the initial commitments $(A_{j,i,\nu})_{j,i,\nu=0}^{1,n/2-1,1}$. P_2 recomputes $(D_{j,i,1-C})_{j,i=0}^{1,m-1}$ using (4), and aborts unless $\mathcal{A}$'s outputs $(D_{j,i,1-C})_{j,i=0}^{1,m-1}$ match these. Likewise, P_2 recomputes $(D_{j,i,C})_{j,i=0}^{1,m-1}$ using (4), and aborts unless $\mathcal{A}$'s outputs $\left(\langle y_{j,i}\rangle_C, s_{j,i,C}\right)_{j,i=0}^{1,m-1}$ decommit to these.

D_2: Same as D_1, except P_2 replaces the final one-out-of-many proof π by a simulation.

D_3: Same as D_2, except all commitments A_{1-C} and $(A_{1-C,i,1-C})_{i=0}^{n/2-1}$ output by P_{1-C} are replaced with random commitments to 0, and all proofs π_{1-C}, $(\pi_{1-C,i})_{i=0}^{n/2-1}$, and π are replaced by simulations.

D_4: Exactly $\mathsf{Ideal}_{\mathcal{F},\mathcal{S}}\left(\lambda, C, (\mathbf{x}_\nu)_{\nu\neq C}\right)$; i.e., $(V_C, (v)_{v\neq C})$, where V_C is the simulated view output by $\mathcal{S}$.

Lemma 5.7. *The distributions D_0 and D_1 are identical.*

Proof. When P_{1-C} is honest, for any choice of honest input $\mathbf{x}_{1-C}$, the openings $\left(\langle y_{j,i}\rangle_{1-C}, s_{j,i,1-C}\right)_{j,i=0}^{1,m-1}$ it outputs in step 3. necessarily commit to the *correctly* computed commitments $(D_{j,i,1-C})_{j,i=0}^{1,m-1}$. In both D_0 and in D_1, P_2's first abort condition thus amounts to checking whether $\mathcal{A}$ correctly computed (4), and these conditions are identical. Separately, when P_{1-C} is honest, the commitments $(D_{j,i,C})_{j,i=0}^{1,m-1}$ it (correctly) outputs in 3. can be immediately computed from the public initial commitments $(A_{j,i,C})_{j,i=0}^{1,n/2-1}$, and nothing changes if P_2 recomputes these freshly in its second abort condition. ☐

Lemma 5.8. *If* $(\mathsf{Gen}, \mathsf{Com})$ *is hiding, then the distributions D_1 and D_2 are indistinguishable.*

Proof. We suppose that some distinguisher D satisfies $|\Pr[D(\mathsf{D}_1(\lambda, \mathbf{x}_{1-C})) = 1] - \Pr[D(\mathsf{D}_2(\lambda, \mathbf{x}_{1-C})) = 1]| \geq \frac{1}{p(\lambda)}$ for some polynomial $p(\lambda)$ and an infinite sequence of pairs $(\lambda, \mathbf{x}_{1-C})$. It follows by a counting argument that, for each such pair $(\lambda, \mathbf{x}_{1-C})$, there must exist *at least one* initial interaction V_C^*—including the ultimate one-out-of-many statement $(x, w) := \left((D_{v,i})_{i=0}^{m-1}, (y_{v,i}, s_{v,i})_{i=0}^{m-1}\right)$, say—for which, even conditioned on V_C's initial portion equalling V_C^*, it remains true that $|\Pr[D(\mathsf{D}_1(\lambda, \mathbf{x}_{1-C})) = 1] - \Pr[D(\mathsf{D}_2(\lambda, \mathbf{x}_{1-C})) = 1]| \geq \frac{1}{p(\lambda)}$. We now define a distinguisher D' acting on proofs. For each among the infinitely many resulting such pairs (λ, x, w), D' may prepend the advice V_C^* to its received proof (a, e, z), run D on the resulting view, and output whatever D outputs. It is clear that $|\Pr[D'(\langle P(\lambda, x, w), V(x)\rangle) = 1] - \Pr[D'(M(\lambda, x)) = 1]|$ equals $|\Pr[D(\mathsf{D}_1(\lambda, \mathbf{x}_{1-C})) = 1] - \Pr[D(\mathsf{D}_2(\lambda, \mathbf{x}_{1-C})) = 1]|$, where again in the latter difference we condition on V_C's initial portion equalling V_C^* throughout. Our assumption on D would thus contradict the honest-verifier zero knowledge property of $\mathsf{OneOutOfMany}$, which holds whenever Com is hiding (see Theorem 2.28). □

Lemma 5.9. *If* $(\mathsf{Gen}, \mathsf{Com})$ *is hiding, then the distributions* D_2 *and* D_3 *are indistinguishable.*

Proof. These distributions differ only in that certain commitments are simulated (we recall that ComEq and $\mathsf{BitProof}$ are Σ-protocols, and admit perfect simulators). The lemma thus follows from a direct reduction to Com's hiding property. Indeed, we fix a distinguisher D between D_2 and D_3 and polynomial $p(\lambda)$ for which $\Pr[D(\mathsf{D}_3(\lambda, \mathbf{x}_{1-C})) = 1] - \Pr[D(\mathsf{D}_2(\lambda, \mathbf{x}_{1-C})) = 1] \geq \frac{1}{p(\lambda)}$ for infinitely many $(\lambda, \mathbf{x}_{1-C})$ (we may remove the absolute value bars without loss of generality, after possibly flipping D's output bit and refining the infinite set of λ). We define a nonuniform adversary $\mathcal{A}'$ attacking $\mathsf{Hiding}_{\mathsf{Com},\mathcal{A}'}$ in the obvious way. Using the advice $\mathbf{x}_{1-C}$, on input params, $\mathcal{A}'$ simulates an execution of D_2 in which the parameters params are used. $\mathcal{A}'$ generates A_{1-C} by calling $\mathsf{LR}_{\mathsf{params},b}\left(\sum_{i=0}^{n/2-1} 2^i \cdot x_{1-C,i}, 0\right)$; likewise, $\mathcal{A}'$ sets $A_{1-C,i,1-C} \leftarrow \mathsf{LR}_{\mathsf{params},b}\left(\langle x_{1-C,i}\rangle_{1-C}, 0\right)$ for each $i \in \{0, \ldots, \frac{n}{2} - 1\}$. Finally, $\mathcal{A}'$ simulates all proofs. Having constructed a view V_C in this way, $\mathcal{A}'$ runs D on V_C, and outputs whatever D outputs.

It is clear that in the cases $b = 0$ and $b = 1$, V_C is distributed exactly as in D_2 and D_3, respectively. It follows exactly as above that:

$$\Pr\left[\mathsf{Hiding}_{\mathsf{Com},\mathcal{A}'}(\lambda) = 1\right] - \frac{1}{2} = \frac{1}{2} \cdot \left(\Pr_{V_C \leftarrow \mathsf{D}_3(\lambda, \mathbf{x}_{1-C})}[D(V_C) = 1] - \Pr_{V_C \leftarrow \mathsf{D}_2(\lambda, \mathbf{x}_{1-C})}[D(V_C) = 1]\right).$$

Our hypothesis on D would contradict the assumed hiding property of Com. □

Lemma 5.10. *If* $(\mathsf{Gen}, \mathsf{Com})$ *is binding, then the distributions* D_3 *and* D_4 *are indistinguishable.*

Proof. We assume by contradiction that, for some distinguisher D, a polynomial $p(\lambda)$ and infinitely many pairs $(\lambda, \mathbf{x}_{1-C})$, it holds that $|\Pr[D(\mathsf{D}_3(\lambda, \mathbf{x}_{1-C})) = 1] - \Pr[D(\mathsf{D}_4(\lambda, \mathbf{x}_{1-C})) = 1]| \geq \frac{1}{p(\lambda)}$. The distributions D_3 and D_4 differ only in the method by which the output v is determined. In the former, it's determined by P_2 through the reconstructed output shares $(y_{j,i})_{j,i=0}^{1,m-1}$; in the latter, it's determined by the extracted input $\mathbf{x}_C$ and $\mathcal{F}$. The latter distribution differs from the former thus only in the event in which $\mathcal{A}$'s proofs pass, and its outputs $\left(\langle y_{j,i}\rangle_C, s_{j,i,C}\right)_{j,i=0}^{1,m-1}$ decommit to $(D_{j,i,C})_{j,i=0}^{1,m-1}$ (see the check of step 5. above), and yet these outputs *differ* from those which $\mathcal{S}$, having extracted the values $\left(\langle x_{j,i}\rangle_C, r_{j,i,C}\right)_{j,i=0}^{1,n/2-1}$ (see step 4. above), may freshly compute from (2) and (3). Because correctly computing (2) and (3) also yields an opening of $(D_{j,i,C})_{j,i=0}^{1,m-1}$, $\mathcal{S}$ thus obtains a binding violation in any instance in which D_3 and D_4 differ.

We now define an adversary $\mathcal{A}'$ attacking $\mathsf{Binding}_{\mathsf{Com},\mathcal{A}'}$. On advice $\mathbf{x}_{1-C}$ and input params, $\mathcal{A}'$ simulates an execution of D_3 on the honest input $\mathbf{x}_{1-C}$ in which params are used. If $\mathcal{A}$'s proofs pass, $\mathcal{A}'$ then runs the procedure of Lemma 2.29 on $\mathcal{A}$ (we may as well specialize this lemma to the case $\tau(\lambda) := 2$). Because D_3 and D_4 can differ only when $\mathcal{A}$'s proofs pass, our hypothesis on D implies *a fortiori* that $\mathcal{A}$'s proofs pass with probability at least $\frac{1}{p(\lambda)}$. Because $\frac{1}{p(\lambda)} \geq \frac{7 \cdot Q(\lambda)}{2^\lambda}$ (for large λ), the lemma's hypothesis holds. Moreover, $\mathcal{A}'$ must run $\mathcal{A}$ at most $32 \cdot Q(\lambda) \cdot p(\lambda)$ times, which is strictly polynomial in λ. Finally, $\mathcal{A}'$ extracts a witness from $\mathcal{A}$ with probability at least $\frac{1}{11}$. Because the distribution of openings extracted by $\mathcal{A}'$ (conditioned on its success) is identical to that output by $\mathcal{S}$, $\mathcal{A}'$ witness yields a binding violation, in strict polynomial time, with probability at least $\frac{1}{11 \cdot p(\lambda)}$. This completes the proof. □

Combining Lemmas 5.7, 5.8, 5.9, and 5.10, and applying the triangle inequality, completes the treatment of the case $C \in \{0, 1\}$.

We now suppose that $C = 2$. $\mathcal{S}$ operates as follows:

1. Since P_2 has no input, $\mathcal{S}$ immediately sends $\varnothing$ to $\mathcal{F}$, and obtains the output v in return.

2. $\mathcal{S}$ runs two instances of the simulator $\mathcal{S}$ of Theorem 3.36. That is, $\mathcal{S}$ samples the full array $(y_{j,i})_{j,i=0}^{1,m-1}$ as random $\mathbb{F}_p^*$-elements. For a randomly chosen index $i^* \in \{0, \ldots, m-1\}$, $\mathcal{S}$ overwrites $y_{v,i^*} := 0$. At this point, $\mathcal{S}$ generates random additive $\mathbb{F}_p$-sharings $y_{j,i} := \langle y_{j,i}\rangle_0 + \langle y_{j,i}\rangle_1$ for each $j \in \{0, 1\}$

and $i \in \{0, \ldots, m-1\}$. Finally, $\mathcal{S}$ samples $(s_{j,i,0}, s_{j,i,1})_{j,i=0}^{1,m-1}$ uniformly in $\mathbb{F}_p$. $\mathcal{S}$ commits $D_{j,i,\nu} := \mathsf{Com}(\langle y_{j,i}\rangle_\nu \, ; s_{j,i,\nu})$ for each $j \in \{0,1\}$, $i \in \{0, \ldots, m-1\}$, and $\nu \in \{0,1\}$. $\mathcal{S}$ records the simulated messages $\left(\langle y_{j,i}\rangle_0, s_{j,i,0}, D_{j,i,1}\right)_{j,i=0}^{1,m-1}$ and $\left(\langle y_{j,i}\rangle_1, s_{j,i,1}, D_{j,i,0}\right)_{j,i=0}^{1,m-1}$ sent by P_0 and P_1, respectively.

3. $\mathcal{S}$ gives these messages internally to $\mathcal{A}$. When $\mathcal{A}$ outputs v and π, $\mathcal{S}$ independently verifies $\mathsf{OneOutOfMany.Verify}\left((D_{v,i})_{i=0}^{m-1}\right)$ (this latter v output by $\mathcal{A}$ could differ, *a priori*, from that already output by $\mathcal{F}$). If the verification fails, then $\mathcal{S}$ sends $\perp$ to $\mathcal{F}$ and aborts (this is $\mathcal{S}$'s chance to "break fairness"). Otherwise, $\mathcal{S}$ instructs $\mathcal{F}$ to continue and to report its output to P_0 and P_1.

We now claim that $\mathsf{Real}_{\Pi,\mathcal{A}}\left(\lambda, 2, (\mathbf{x}_\nu)_{\nu\in\{0,1\}}\right)$ and $\mathsf{Ideal}_{\mathcal{F},\mathcal{S}}\left(\lambda, 2, (\mathbf{x}_\nu)_{\nu\in\{0,1\}}\right)$ are indistinguishable. We define a sequence of hybrid distributions.

D_0: Corresponds to $\mathsf{Real}_{\Pi,\mathcal{A}}\left(\lambda, 2, (\mathbf{x}_\nu)_{\nu\in\{0,1\}}\right)$, i.e., the pair $(V_2, (v_\nu)_{\nu\in\{0,1\}})$.

D_1: Same as D_0, except that instead of the computational shared secret key ξ, P_0 and P_1 are given a truly random string $\hat{\xi} \in \{0,1\}^\lambda$.

D_2: Same as D_1, except instead of obtaining them through $\hat{\xi}$ and G, P_0 and P_1 are given truly random quantities $(\alpha_{j,i})_{j,i=0}^{1,m-1}$ in $\mathbb{F}_p^*$, $(\rho_j)_{j=0}^1$ in $\langle(0,1,\ldots,m-1)\rangle \subset \mathbf{S}_m$, and $(\beta_{j,i,\nu})_{j,i,\nu=0}^{1,n/2-1,1}$ in $\mathbb{F}_p$.

D_3: Exactly $\mathsf{Ideal}_{\mathcal{F},\mathcal{S}}\left(\lambda, 2, (\mathbf{x}_\nu)_{\nu\in\{0,1\}}\right)$; i.e., $(V_2, (v)_{v\in\{0,1\}})$, where V_2 is the simulated view output by $\mathcal{S}$.

Lemma 5.11. *If Ξ is secure, then the distributions D_0 and D_1 are computationally indistinguishable.*

Proof. If D_0 and D_1 were distinguishable, then there would be a distinguisher D and a polynomial $p(\lambda)$ for which, for a sequence of triples $(\lambda, \mathbf{x}_0, \mathbf{x}_1)$ in which infinitely many distinct values λ are represented, $|\Pr[D(\mathsf{D}_0(\lambda, \mathbf{x}_0, \mathbf{x}_1)) = 1] - \Pr[D(\mathsf{D}_1(\lambda, \mathbf{x}_0, \mathbf{x}_1)) = 1]| \geq \frac{1}{p(\lambda)}$. Without loss of generality—and after possibly flipping D's output bit—we may assume that $\Pr[D(\mathsf{D}_1(\lambda, \mathbf{x}_0, \mathbf{x}_1)) = 1] - \Pr[D(\mathsf{D}_0(\lambda, \mathbf{x}_0, \mathbf{x}_1)) = 1] \geq \frac{1}{p(\lambda)}$ for infinitely many λ and appropriate $(\mathbf{x}_0, \mathbf{x}_1)$. We define a nonuniform adversary $\mathcal{A}$ attacking the key-exchange experiment $\mathsf{KE}_{\Xi,\mathcal{A}}$. Upon receiving trans and $\hat{\xi}$, using the "advice" $(\mathbf{x}_0, \mathbf{x}_1)$, $\mathcal{A}$ simulates an execution of Protocol 5.5 on inputs $\mathbf{x}_0$ and $\mathbf{x}_1$, in which the challenge $\hat{\xi}$ is used in place of P_0 and P_1's shared secret string ξ. In this way, $\mathcal{A}$ obtains an output view V_2. $\mathcal{A}$ then runs D on V_2, and outputs whatever D outputs. If $\hat{\xi}$ is a computational shared secret, then the view V_2 is distributed exactly as in $\mathsf{D}_0(\lambda, \mathbf{x}_0, \mathbf{x}_1)$. If $\hat{\xi}$ is a random λ-bit string, then the view V_2 is distributed exactly as in $\mathsf{D}_1(\lambda, \mathbf{x}_0, \mathbf{x}_1)$. Writing b for the key-exchange experimenter's hidden bit, we thus have:

$$\begin{aligned}
\Pr[\mathsf{KE}_{\Xi,\mathcal{A}}(\lambda) = 1] &= \frac{1}{2} \cdot \Pr[\mathsf{out}_\mathcal{A}(\mathsf{KE}_{\Xi,\mathcal{A}}(\lambda)) = 0 \mid b = 0] + \frac{1}{2} \cdot \Pr[\mathsf{out}_\mathcal{A}(\mathsf{KE}_{\Xi,\mathcal{A}}(\lambda)) = 1 \mid b = 1] \\
&= \frac{1}{2} \cdot \left(1 - \Pr_{V_2 \leftarrow \mathsf{D}_0(\lambda, \mathbf{x}_0, \mathbf{x}_1)}[D(V_2) = 1] + \Pr_{V_2 \leftarrow \mathsf{D}_1(\lambda, \mathbf{x}_0, \mathbf{x}_1)}[D(V_2) = 1]\right) \\
&= \frac{1}{2} + \frac{1}{2} \cdot \left(\Pr_{V_2 \leftarrow \mathsf{D}_1(\lambda, \mathbf{x}_0, \mathbf{x}_1)}[D(V_2) = 1] - \Pr_{V_2 \leftarrow \mathsf{D}_0(\lambda, \mathbf{x}_0, \mathbf{x}_1)}[D(V_2) = 1]\right).
\end{aligned}$$

Our assumption on D would now show that $\Pr[\mathsf{KE}_{\Xi,\mathcal{A}}(\lambda) = 1] - \frac{1}{2} \geq \frac{1}{2 \cdot p(\lambda)}$ for infinitely many λ, which would contradict the security of Ξ. □

Lemma 5.12. *If G is pseudorandom, then the distributions D_1 and D_2 are computationally indistinguishable.*

Proof. If D_1 and D_2 were distinguishable, then there would be a distinguisher D, a polynomial $p(\lambda)$, and an infinite sequence of triples $(\lambda, \mathbf{x}_0, \mathbf{x}_1)$ for which $|\Pr[D(\mathsf{D}_1(\lambda, \mathbf{x}_0, \mathbf{x}_1)) = 1] - \Pr[D(\mathsf{D}_2(\lambda, \mathbf{x}_0, \mathbf{x}_1)) = 1]| \geq \frac{1}{p(\lambda)}$. We define a PPT distinguisher D' attacking G in the following way. On a $l(\lambda)$-bit input string t and advice $(\mathbf{x}_0, \mathbf{x}_1)$, D' simulates an execution of Protocol 5.5 on $\mathbf{x}_0$ and $\mathbf{x}_1$, in which the string t is used by P_0 and P_1 to construct $(\alpha_i)_{i=0}^{m-1}$ and ρ. D' then runs D on the resulting view V_2, and outputs whatever D outputs. If

$t = G(s)$ for a uniform seed $s \leftarrow \{0,1\}^\lambda$, then the view V_2 is distributed exactly as in $\mathsf{D}_1(\lambda, \mathbf{x}_0, \mathbf{x}_1)$. If $t = r$ for a uniform string $r \leftarrow \{0,1\}^{l(\lambda)}$, then V_2 is distributed exactly as in $\mathsf{D}_2(\lambda, \mathbf{x}_0, \mathbf{x}_1)$. It follows that:

$$\left| \Pr_{s \leftarrow \{0,1\}^\lambda}[D'(G(s)) = 1] - \Pr_{r \leftarrow \{0,1\}^{l(\lambda)}}[D'(r) = 1] \right| = \left| \Pr_{V_2 \leftarrow \mathsf{D}_1(\lambda, \mathbf{x}_0, \mathbf{x}_1)}[D(V_2) = 1] - \Pr_{V_2 \leftarrow \mathsf{D}_2(\lambda, \mathbf{x}_0, \mathbf{x}_1)}[D(V_2) = 1] \right|.$$

Our hypothesis on D would now show that G is not pseudorandom, contradicting the lemma's hypothesis. □

Lemma 5.13. *If* $(\mathsf{Gen}, \mathsf{Com})$ *is binding, then* D_2 *and* D_3 *are computationally indistinguishable.*

Proof. We first argue that the openings and commitments $\left(\langle y_{j,i} \rangle_0, s_{j,i,0}, D_{j,i,1}\right)_{j,i=0}^{1,m-1}$ and $\left(\langle y_{j,i} \rangle_1, s_{j,i,1}, D_{j,i,0}\right)_{j,i=0}^{1,m-1}$ sent by P_0 and P_1 in D_2 exactly match those simulated by $\mathcal{S}$ in D_3. Indeed, for each choice of inputs $\mathbf{x}_0$ and $\mathbf{x}_1$, in D_2, the initial shares $\left(\langle x_{\nu,i} \rangle_k\right)_{i=0}^{n/2-1}$, for each $\nu \in \{0,1\}$ and $k \in \{0,1\}$, are uniformly random, subject to the condition $\langle x_{\nu,i} \rangle_0 + \langle x_{\nu,i} \rangle_1 = x_{\nu,i}$ for each $\nu \in \{0,1\}$ and $i \in \{0, \ldots, \frac{n}{2} - 1\}$. It follows that the final randomized output shares $\langle y_{j,i} \rangle_\nu$ are thus also uniformly random, subject to the condition $y_{j,i} = \langle y_{j,i} \rangle_0 + \langle y_{j,i} \rangle_1$, where $(y_{j,i})_{j,i=0}^{1,m-1} = \left(\alpha_{j,i} \cdot H_{j,\rho(i)}(x_0, \ldots, x_{n-1})\right)_{j,i=0}^{1,m-1}$. The correctness of the distribution of the shares $\langle y_{j,i} \rangle_\nu$ thus follows from the perfect privacy property established by Theorem 3.36. The randomnesses $s_{j,i,0}$ and $s_{j,i,1}$ are independently random in both distributions.

The distributions D_2 and D_3 thus differ only perhaps in their outputs. In the former, the parties' outputs are derived from P_2's final message v; in the latter, P_0 and P_1 receive v directly from $\mathcal{F}$. These distributions thus differ only in the case that $\mathcal{A}$ outputs a successful one-out-of-many proof on the "wrong" v. For notational consistency, we write $v := \mathcal{F}(\mathbf{x}_0, \mathbf{x}_1)$ in what follows (as opposed to $\mathcal{A}$'s final output).

For contradiction, we fix a distinguisher D, a polynomial $p(\lambda)$, and an infinite sequence of triples $(\lambda, \mathbf{x}_0, \mathbf{x}_1)$ for which $|\Pr[D(\mathsf{D}_2(\lambda, \mathbf{x}_0, \mathbf{x}_1)) = 1] - \Pr[D(\mathsf{D}_3(\lambda, \mathbf{x}_0, \mathbf{x}_1)) = 1]| \geq \frac{1}{p(\lambda)}$. We define a nonuniform adversary $\mathcal{A}'$ attacking $\mathsf{Binding}_{\mathsf{Com}, \mathcal{A}'}$ in the following way. On advice $(\mathbf{x}_0, \mathbf{x}_1)$, $\mathcal{A}'$ simulates an execution of D_2 on the inputs $(\mathbf{x}_0, \mathbf{x}_1)$. If $\mathcal{A}$ outputs a successful one-out-of-many proof on the "wrong" statement $(D_{1-v,i})_{i=0}^{m-1}$—this happens with probability at least $\frac{1}{p(\lambda)}$, by hypothesis on D—then $\mathcal{A}'$ runs the machine M of Lemma 2.29 above (whose hypothesis necessarily holds for large λ) on $\mathcal{A}$. In this way, in strict polynomial time and with probability at least $\frac{1}{11 \cdot p(\lambda)}$, $\mathcal{A}'$ obtains a witness $(y_{1-v,i}, s_{1-v,i})_{i=0}^{m-1}$ for $(D_{1-v,i})_{i=0}^{m-1}$, where in particular $y_{1-v,i^*} = 0$ for some $i^* \in \{0, \ldots, m-1\}$. The quantity y_{1-v,i^*} *already* reconstructed from P_0 and P_1 is necessarily nonzero; the second opening $(y_{1-v,i^*}, s_{1-v,i^*})$ with $y_{1-v,i^*} = 0$ thus immediately yields a binding violation for D_{1-v,i^*}. $\mathcal{A}'$ outputs this violation, and wins with probability at least $\frac{1}{11 \cdot p(\lambda)}$. □

Combining Lemmas 5.11, 5.12, and 5.13 completes the proof of the case $C = 2$ and of the theorem. □

Remark 5.14. Though we refrain from proving it here, we note that Protocol 5.5 remains secure *even* if P_2 sees *all* the initial data exchanged between P_0 and P_1 in step 1. above. This fact strengthens the protocol's security. Indeed, it allows P_0 and P_1 to communicate with each other *through* P_2—as opposed to directly—if they prefer to do so. To thwart person-in-the-middle attacks, it is necessary only that P_0 and P_1 sign their initial commitments A_0 and A_1 using known (i.e., authenticated) public keys.

Theorem 5.15. *Suppose that* $\{f_n\}_{n \in \mathbb{N}}$ *is efficiently computable by disjoint hyperplanes, and in particular admits coverings* $f_n^{-1}(0) = \bigsqcup_{i=0}^{m-1} H_{0,i} \cap \{0,1\}^n$ *and* $f_n^{-1}(1) = \bigsqcup_{i=0}^{m-1} H_{1,i} \cap \{0,1\}^n$ *for* $m = \mathsf{poly}(n)$. *Then Protocol 5.5 above evaluates* f_n *in* $O(n \cdot m)$ *time, using* $O(n + m)$ *communication, and in three rounds.*

Proof. As in Protocol 5.1, the evaluations (2), (3), and (4) take a total of $O(n \cdot m)$ time to evaluate for each party P_ν, $\nu \in \{0,1\}$. When P_2 receives all output shares and commitments, it must perform $O(m)$ work to reconstruct and check them. Finally, P_2 must perform $O(m \cdot \log m)$ work to generate the one-out-of-many proof; P_0 and P_1 must perform $O(m)$ work to verify it (see [GK15, p. 268]). By hypothesis on $\{f_n\}_{n \in \mathbb{N}}$, we have that m is polynomial in n, and hence that P_0 and P_1's $O(n \cdot m)$ to evaluate the hyperplanes dominates P_2's $O(m \cdot \log m)$ effort in generating the proof (and of course their own $O(m)$ effort in verifying it).

Also as before, each party P_ν must send $O(n)$ bits' worth of shares *and* commitments to $P_{1-\nu}$, as well as $O(m)$ bits' worth of output shares and commitments to P_2. The one-out-of-many proof sent back to P_0 and P_1 by P_2 requires only $O(\log m)$ bits.

Again as in Protocol 5.1 (the main online phase of) the protocol clearly takes 3 rounds. □

5.3 Implementation

In this subsection, we describe an implementation of the Protocol 5.5, and report on its performance. For the sake of example, we specialize to the integer-comparison function of Example 3.22. Below, we describe further applications to a secure "volume-matching" utility.

For our implementation, we use a slight variant of Example 3.22 in which both $\mathbf{x}_0 \leq \mathbf{x}_1$ and $\mathbf{x}_0 \geq \mathbf{x}_1$ are computed, where $\mathbf{x}_0$ and $\mathbf{x}_1$ are $\frac{n}{2}$-bit (unsigned, little-endian) arguments; in this setting, the two coverings are not disjoint, but rather intersect at the locus $\mathbf{x}_0 = \mathbf{x}_1$. (We describe our rationale for this below.) We use the concrete hyperplane covering already discussed in Example 3.22. Each set $\mathbf{x}_0 \leq \mathbf{x}_1$ and $\mathbf{x}_0 \geq \mathbf{x}_1$ requires $\frac{n}{2}+1$ hyperplanes to compute (with one in common). We also use the subexpression-sharing scheme described in Example 3.22, so that both coverings can be evaluated in $O(n)$ total time.

We implement the case $n = 62$. In this setting, the number $\frac{n}{2} + 1 = 32$ of hyperplanes required by *each* covering is a power of 2, suitable for use in one-out-of-many proofs. The individual arguments $\mathbf{x}_0$ and $\mathbf{x}_1$ are thus 31-bit unsigned integers. We use the NIST P-256 prime $p = 2^{256} - 2^{224} + 2^{192} + 2^{96} - 1$. We take as $(\mathsf{Gen}, \mathsf{Com})$ the Pedersen commitment scheme over the NIST P-256 curve, whose order is p [Inf13, D.2.3].

Our implementation is targeted towards real-world use. We have separate dedicated components for the "client" players P_0 and P_1 and for the "server" player P_2. For the purposes of practical convenience and portability, our client module is entirely browser-based, written in JavaScript. Its cryptographically intensive components are written in efficient, side-channel-resistant C, compiled using Emscripten into WebAssembly (which also runs natively in the browser). Our server is written in Python, and also executes its cryptographically intensive code in C. Both components are multi-threaded—using WebWorkers on the client side and a thread pool on the server's—and can execute arbitrarily many concurrent instances of the protocol in parallel (i.e., constrained only by hardware). All players communicate by sending binary data on WebSockets (all commitments, proofs, and messages are serialized).

We benchmark our protocol by executing multiple total instances of the protocol, with parallelism (see the left-most column below). We run on commodity hardware throughout. Specifically, each of our clients runs on an Intel Core i7 processor, with 6 cores, each 2.6Ghz. (One is a Mac; the other Windows.) Our server runs in a Linux AWS instance of type `c5.4xlarge`, with 16 vCPUs. Our benchmarks include communication time; all communication takes place over a WAN.

The field "Wall Time" is self-explanatory. The field "Server CPU" refers to the *cumulative* time spent by all of the server's CPU cores during its execution (and may be higher than the wall time). The fields "Client Exchanged" and "Server Exchanged" reflect the total number of bytes—either sent *or* received—which travel through an individual client's and the server's WebSockets, respectively. The field "Total Exchanged" gives the total number of bytes which flow through *all* (combined) WebSockets.

Table 4: Performance characteristics of Protocol 5.5 implementation.

Executions	Wall Time	Server CPU	Client CPU	Server Exch'd	Client Exch'd	Total Exch'd
	(ms)	(ms)	(ms / worker)	(kilobytes)	(kilobytes)	(kilobytes)
1	573	565	322	13	25	64
2	632	737	291	27	50	127
4	841	987	268	53	101	255
8	1,082	1,528	377	107	202	510
16	1,958	3,025	578	213	403	1,020
32	3,497	5,299	1,378	427	806	2,040
64	6,179	10,345	2,602	854	1,613	4,097
(asymp.) 1	$O(n \cdot m)$	$O(m \cdot \log m)$	$O(n \cdot m)$	$O(m)$	$O(n+m)$	$O(n+m)$

We note the effect of parallelism on our measurements. For example, when 64 executions are conducted, each *individual* execution takes about 97 milliseconds, amortized.

To exhibit the practical utility of our paradigm, we now exhibit a concrete application, in which our protocol's homomorphic commitment-consistency property plays an essential role. We consider the problem of *volume matching*, sometimes called *midpoint matching* in the economic literature (see for example Zhu

[Zhu13]). A volume matching service accumulates—throughout its initial *registration* phase—a plurality of *orders*, each specifying a security, a direction (either "long" or "short") and a quantity (a non-negative integer). When matching begins, orders pertaining to the same security and of opposite direction are matched. After each particular match, the service decrements both orders' volumes by the matched amount, and dequeues whichever among the two orders is empty (necessarily at least one will be).

Abstractly, volume-matching is described by the following functionality.

FUNCTIONALITY 5.16 ($\mathcal{F}_{\text{match}}$—volume-matching functionality)**.**
Upon initialization, $\mathcal{F}_{\text{match}}$ initializes two empty first-in, first-out queues, $\mathcal{Q}_0$ and $\mathcal{Q}_1$.

- During the registration phase, $\mathcal{F}_{\text{match}}$ exposes the following function:

```
1: procedure $\mathcal{F}_{\text{match}}$.Register($d, \mathbf{x}$)        ▷ Direction $d \in \{0,1\}$, volume $\mathbf{x} \in \{0,\ldots,2^{n/2}-1\}$
2:     Write $P_\nu$ for the invoking party.
3:     $\mathcal{Q}_d$.Enqueue({party : $P_\nu$, volume : $\mathbf{x}$}).
```

- When the processing phase begins, $\mathcal{F}_{\text{match}}$ executes the following function:

```
1: procedure $\mathcal{F}_{\text{match}}$.Process()
2:     while not $\mathcal{Q}_0$.Empty() and not $\mathcal{Q}_1$.Empty() do
3:         $\mathbf{x} := \min(\mathcal{Q}_0$.Front().volume, $\mathcal{Q}_1$.Front().volume$)$
4:         Output $\mathbf{x}$, $\mathcal{Q}_0$.Front().party, and $\mathcal{Q}_1$.Front().party
5:         for $d \in \{0,1\}$ do
6:             $\mathcal{Q}_d$.Front().volume $-= \mathbf{x}$
7:             if $\mathcal{Q}_d$.Front().volume $= 0$ then $\mathcal{Q}_d$.Dequeue()
```

Our matching engine simultaneously maintains many independent instances of Functionality 5.16 (one for each security). During its registration phase, the engine solicits homomorphic (i.e., Pedersen) commitments. During the processing phase, our engine orchestrates an instance of Protocol 5.5 (with f the comparison function discussed above) for *each* iteration of *each* instance of Functionality 5.16's main **while** loop. The engine uses the output values $\mathbf{x}_0 \leq \mathbf{x}_1$ and $\mathbf{x}_0 \geq \mathbf{x}_1$ to control its inner queues. We repeat that each player handles *all securities* on a single process, which coordinates all scheduling and communication; the cryptographic work is sped up using multi-threading (thus our implementation is *not* "embarassingly parallel").

$\mathcal{F}_{\text{match}}$'s commitment-consistency establishes an important "fairness" guarantee, whereby client's can't register early, obtain desirable positions in the queues, and then only *later* decide which secret inputs to use. Finally, the commitments' *homomorphic* property allows the engine to appropriately decrement registrations between matches. Our engine is being deployed live in production for real use at a large financial institution.

References

[ABT18] Benny Applebaum, Zvika Brakerski, and Rotem Tsabary. Perfect secure computation in two rounds. In Amos Beimel and Stefan Dziembowski, editors, *Theory of Cryptography*, pages 152–174, Cham, 2018. Springer International Publishing.

[AF93] Noga Alon and Zoltán Füredi. Covering the cube by affine hyperplanes. *European Journal of Combinatorics*, 14(2):79–83, 1993.

[AL17] Gilad Asharov and Yehuda Lindell. A full proof of the BGW protocol for perfectly secure multiparty computation. *Journal of Cryptology*, 30(1):58–151, 2017.

[BB66] R. C. Bose and R. C. Burton. A characterization of flat spaces in a finite geometry and the uniqueness of the Hamming and the MacDonald codes. *Journal of Combinatorial Theory*, 1(1):96–104, 1966.

[BBB+18] Benedikt Bünz, Jonathan Bootle, Dan Boneh, Andrew Poelstra, Pieter Wuille, and Greg Maxwell. Bulletproofs: Short proofs for confidential transactions and more. In *IEEE Symposium on Security and Privacy*, volume 1, 2018. Full version.

[BHMSV84] Robert K. Brayton, Gary D. Hachtel, Curtis T. McMullen, and Alberto L. Sangiovanni-Vincentelli. *Logic Minimization Algorithms for VLSI Synthesis.* The Kluwer International Series in Engineering and Computer Science. Kluwer Academic Publishers, 1984.

[BOGW88] 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*, pages 1–10, 1988.

[Cam95] Peter L. Cameron. *Handbook of Combinatorics*, volume I, chapter Finite Geometries. The MIT Press, 1995.

[CDN15] Ronald Cramer, Ivan Bjerre Damgård, and Jesper Buus Nielsen. *Secure Multiparty Computation and Secret Sharing.* Cambridge University Press, Cambridge, 2015.

[CHLL97] Gérard Cohen, Iiro Honkala, Simon Litsyn, and Antoine Lobstein. *Covering Codes*, volume 54 of *North-Holland Mathematical Library.* North-Holland, 1997.

[Cir01] Valentina Ciriani. Logic minimization using exclusive OR gates. In *Proceedings of the 38th annual Design Automation Conference*, pages 115–120. Association for Computing Machinery, June 2001.

[Cir03] V. Ciriani. Synthesis of SPP three-level logic networks using affine spaces. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 22(10):1310–1323, 2003.

[CK08] F. Cazals and C. Karande. A note on the problem of reporting maximal cliques. *Theoretical Computer Science*, 407(1):564–568, 2008.

[CLRS09] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms.* The MIT Press, third edition, 2009.

[Coh74] P.M. Cohn. *Algebra*, volume 1. John Wiley & Sons, 1974.

[DFK$^+$06] Ivan Damgård, Matthias Fitzi, Eike Kiltz, Jesper Buus Nielsen, and Tomas Toft. Unconditionally secure constant-rounds multi-party computation for equality, comparison, bits and exponentiation. In Shai Halevi and Tal Rabin, editors, *Theory of Cryptography*, pages 285–304, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[Dud14] Adrian W. Dudek. On the Riemann hypothesis and the difference between primes. *International Journal of Number Theory*, 11(03):771–778, 01 2014.

[FPY18] Tore K. Frederiksen, Benny Pinkas, and Avishay Yanai. Committed mpc. In Michel Abdalla and Ricardo Dahab, editors, *Public-Key Cryptography – PKC 2018*, pages 587–619, Cham, 2018. Springer International Publishing.

[GK15] Jens Groth and Markulf Kohlweiss. One-out-of-many proofs: Or how to leak a secret and spend a coin. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology – EUROCRYPT 2015*, volume 9057 of *Lecture Notes in Computer Science*, pages 253–280. Springer Berlin Heidelberg, 2015.

[GRR98] Rosario Gennaro, Michael O. Rabin, and Tal Rabin. Simplified VSS and fast-track multiparty computations with applications to threshold cryptography. In *Proceedings of the Seventeenth Annual ACM Symposium on Principles of Distributed Computing*, pages 101–111, 1998.

[Har77] Robin Hartshorne. *Algebraic Geometry*, volume 52 of *Graduate Texts in Mathematics.* Springer, 1977.

[HBG84] D. R. Heath-Brown and D. A. Goldston. A note on the differences between consecutive primes. *Mathematische Annalen*, 266(3):317–320, 1984.

[HL10] Carmit Hazay and Yehuda Lindell. *Efficient Secure Two-Party Protocols.* Information Security and Cryptography. Springer, 2010.

[HT15] J. W. P. Hirschfeld and J. A. Thas. Open problems in finite projective spaces. *Finite Fields and Their Applications*, 32:44–81, 2015.

[IK00] Y. Ishai and E. Kushilevitz. Randomizing polynomials: A new representation with applications to round-efficient secure computation. In *Proceedings 41st Annual Symposium on Foundations of Computer Science*, pages 294–304, 2000.

[IK02] Yuval Ishai and Eyal Kushilevitz. Perfect constant-round secure computation via perfect randomizing polynomials. In Peter Widmayer, Stephan Eidenbenz, Francisco Triguero, Rafael Morales, Ricardo Conejo, and Matthew Hennessy, editors, *Automata, Languages and Programming*, pages 244–256, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg.

[IKM$^+$13] Yuval Ishai, Eyal Kushilevitz, Sigurd Meldgaard, Claudio Orlandi, and Anat Paskin-Cherniavsky. On the power of correlated randomness in secure computation. In Amit Sahai, editor, *Theory of Cryptography*, pages 600–620, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[Inf13] Information Technology Laboratory. Digital signature standard (DSS). Technical Report FIPS 186-4, National Institute of Standards and Technology, July 2013.

[KL97] Ilia Krasikov and Simon Litsyn. Linear programming bounds for codes of small size. *European Journal of Combinatorics*, 18(6):647–656, 1997.

[KL21] Jonathan Katz and Yehuda Lindell. *Introduction to Modern Cryptography.* CRC Press, third edition, 2021.

[Knu11] Donald E. Knuth. *The Art of Computer Programming*, volume 4A: Combinatorial Algorithms. Addison–Wesley, 2011.

[Lin17] Yehuda Lindell. *Tutorials on the Foundations of Cryptography: Dedicated to Oded Goldreich*, chapter How to Simulate It – A Tutorial on the Simulation Proof Technique, pages 277–346. Information Security and Cryptography. Springer International Publishing, 2017.

[LP99] Fabrizio Luccio and Linda Pagli. On a new boolean function with applications. *IEEE Transactions on Computers*, 48(3):296–310, March 1999.

[Mey00] Carl D. Meyer. *Matrix Analysis and Applied Linear Algebra.* SIAM, 2000.

[MJ56] E.J. McCluskey Jr. Minimization of boolean functions. *Bell System Technical Journal*, 35(6):1417–1444, 1956.

[Nag52] Jitsuro Nagura. On the interval containing at least one prime number. *Proceedings of the Japan Academy*, 28(4):177–181, 1952.

[Nus82] H. Nussbaumer. *Fast Fourier Transform and Convolution Algorithms.* Springer-Verlag, 1982.

[Odl81] A. M. Odlyzko. On the ranks of some (0, 1)-matrices with constant row sums. *Journal of the Australian Mathematical Society (Series A)*, 31(2):193–201, 1981.

[PS00] David Pointcheval and Jacques Stern. Security arguments for digital signatures and blind signatures. *Journal of Cryptology*, 13(3):361–396, 2000.

[Tie80] Aimo Tietäväinen. Bounds for binary codes just outside the Plotkin range. *Information and Control*, 47(2):85–93, 1980.

[vzGG13] Joachim von zur Gathen and Jürgen Gerhard. *Modern Computer Algebra.* Cambridge University Press, third edition, 2013.

[Weg87] Ingo Wegener. *The Complexity of Boolean Functions.* Wiley–Teubner Series in Computer Science. Wiley, 1987.

[WGC19] Sameer Wagh, Divya Gupta, and Nishanth Chandran. SecureNN: 3-party secure computation for neural network training. *Proceedings on Privacy Enhancing Technologies*, 2019(3):26–49, 2019.

[Zhu13] Haoxiang Zhu. Do dark pools harm price discovery? *The Review of Financial Studies*, 27(3):747–789, December 2013.

[Zob70] Albert L. Zobrist. A new hashing method with application for game playing. Technical Report 88, The University of Wisconsin, April 1970.

LINEAR ALGEBRA AND THE UNIT CUBE

and related problems involving extension of affine flats

Benjamin E. Diamond
J.P. Morgan AI Research
benjamin.e.diamond@jpmchase.com

Jason Long
J.P. Morgan AI Research
jason.x.long@jpmchase.com

A Extension of Flats

In this section, we theoretically treat the following challenging question:

Question A.1. *Fix a natural number n. Does there exist an n-bit prime p such that each proper affine flat $K \subset \mathbb{F}_p^n$ admits an affine hyperplane $H \subset \mathbb{F}_p^n$ for which $K \cap \{0,1\}^n = H \cap \{0,1\}^n$?*

The condition of this question trivially holds when $p \geq 2^n$, as we demonstrated at the beginning of Subsection 4.4 above. In fact, the thrust of that argument is captured by a classical result of Bose and Burton [BB66, Thm. 2], which studies *blocking sets* in $\mathbb{PF}_p^{n-k-l}$. Their main theorem states that any set $\mathscr{B} \subset \mathbb{PF}_p^{n-k-l}$ which intersects *every* projective hyperplane must satisfy $|\mathscr{B}| \geq p+1$. Applying this result directly to the image of $\{0,1\}^n$ in $\mathbb{F}_p^{n-k}$—we recall that $|A(\{0,1\}^n)| \leq 2^n < p+1$—we see immediately that the required hyperplane must exist (under the same hypothesis).

Broadly speaking, we spend this section studying the existence of primes $p < 2^n$ which nonetheless satisfy the condition of Question A.1. This is an important theoretical question, in light of Definition 3.3. (If larger primes p were allowed, then the bound of Theorem 3.4 would be weakened.) This task presents serious challenges. For technical reasons (discussed below), we treat separately the cases of low-dimensional and high-dimensional flats K. Our primary difficulty resides in answering the following question:

Question A.2. *Fix a natural number n, a prime p, and a k-dimensional affine flat $K \subset \mathbb{F}_p^n$ generated by cube elements, with quotient map $A : \mathbb{F}_p^n \to \mathbb{F}_p^{n-k}$, say. How can we bound from above the size of $A(\{0,1\}^n)$?*

Our main result answering this question is given by Theorem A.14 below. This is an interesting combinatorial result in its own right. We do not know whether a tighter bound can be attained. In fact, (an appropriate analogue of) Theorem A.14 holds when $\mathbb{F}_p$ is replaced by $\mathbb{Q}$ or $\mathbb{R}$.

A.1 Lower-dimensional flats

Bose and Burton observe [BB66, Thm. 2] that their lower-bound $|\mathscr{B}| \geq p+1$ is tight, attained for example when $\mathscr{B}$ is a projective line. This fact—at first—may appear to bode poorly for our efforts to reduce the requirement on p in Question A.1. In fact, the further study of blocking sets, by Blockuis and Heim—surveyed in Hirschfeld and Thas [HT15, §9]—demonstrates that, when one restricts to so-called *nontrivial* blocking sets, which, by definition, fail to contain a projective line, one obtains stronger lower bounds. Theorem A.4 below exploits this fact, and settles the case of flats of dimension $k < n-2$. We adopt the terminology of [HT15, Def. 9.1] in this subsection.

Theorem A.3 (Blockhuis–Heim)**.** *Any nontrivial blocking set $\mathscr{B} \subset \mathbb{PF}_p^{n-k-1}$ satisfies $|\mathscr{B}| \geq 3 \cdot \frac{p+1}{2}$.*

Proof. This is a combination of the results [HT15, Thm. 9.6 (i) and Thm. 9.7 (ii)] □

Corollary A.4. *Suppose that $n > 2$ and $3 \cdot \frac{p+1}{2} \geq 2^n$. Then any k-dimensional affine flat $K \subset \mathbb{F}_p^n$ for which $k < n-2$ admits an affine hyperplane $H \subset \mathbb{F}_p^n$ for which $K \cap \{0,1\}^n = H \cap \{0,1\}^n$.*

Proof. We assume as usual that K contains the origin, and fix a quotient map $A : \mathbb{F}_p^n \to \mathbb{F}_p^{n-k}$. It suffices to show that $A(\{0,1\}^n - K)$—or more precisely, its projectivization in $\mathbb{P}\mathbb{F}_p^{n-k-1}$—is not a blocking set.

By hypothesis on p, $|A(\{0,1\}^n - K)| \leq 2^n - 1 < 3 \cdot \frac{p+1}{2}$; Theorem A.3 thus shows that $A(\{0,1\}^n - K)$ cannot be a *nontrivial* blocking set. We must show therefore only that $A(\{0,1\}^n - K)$ is not a trivial blocking set, or that, in other words, it does not contain a projective line.

We consider an arbitrary two-dimensional linear subspace $L \subset \mathbb{F}_p^{n-k}$ (containing the origin). The pullback of L along the quotient map A yields a $k+2$-dimensional linear subspace of $\mathbb{F}_p^n$, which, by hypothesis on k, is proper (of dimension less than n). The intersection $\{0,1\} \cap A^{-1}(L)$ can thus contain at most 2^{n-1} points, by Lemma 2.5. The same is thus true of this intersection's image under A. We see that necessarily $|A(\{0,1\}^n - K) \cap L| \leq |A(\{0,1\}^n) \cap L| \leq 2^{n-1} < p+1$ for any two-dimensional subspace $L \subset \mathbb{F}_p^{n-k}$; this implies that $A(\{0,1\}^n - K)$ cannot contain a projective line, and completes the proof. $\square$

The following result answers Question A.1 affirmatively for flats whose dimension is less than $n-2$:

Corollary A.5. *For each $n > 2$, there exists an odd prime $p < 2^n$ such that each k-dimensional affine flat $K \subset \mathbb{F}_p^n$ of dimension $k < n-2$ admits a hyperplane extension $H \subset \mathbb{F}_p^n$ such that $K \cap \{0,1\}^n = H \cap \{0,1\}^n$.*

Proof. By Theorem A.4, it suffices to produce a prime $p < 2^n$ which moreover satisfies $3 \cdot \frac{p+1}{2} \geq 2^n$. An old result of Nagura [Nag52, p. 180], which strengthens Bertrand's postulate, states that the interval $(x, \frac{3}{2} \cdot x)$ contains a prime p for each $x \geq 8$. This result implies *a fortiori* that the interval $\{\lceil \frac{2^n}{3} - 1 \rceil, \ldots, 2^n\}$ contains an odd prime p for each $n \geq 2$. $\square$

The above argument fails for subspaces $K \subset \mathbb{F}_p^n$ of dimension $n-2$; in this setting, $A^{-1}(L)$ is not a proper subspace, and Lemma 2.5 fails to constrain $(\{0,1\} - K) \cap A^{-1}(L)$.

A.2 Higher-dimensional flats

We develop a different strategy suitable for higher-dimensional flats. Our idea is to show that any subspace $K \subset \mathbb{F}_p^n$ of large dimension must contain "many" cube elements of "low" Hamming weight. By studying the effects of these vectors on the quotient-by-K map, we may show that $A : \mathbb{F}_p^n \to \mathbb{F}_p^{n-k}$ exhibits sufficient "collapsing" behavior on the cube as to reduce the size of the image $A(\{0,1\}^n)$. We begin with a handful of definitions and introductory lemmas.

Definition A.6. We will call *nonzero* $\{-1,0,1\}$-vectors (with components taken in $\mathbb{Z}$) *cube displacements.*

Each cube displacement can be translated in such a way—in fact, generally, in many ways—that both of its endpoints reside in the cube.

Definition A.7. Fix a cube displacement $\mathbf{y}$. We call $\mathscr{X}_0(\mathbf{y}) := \{\mathbf{v}_0 \in \{0,1\}^n \mid \mathbf{v}_0 + \mathbf{y} \in \{0,1\}^n\}$ and $\mathscr{X}_1(\mathbf{y}) := \{\mathbf{v}_1 \in \{0,1\}^n \mid \mathbf{v}_1 - \mathbf{y} \in \{0,1\}^n\}$ (where all addition and subtraction takes place in $\mathbb{Z}^n$) the *originating* and *terminating* sets, respectively, of $\mathbf{y}$.

Lemma A.8. *Fix a cube displacement $\mathbf{y}$, and write d for the number of components of $\mathbf{y}$ which are zero. The originating and terminating sets $\mathscr{X}_0(\mathbf{y})$ and $\mathscr{X}_1(\mathbf{y})$ are proper subcubes of $\{0,1\}^n$, each of dimension d.*

Proof. We write $\mathbf{y} = (y_0, \ldots, y_{n-1})$ and $\{c_0, \ldots, c_{n-d-1}\} \subset \{0, \ldots, n-1\}$ for the indices at which $\mathbf{y}$ is nonzero. It is clear that $\mathscr{X}_0(\mathbf{y}) = \left\{(x_0, \ldots, x_{n-1}) \in \{0,1\}^n \,\middle|\, x_{c_0} = \frac{1-y_{c_0}}{2}, \ldots, x_{c_{n-d-1}} = \frac{1-y_{c_{n-d-1}}}{2}\right\}$ and $\mathscr{X}_1(\mathbf{y}) = \left\{(x_0, \ldots, x_{n-1}) \in \{0,1\}^n \,\middle|\, x_{c_0} = \frac{1+y_{c_0}}{2}, \ldots, x_{c_{n-d-1}} = \frac{1+y_{c_{n-d-1}}}{2}\right\}$. $\square$

We refer to Cohen, Honkala, Litsyn, and Lobstein [CHLL97, §2] for background on codes. We assume the usual notion of *Hamming distance* $d(\mathbf{x}_i, \mathbf{x}_j)$ for elements $\mathbf{x}_i$ and $\mathbf{x}_j$ of $\{0,1\}^n$, as well as that of the *distance* of a code $C \subset \{0,1\}^n$, defined as the $\min_{\mathbf{x}_i \neq \mathbf{x}_j} d(\mathbf{x}_i, \mathbf{x}_j)$ (where $\mathbf{x}_i$ and $\mathbf{x}_j$ are in C). We record the following result of Tietäväinen; actually, we give a slight variant stated in Krasikov and Litsyn [KL97, (3)].

Theorem A.9 (Tietäväinen [Tie80, Thm. 2])**.** *Fix an integer sequence $j = j(n)$ such that $j \in o(n^{1/3})$. Then for each large enough n, any code $C \subset \{0,1\}^n$ of distance at least $d = \frac{n-j}{2}$ satisfies $|C| < \frac{n \cdot \ln j}{2} \cdot (1 + o(1))$.*

We refer to [CLRS09, §B.4] for preliminaries on graphs. We write $c(G)$ for the number of connected components of an undirected graph $G = (V, E)$.

Lemma A.10. *If $G = (V, E)$ is such that each vertex $\mathbf{v}$ of V has degree bounded by d, then $c(G) \leq |V| - \frac{|E|}{d}$.*

Proof. We first argue that it suffices to assume that G is connected. Indeed, assuming that the claim were true for connected graphs $G = (V, E)$—that is, that $|V| - \frac{|E|}{d} \geq 1$ for each such graph—we could partition both V *and* E along G's components, and apply the lemma component-wise. The degree bound d would clearly continue to hold component-wise. Adding up the resulting inequalities would yield the lemma.

We thus assume that G is connected. If G is edgeless—that is, a single vertex—then the desired conclusion trivially holds. We thus assume that G has at least one edge. We may freely replace d by the maximal degree *actually* attained in G; indeed, doing so can only decrease d, and so strengthen the conclusion being proven. It follows in particular that $|E| \geq d$. By the handshaking lemma (see e.g. [CLRS09, Ex. B.4-1]), we also have $\sum_{\mathbf{v} \in V} \deg(v) = 2 \cdot |E|$, so that $d \cdot |V| \geq 2 \cdot |E|$, by hypothesis on G's degree. We thus finally have that $|V| \geq \frac{2 \cdot |E|}{d}$, so that $|V| - \frac{|E|}{d} \geq \frac{2 \cdot |E|}{d} - \frac{|E|}{d} = \frac{|E|}{d} \geq 1$. This completes the proof. □

We would like to thank the Mathematics StackExchange user Hagen von Eitzen for the above argument.

Definition A.11. Fix a set $C = \{\mathbf{x}_0, \ldots, \mathbf{x}_{k-1}\} \subset \{0,1\}^n$. The *displacement graph* $G = (V, E)$ associated with C has vertex set $V = \{0,1\}^n$ and an undirected edge between nodes $\mathbf{v}_0$ and $\mathbf{v}_1$ if and only if $\mathbf{v}_1 - \mathbf{v}_0 = \mathbf{x}_j - \mathbf{x}_i$ for some pair of distinct elements $\mathbf{x}_i$ and $\mathbf{x}_j$ of C.

Lemma A.12. *Fix a set $C \subset \{0,1\}^n$ with displacement graph $G = (V, E)$. Given any particular pair of elements $\mathbf{x}_i$ and $\mathbf{x}_j$ of C, the number of connected components $c(G) \leq 2^n - 2^{n - d(\mathbf{x}_i, \mathbf{x}_j)}$.*

Proof. The cube displacement $\mathbf{y} := \mathbf{x}_j - \mathbf{x}_i$ has exactly $n - d(\mathbf{x}_i, \mathbf{x}_j)$ 0-valued components; by Lemma A.8, the originating and terminating sets $\mathscr{X}_0(\mathbf{y})$ and $\mathscr{X}_1(\mathbf{y})$ have dimension $n - d(\mathbf{x}_i, \mathbf{x}_j)$. The reachability relation on G thus identifies *at least* $2^{n - d(\mathbf{x}_i, \mathbf{x}_j)}$ disjoint pairs of vertices; in fact, the sets $\mathscr{X}_0(\mathbf{y})$ and $\mathscr{X}_1(\mathbf{y})$ are identified under G in one-to-one manner. This identification alone reduces the cardinality $c(G)$ by $2^{n - d(\mathbf{x}_i, \mathbf{x}_j)}$. □

Lemma A.13. *Fix a k-dimensional linear subspace $K \subset \mathbb{F}_p^n$, generated by cube elements $\{\mathbf{x}_0, \ldots, \mathbf{x}_{k-1}\}$, say. Write $G = (V, E)$ for K's displacement graph. Then if $A : \mathbb{F}_p^n \to \mathbb{F}_p^{n-k}$ annihilates exactly K, then*

$$|A(\{0,1\}^n)| \leq c(G).$$

Proof. The idea is to show that the reachability relation in G is an (in general, strict) refinement of the quotient-by-K relation. Indeed, any reachable elements $\mathbf{v}_0$ and $\mathbf{v}_1$ in $\{0,1\}^n$ necessarily differ by a combination (with $\{-1, 1\}$-coefficients) of displacements $\mathbf{x}_j - \mathbf{x}_j$ in K; this sum of course itself resides in K. □

We now give the main technical result of this subsection. This theorem bounds from above the number of connected components of displacement graphs G on *linearly independent* sets $C \subset \{0,1\}^n$.

Theorem A.14. *Fix a constant $c \in \mathbb{N}$. If n is large enough (depending only on c) then for each odd prime p, dimension $n - c \leq k < n$, and linear subspace $K \subset \mathbb{F}_p^n$ generated over $\mathbb{F}_p$ by cube elements $\{\mathbf{x}_0, \ldots, \mathbf{x}_{k-1}\}$,*

$$|A(\{0,1\}^n)| \leq 2^n - \frac{n}{\log \log n} \cdot \sqrt{2^n},$$

where $A : \mathbb{F}_p^n \to \mathbb{F}_p^{n-k}$ is any linear map annihilating exactly K.

Proof. We write $G = (V, E)$ for the displacement graph on the set $K \cap \{0,1\}^n = \{\mathbf{x}_0, \ldots, \mathbf{x}_{k-1}\}$. We begin with the following sub-claim, which bounds from below the number of edges $|E|$ in G.

Lemma A.15. *For each $\varepsilon > 0$ and large enough n, the graph $G = (V, E)$ has at least $\frac{7}{10} \cdot \frac{n^2}{2} \cdot 2^{n/2}$ edges.*

Proof. Each pair of elements $\mathbf{x}_i$ and $\mathbf{x}_j$ of C for which $i < j$ contributes $2^{n-d(\mathbf{x}_i,\mathbf{x}_j)}$ edges to E; indeed, the edges induced by $\mathbf{x}_i$ and $\mathbf{x}_j$ identify pairs of elements in originating and terminating sets $\mathscr{X}_0(\mathbf{x}_j - \mathbf{x}_i)$ and $\mathscr{X}_1(\mathbf{x}_j - \mathbf{x}_i)$ (see also Lemma A.8). Moreover, the $\binom{k}{2}$ resulting such edge sets are disjoint; this is exactly because the differences $\mathbf{x}_j - \mathbf{x}_i$ (for $i < j$) are pairwise distinct (we use the linear independence here).

We now construct from each element $\mathbf{x}_i = (x_{i,0}, \ldots, x_{i,n-1})$ of C the $\{-1,1\}$-vector $\mathbf{x}'_i := ((-1)^{x_{i,0}}, \ldots, (-1)^{x_{i,n-1}}) \in \mathbb{Z}^n$. It is clear that for any two elements $\mathbf{x}_i$ and $\mathbf{x}_j$ of C, $\langle \mathbf{x}'_i, \mathbf{x}'_j \rangle = n - 2\cdot d(\mathbf{x}_i, \mathbf{x}_j)$. Interpreting all quantities over $\mathbb{Z}^n$, we have the inequality:

$$0 \leq \left| \sum_{i=0}^{k-1} \mathbf{x}'_i \right|^2 = \left\langle \sum_{i=0}^{k-1} \mathbf{x}'_i, \sum_{i=0}^{k-1} \mathbf{x}'_i \right\rangle = \sum_{i=0}^{k-1} |\mathbf{x}'_i|^2 + \sum_{i<j} \langle \mathbf{x}'_i, \mathbf{x}'_j \rangle \leq k \cdot n + 2 \cdot \sum_{i<j} (n - 2 \cdot d(\mathbf{x}_i, \mathbf{x}_j)),$$

so that $\sum_{i<j} d(\mathbf{x}_i, \mathbf{x}_j) \leq \frac{1}{4} \cdot (k \cdot n + k \cdot (k-1) \cdot n) = \frac{k^2 \cdot n}{4}$.

Putting these facts together, we see that the number of edges:

$$|E| = \sum_{i<j} 2^{n-d(\mathbf{x}_i,\mathbf{x}_j)} \geq \binom{k}{2} \cdot 2^{n - \frac{\sum_{i<j} d(\mathbf{x}_i,\mathbf{x}_j)}{\binom{k}{2}}} \geq \binom{k}{2} \cdot 2^{n - \frac{k^2 \cdot n/4}{\binom{k}{2}}} = \binom{k}{2} \cdot 2^{\frac{k-2}{k-1} \cdot \frac{n}{2}},$$

where the first inequality is a direct consequence of (the simple form of) Jensen's inequality $\varphi\left(\frac{\sum_i x_i}{n}\right) \leq \frac{\sum_i \varphi(x_i)}{n}$ for any convex function φ (we take here $\varphi : t \mapsto 2^{n-t}$) and the second inequality follows from the above discussion.

Because $k \geq n - c$, for any $\varepsilon > 0$ and large enough n, it holds that

$$\binom{k}{2} \cdot 2^{\frac{k-2}{k-1} \cdot \frac{n}{2}} \geq (1-\varepsilon) \cdot \frac{n^2}{2} \cdot 2^{n/2 - \frac{1}{2} - \varepsilon} = \frac{(1-\varepsilon)}{2^{1/2+\varepsilon}} \cdot \frac{n^2}{2} \cdot 2^{n/2}.$$

Because $\frac{1}{\sqrt{2}} > \frac{7}{10}$, it follows that $\frac{(1-\varepsilon)}{2^{1/2+\varepsilon}} \geq \frac{7}{10}$ for small enough ε. $\square$

The next lemma bounds from above the degree of any particular node $\mathbf{x} \in V$:

Lemma A.16. *For large enough n, we may assume that each node of V has degree at most $\frac{7}{10} \cdot \frac{n}{2} \cdot \log\log n$.*

Proof. Applying Lemma 2.1, we assume without loss of generality that $\mathbf{v}$ is the origin. Each edge incident on $\mathbf{v}$ corresponds to a cube displacement $\mathbf{y} = \mathbf{x}_j - \mathbf{x}_i$ for which $\mathbf{v} \in \mathscr{X}_0(\mathbf{y})$ (where $\mathbf{x}_i$ and $\mathbf{x}_j$ are unequal elements of $K \cap \{0,1\}^n$). Any such $\mathbf{y}$ satisfies $\mathbf{v} + \mathbf{y} = \mathbf{y} \in \{0,1\}^n$, so that $\mathbf{y}$ (under our assumption $\mathbf{v}$) is a $\{0,1\}$-*vector* (as opposed to a $\{-1,0,1\}$-vector).

If any pair of displacements $\mathbf{y}_0$ and $\mathbf{y}_1$ originating from $\mathbf{v}$ satisfied $d(\mathbf{y}_0, \mathbf{y}_1) < \frac{n}{2} - \log n$, then the difference $\mathbf{y}_1 - \mathbf{y}_0$ would *itself* be a cube displacement, with $n - d(\mathbf{y}_0, \mathbf{y}_1) > \frac{n}{2} + \log n$ 0-valued components. Lemma A.12 would immediately then assert that $c(G) < 2^n - 2^{\frac{n}{2} + \log n} = 2^n - n \cdot \sqrt{2^n}$, which would imply the statement of the theorem by Lemma A.13.

We thus assume freely that each pair of displacements $\mathbf{y}_0$ and $\mathbf{y}_1$ originating from $\mathbf{v}$ satisfies $d(\mathbf{y}_0, \mathbf{y}_1) \geq \frac{n}{2} - \log n$. By definition, the set of displacements $\mathbf{y}$ originating from $\mathbf{v}$ thus gives a code $C \subset \{0,1\}^n$ of minimum distance at least $\frac{n}{2} - \log n$. Applying Theorem A.9 with $j(n) := 2 \cdot \log n$, we see that C satisfies:

$$\begin{aligned}
|C| &\leq \frac{n \cdot \ln(2 \cdot \log n)}{2} \cdot (1 + o(1)) \\
&= \ln 2 \cdot \frac{n}{2} \cdot (1 + \log\log n) \cdot (1 + o(1)) \\
&= \ln 2 \cdot \frac{n}{2} \cdot \log\log n + \left(o(1) \cdot \ln 2 \cdot \frac{n}{2} \cdot \log\log n + \ln 2 \cdot \frac{n}{2} \cdot (1 + o(1)) \right).
\end{aligned}$$

Because $\ln 2 < \frac{7}{10}$, we see that $|C| \leq \frac{7}{10} \cdot \frac{n}{2} \cdot \log\log n$ (for large n). This implies the lemma. $\square$

Lemmas A.15 and A.16—together with Lemma A.10—finally give:

$$c(G) \leq 2^n - \frac{\frac{7}{10} \cdot \frac{n^2}{2} \cdot 2^{n/2}}{\frac{7}{10} \cdot \frac{n}{2} \cdot \log\log n} = 2^n - \frac{n}{\log\log n} \cdot 2^{n/2}.$$

The proof of the theorem follows from Lemma A.13. $\square$

We now have the following analogue of Corollary A.4:

Corollary A.17. *For each large enough n, that a prime p satisfies $p \geq 2^n - \frac{n}{\log\log n} \cdot \sqrt{2^n}$ implies that any $n-2$-dimensional affine flat $K \subset \mathbb{F}_p^n$ admits an affine hyperplane $H \subset \mathbb{F}_p^n$ for which $K \cap \{0,1\}^n = H \cap \{0,1\}^n$.*

Proof. We fix a quotient map $A : \mathbb{F}_p^n \to \mathbb{F}_p^2$ of K. By Theorem A.14 (with $c = 2$), for large enough n the image $A(\{0,1\}^n) \subset \mathbb{F}_p^2$ has cardinality at most $2^n - \frac{n}{\log\log n} \cdot \sqrt{2^n}$. The image space $\mathbb{F}_p^2$ contains $p+1$ lines through the origin; by hypothesis on p, this number exceeds the cardinality of $A(\{0,1\}^n - K)$, and at least one of these lines must "miss" the image. Annihilating this line yields a composition $H : \mathbb{F}_p^n \xrightarrow{A} \mathbb{F}_p^2 \xrightarrow{A_1} \mathbb{F}_p$, which defines the desired hyperplane. □

We discuss now whether an analogue of Corollary A.5 can be achieved in this setting; that is, whether the existence of a prime p which satisfies the hypothesis of Corollary A.17 *and* satisfies $p < 2^n$ can be guaranteed. Despite significant empirical evidence, even the existence of primes in intervals of the form $(x - (1+\varepsilon) \cdot \log x \cdot \sqrt{x}, x)$ has not been unconditionally established; indeed, their existence is implied by the Riemann hypothesis, as demonstrated by recent work of Dudek [Dud14, Thm. 1.3].

Thus Corollary A.17 *alone* fails to guarantee the existence of primes $p < 2^n$ which admit extensions in the sense of Question A.1, even conditioned on the Riemann hypothesis. Nonetheless, the existence of such primes would follow from certain stronger conjectures, like Montgomery's pair correlation conjecture; we refer to Heath-Brown and Goldston [HBG84]. We thus record the following analogue of Corollary A.5:

Corollary A.18. *Assume Montgomery's pair correlation conjecture. Then for each large enough n, there exists a prime $p < 2^n$ such that every $n-2$-dimensional affine flat $K \subset \mathbb{F}_p^n$ admits a hyperplane extension $H \subset \mathbb{F}_p^n$ such that $K \cap \{0,1\}^n = H \cap \{0,1\}^n$.*

Proof. Assuming the pair correlation conjecture, [HBG84] implies that primes eventually exist in the intervals $(2^n - \sqrt{n \cdot 2^n}, 2^n]$. Because $\sqrt{n} \cdot 2^{n/2}$ is (eventually) smaller than $\frac{n}{\log\log n} \cdot 2^{n/2}$, this in turn guarantees the existence of primes $p < 2^n$ which also satisfy the hypothesis of Corollary A.17. □