Randomizing Polynomials: A New Representation with Applications to Round-Efficient Secure Computation

(Extended Abstract)

Yuval Ishai *   Eyal Kushilevitz †

Abstract

Motivated by questions about secure multi-party computation*, we introduce and study a new natural representation of functions by polynomials, which we term* randomizing polynomials. *"Standard" low-degree polynomials over a finite field are easy to compute with a small number of communication rounds in virtually any setting for secure computation. However, most Boolean functions cannot be evaluated by a polynomial whose degree is smaller than their input size. We get around this barrier by relaxing the requirement of evaluating f into a weaker requirement of* randomizing f*: mapping the inputs of f along with independent random inputs into a vector of outputs, whose distribution depends only on the value of f. We show that degree-3 polynomials are sufficient to randomize any function f, relating the efficiency of such a randomization to the* branching program *size of f. On the other hand, by characterizing the exact class of Boolean functions which can be randomized by degree-2 polynomials, we show that 3 is the minimal randomization degree of most functions.*

As an application, randomizing polynomials provide a powerful, general, and conceptually simple tool for the design of round-efficient secure protocols. Specifically, the secure evaluation of any function can be reduced to a secure evaluation of degree-3 polynomials. One corollary of this reduction is that two (respectively, three) communication rounds are sufficient for k parties to compute any Boolean function f of their inputs, with perfect information-theoretic $\lfloor\frac{k-1}{3}\rfloor$-privacy (resp., $\lfloor\frac{k-1}{2}\rfloor$-privacy), and communication complexity which is at most quadratic in the branching program size of f (with a small probability of one-sided error).

1. Introduction

A multivariate polynomial $p(x_1,\ldots,x_n)$ over a field $\mathcal{K}$ is said to represent a Boolean function $f : \{0,1\}^n \rightarrow \{0,1\}$ if $p(x) = f(x)$ for any $x \in \{0,1\}^n$. Representation of Boolean functions by polynomials has proved to be an extremely useful tool in complexity theory. In the cryptographic context of secure multi-party computation, low-degree polynomials serve as the most natural candidates for functions admitting round-efficient protocols.[1] Unfortunately, while all Boolean functions on n bits can be represented by a degree-n polynomial, most cannot be represented by a polynomial of a lower degree. This is the case even for some very simple functions, such as the logical AND and logical OR of n input bits. For this reason, constant-degree polynomials are typically viewed as a very weak computation model.

We introduce a more general notion of *randomizing* polynomials. This new notion syntactically differs from the standard polynomial representation defined above in two aspects: (1) to represent a function $f(x_1,\ldots,x_n)$ we allow the polynomials to depend not only on the input variables $x_1,\ldots,x_n$ but also on some *random inputs* $r_1,\ldots,r_m$; (2) the representation consists of an s-tuple of polynomials $p = (p_1,\ldots,p_s)$ instead of a single polynomial. On input $x = (x_1,\ldots,x_n)$, independent random choices of the m random inputs $r_1,\ldots,r_m$ naturally induce an *output distribution* on s-tuples of field elements. To represent f, we require p to *randomize* every input instance x into an output distribution which is uniquely associated with the output $f(x)$. That is, for any output value b there should exist a distinct distribution D_b, such that D_b is the output distribution on any input x with $f(x) = b$. Moreover, we require that the distributions D_b associated with different output values of f be statistically far from each other. Note that for the special case where $s = 1$ and $m = 0$, one gets the standard polynomial representation.

We obtain the following results:

- Degree-3 randomizing polynomials are sufficient to represent any function f over any finite field; the number of outputs, s, and the number of random inputs, m, are at most quadratic in the branching program size of f (see Section 2.3 for definitions of the branching program model).

*DIMACS and AT&T Labs Research. E-mail: yuval@dimacs.rutgers.edu.

†Department of Computer Science, Technion. E-mail: eyalk@cs.technion.ac.il . Work done while at IBM T.J. Watson Research Center.

[1] The reason being the simplicity of implementing the addition operation in a non-interactive manner in protocols based on linear secret-sharing.

- Degree-2 randomizing polynomials cannot represent any Boolean function, except those defined by systems of linear equations and those which can be represented by standard degree-2 polynomials.

We demonstrate the usefulness of randomizing polynomials by applying them in the context of secure multi-party computation (see Section 2.2 for problem description and definitions). The main conceptual contribution of our results can be concisely stated as follows:

> A secure evaluation of an arbitrary function f may be reduced to a secure evaluation of degree-3 polynomials.

Moreover, if f admits an *efficient* representation by randomizing polynomials, as is the case for all functions in NL, then the above reduction becomes efficient, and consequently *any* reasonable notion of security, computational or information-theoretic, can be substituted for the term "secure" above. Otherwise, the applicability of this reduction is restricted to the information-theoretic setting.[2] A secure evaluation of the degree-3 polynomials obtained via our reduction can be carried out in a round-efficient manner by using virtually any general protocol for secure computation (e.g., [9, 12, 21, 16, 27, 15]).

The simplest illustration of the type of bounds one may obtain using our approach is for the setting of *information-theoretic* security against a *passive* adversary. Let $f(y_1, \ldots, y_k)$ be a Boolean function, where each y_j is a bit string. A k-party distributed protocol for evaluating f is said to be (perfectly) t-*private* if any collusion of at most t parties cannot learn any information from their joint view of the communication, other than what follows from their inputs and the value of f. For this setting, we obtain:

> Two (respectively, three) communication rounds are sufficient to evaluate any k-argument function f with perfect information-theoretic $\lfloor \frac{k-1}{3} \rfloor$-privacy (resp., $\lfloor \frac{k-1}{2} \rfloor$-privacy), probabilistic correctness, and communication complexity which is at most quadratic in the branching program size of f and the number of parties k.

Alternatively, one can obtain protocols with zero error probability, perfect privacy, and *expected* $2 + \epsilon$ (resp., $3 + \epsilon$) rounds, for an arbitrarily small ϵ. This improves both the round complexity and the communication complexity of previous constant-round protocols (see Section 1.1 for an account on related work).

We would like to stress that apart from the above concrete results, a major contribution of this work is in providing a very natural *paradigm* for round-efficient secure computation, which both simplifies and improves over previous approaches. Moreover, as there are yet no known lower bounds on the round-complexity of general protocols for secure computation, our ability to obtain non-trivial lower bounds on the power of randomizing polynomials may be viewed as a promising progress.

[2] Even for functions f for which our constructions are inefficient, they still improve over previous solutions for the information-theoretic setting.

1.1. Related Work

Different notions of representation of Boolean functions by polynomials (and, in particular, by low-degree polynomials) have been applied in numerous complexity theoretic contexts. For instance, polynomial representations have been applied in the areas of circuit lower-bounds, interactive proofs and PCP characterizations of NP, average-case to worst-case reductions for hard problems, and program checking (see [19] for a survey of some related work). The reader is referred to [28, 31, 24, 32, 3, 33, 26, 7] for results on polynomial representations and their applications in complexity theory. Our notion of randomizing polynomials may be viewed as another natural form of polynomial representation, with a different flavor of applications.

The *round complexity* of interactive protocols is one of their most important complexity measures. Consequently, a substantial amount of research effort has been invested into characterizing the exact round complexity of various tasks (such as zero-knowledge proofs, Byzantine Agreement, and others). In particular, the problem of obtaining constant-round secure computation protocols has attracted a considerable amount of attention [2, 6, 5, 17, 22, 29, 4, 10, 14]. In the following we briefly survey some relevant works, starting with the information-theoretic setting.

The problem was first explicitly considered by Bar-Ilan and Beaver [2]. They constructed an efficient, (expected) constant-round, information-theoretically secure protocol with an optimal security threshold for every function in NC^1.[3] Feige, Kilian, and Naor [17], followed by Ishai and Kushilevitz [22], study a specialized non-interactive model for secure computation. Their results can be used to extend the class of functions for which efficient constant-round protocols are known from NC^1 to NL and similar log-space classes. A slight further extension of this class, as well as communication-efficient constant-round protocols for natural linear algebra problems, are given by Cramer and Damgård [14].

A key idea in the constructions of [2, 17], originally due to Kilian [23], is a reduction of the evaluation of f to an evaluation of an iterated product in a finite group. The technique which was used for a constant-round secure evaluation of this product, in turn, required a distributed generation of secret group elements coupled with their inverses. This was realized in [2] by using an interactive inversion sub-protocol, whose output was subsequently used by the main protocol (incurring additional rounds of interaction). In the current work we utilize a different, inversion-free, randomization approach, extending a technique from [22]. An alternative inversion-free randomization technique (though generally less efficient than that of [22]) was recently suggested by Beaver [4]. While protocols from [22, 4] still require a distributed generation of secret *invertible* matrices (realized

[3] Although the protocols in [2] are running in *expected* constant-rounds, they can be transformed into constant-round protocols with the penalty of introducing a small probability of error.

by a sub-protocol from [2]) to guarantee their perfect privacy and correctness, this step is avoided by our protocols, even in their zero-error variants.[4]

In the *computational* model of secure computation, constant-round protocols were given for every polynomial-time computable function by Beaver, Micali, and Rogaway [6], generalizing a 2-party "garbled circuit" construction due to Yao [36]. Since our constructions do not efficiently extend to all polynomial-time computable functions, they are generally less appealing in the computational setting. Nevertheless, our information-theoretic approach may be useful in the computational setting as well, even in the 2-party case.[5] One advantage of information-theoretic randomization techniques over Yao's garbled circuit technique is that they better scale to arithmetic computation over large moduli. An additional application is to the 2-party 2-message setting of Sandler, Young, and Yung [29]; our techniques can be used to obtain protocols which are perfectly private for the party sending the *second* message (improving the efficiency of constructions from [29, 4]). This is contrasted with the solutions of Cachin et al. [10], which are perfectly private for the party sending the first message.

Organization: Section 2 includes definitions of *randomizing polynomials*, *secure computation*, and *branching programs*. Section 3 contains constructions of randomizing polynomials, and Section 4 discusses the applications of these constructions to the design of round-efficient secure computation protocols. In Section 5 we characterize the power of degree-2 randomizing polynomials, and in particular show that they cannot represent all functions. Finally, Section 6 includes some concluding remarks, and the appendix contains proofs deferred from the main body of text.

2. Preliminaries

Throughout the following we let $\mathcal{K}$ denote a finite field. The underlying field $\mathcal{K}$ will often be omitted, under the understanding that it may be taken to be *any* finite field. By $\mathcal{K}^{n,\ell}$ we denote the set of all matrices over $\mathcal{K}$ with n rows and ℓ columns. We use capital letters to denote random variables, probability distributions, or matrices. Vectors are denoted by small letters, and are regarded by default as column vectors; x^t denotes the row-vector form of x. We write $Y_0 \equiv Y_1$ to indicate that the probability distributions or random variables Y_0 and Y_1 are identically distributed.

Given discrete probability distributions Y_0, Y_1, the statistical distance between Y_0 and Y_1 is defined as $\mathrm{SD}(Y_0, Y_1) = \frac{1}{2}\sum_y |\Pr[Y_0 = y] - \Pr[Y_1 = y]|$. Equivalently, the statistical distance between Y_0 and Y_1 may be defined as the maximum, over all functions A, of the *distinguishing advantage* $|\Pr[A(Y_0) = 1] - \Pr[A(Y_1) = 1]|$.

2.1. Randomizing Polynomials

Let $p = (p_1, \ldots, p_s)$ be a vector of polynomials in the $n + m$ variables $x = (x_1, \ldots, x_n)$ and $r = (r_1, \ldots, r_m)$ over $\mathcal{K}$. That is, for any $x \in \mathcal{K}^n$ and $r \in \mathcal{K}^m$, $p(x, r) = (p_1(x, r), \ldots, p_s(x, r))$ outputs an s-tuple $y \in \mathcal{K}^s$. The variables $x_1, \ldots, x_n$ will be referred to as the *inputs* of p and $r_1, \ldots, r_m$ as its *random inputs*. For any $x \in \mathcal{K}^n$, let $P(x)$ denote the probability distribution of the output $y = p(x, r)$, induced by a uniform choice of $r \in \mathcal{K}^m$.

Definition 2.1 *A polynomial vector $p(x, r)$ is said to* randomize *a Boolean function $f : \{0, 1\}^n \to \{0, 1\}$ if there exist probability distributions D_0, D_1 such that:*

1. *The output distribution depends only on $f(x)$; that is, for any input $x \in \{0, 1\}^n$, $P(x) \equiv D_{f(x)}$.*
2. *The distributions D_0 and D_1 are statistically far; specifically, we require* $\mathrm{SD}(D_0, D_1) \geq 1/2$.

Motivated by the application to secure computation, the first requirement will sometimes be referred to as the *privacy* property and the second as the *correctness* property.

An important parameter for randomizing polynomials is their *degree*. For the purpose of this definition, we view all polynomials in their canonical form as sums of monomials. The degree of each monomial is the sum of the degrees of all variables (both x_j's and r_j's); the degree of a polynomial is the degree of its maximal-degree monomial, and the degree of a polynomial vector p is the degree of its maximal-degree entry p_i. The term "degree-d polynomial" should be read as "a polynomial of degree *at most* d".

While the main parameter of interest in this work is the degree of the polynomial vector p, we will also be interested in its *output complexity* s and its *randomness complexity* m. The *complexity* of p is the sum of its output complexity and its randomness complexity.

Note that the above definitions are purely information-theoretic; they do not require the existence of an *efficient* distinguisher between the output distributions D_0 and D_1, or even that the randomizing polynomials be efficiently computable. However, all of the constructions in this work are computationally efficient in their description size. In particular, the output distributions of the randomizing polynomials we construct will always be efficiently distinguishable from each other.

The constant $1/2$ in Definition 2.1 is arbitrary; the statistical distance between the output distributions D_0 and D_1 can be amplified to $1-\epsilon$ by concatenating $O(\log 1/\epsilon)$ copies of p, each with a disjoint set of random inputs. While such an amplification moderately increases the complexity of p, it keeps its degree unchanged. We refer to randomizing polynomials with statistical distance of $1 - \epsilon$ as being *ϵ-correct*.

[4] An obvious way to avoid generating invertible matrices is to replace them by uniformly random matrices over a very large field. However, protocols based on this approach may lose their perfect privacy and correctness, and will lack the efficiency advantages of small-field arithmetic.

[5] In the two-party case, multivariate polynomials may be privately evaluated using a homomorphic encryption scheme.

Furthermore, we say that p is ϵ-correct with a *one-sided error* if there exists a distinguisher A such that $\Pr[A(D_0) = 1] = 0$ and $\Pr[A(D_1) = 1] \geq 1 - \epsilon$. Note that if a Boolean function f is randomized by an ϵ-correct $p(x, r)$ (with one-sided error), then $f(x)$ can be reconstructed from $P(x)$ with (one-sided) error probability of at most ϵ.

Remark 2.2 Both Definition 2.1 and the results of the following sections can be generalized to non-Boolean functions. A naive way for randomizing a non-Boolean function is by concatenating randomizations of the Boolean functions defined by its individual output bits. For simplicity, we focus in this work on the Boolean case.

Example 2.3 *Consider the parity function $\oplus(x_1, \ldots, x_n)$. It can be evaluated, over $\mathcal{K} = \mathrm{GF}(2)$, by the polynomial $p(x_1, \ldots, x_n) = x_1 + x_2 + \ldots + x_n$. Hence the parity function trivially admits a degree-1 randomization over $\mathrm{GF}(2)$ with output complexity 1 and randomness complexity 0.*

Example 2.4 *Consider the OR function $\vee(x_1, \ldots, x_n)$. This function can be randomized by the following degree-2 polynomial vector:*

$$p(x_1, \ldots, x_n, r_1, \ldots, r_n) \stackrel{\text{def}}{=} \sum_{i=1}^{n} x_i r_i.$$

Here the output complexity is again 1, but the randomness complexity is n. Note that if $\vee(x_1, \ldots, x_n) = 0$ then $p(x, r)$ is constantly 0 while if $\vee(x_1, \ldots, x_n) = 1$ then $p(x, r)$ is uniformly distributed over $\mathcal{K}$. This gives D_0 and D_1 which satisfy $\mathrm{SD}(D_0, D_1) = 1 - 1/|\mathcal{K}| \geq 1/2$. In the following we use vector notation, under which the above may be written as $p(x, r) = r^t x$. Note, again, that using ℓn random inputs and ℓ outputs, the statistical distance may be amplified to $1 - 2^{-\ell}$. This is realized by the degree-2 mapping $p(x, R) = Rx$, where the random inputs are viewed as an $\ell \times n$ matrix R.

2.2. Secure Multi-Party Computation

Let $\mathcal{P}_1, \ldots, \mathcal{P}_k$ be k parties, where each party $\mathcal{P}_j$ holds an input y_j. In this paper, each y_j will either be a bit string or a vector over the field $\mathcal{K}$. We will use y to denote the k-tuple of inputs, and $x = (x_1, \ldots, x_n)$ to denote the concatenation of the k inputs, viewed as a vector from $\mathcal{K}^n$. The parties wish to compute some function f on their inputs. For simplicity, we will only consider here the case of *deterministic*, *single-output* functions.

To the end of computing f, the parties engage in a randomized distributed protocol. Initially, each party $\mathcal{P}_j$ holds its input y_j and an independent random input. A *protocol* $\mathcal{F}$ proceeds by synchronous rounds, where at each round each party may send messages to all other parties depending on its input, its random input, and the messages it received in previous rounds. At the end of each round, each party $\mathcal{P}_j$ may decide to terminate and output some value z_j. The execution terminates when all parties terminate. The *round complexity* of $\mathcal{F}$ is defined as the maximal number of rounds before termination, taken over all possible executions of $\mathcal{F}$.

There are two major settings for secure computation: (a) The secure channels model [9, 12, 27, 16], where each pair of parties may secretly communicate with each other. In this model security is usually required to be *information theoretic*. (b) The computational model [35, 21], where no secure channels are assumed but instead parties are restricted to probabilistic polynomial-time computations; as a result intractability assumptions can be made and the notion of security becomes *computational*. Intuitively, a protocol is said to be *t-secure*, in any of the settings, if it is resilient to an adversary which is allowed to corrupt at most t parties. This is formalized by requiring that whatever an adversary can achieve (and learn) in the "real-life" execution of the protocol, it could have also achieved in an ideal model, where a trusted party is being used to perform the computation. The different notions of security differ in the amount of "corruption power" the adversary may have. An *active* adversary is allowed to maliciously alter the behavior of the parties it corrupts, whereas a passive adversary only learns their view of the protocol.

In the following we define the simplest variant of *information-theoretic* security against a *passive* adversary. Security in this setting will be referred to as *privacy*. We say that $\mathcal{F}$ is an *ϵ-correct*, (perfectly) *t-private* protocol for computing f, if it satisfies the following two properties: (1) ϵ-correctness – for every input y, the probability (over the random inputs) that some party fails to output $f(x)$ is bounded by ϵ; in the current work we mostly consider this probabilistic notion of correctness (where ϵ may be taken to be exponentially small), rather than the perfect one. (2) t-privacy – a collusion of at most t parties cannot learn any information about the inputs of the remaining parties, other than what follows from their inputs and the value of f. This can be formalized by requiring the existence of a probabilistic *simulator* algorithm S, satisfying the following condition. For any input y and collusion B of at most t parties, the output generated by S on input $(B, y_B, f(y))$, where y_B denotes y restricted to its B-entries, is distributed identically to the joint view of parties from B in the execution of $\mathcal{F}$ on input y (including the inputs, random inputs, and communication). Finally, if f is a Boolean function, we say that $\mathcal{F}$ is ϵ-correct with a *one-sided error* if $\mathcal{F}$ is ϵ-correct and it never errs on input y such that $f(y) = 0$.

The exact definition of security in more general or more adversarial settings is a delicate issue; for a more complete treatment the reader is referred to, e.g., [11, 20, 25]. We emphasize though that the issues addressed in this paper are quite insensitive to the exact notion of security. Our results may be viewed as providing an *information-theoretic reduction* from the problem of securely computing a function f to the problem of securely computing a vector of low-degree polynomials.

2.3. Branching Programs

Our constructions use the following notion of *modular branching programs*, which generalizes the standard notions of deterministic and non-deterministic branching programs.

Definition 2.5 *A (non-deterministic)* mod-p branching program *is defined by a quadruple BP* $= (G, \phi, s, t)$*, where* $G = (V, E)$ *is a directed acyclic graph,* ϕ *is a labeling function assigning each edge a negative literal* x_i^0*, a positive literal* x_i^1 *or the constant 1, and* s, t *are two special vertices. Each input assignment* $x = (x_1, \ldots, x_n)$ *naturally induces an unlabeled subgraph* G_x*, whose edges include every* $e \in E$ *such that* $\phi(e)$ *is satisfied by* x*. The* function computed by *BP is the Boolean function* f *satisfying* $f(x) = 1$ *iff the number of "accepting"* $s - t$ *paths in* G_x *is nonzero modulo-*p*. Finally, define the* size *of BP to be* $|V|$*, the number of vertices in* G*.*

The better known model of *deterministic* branching programs may be defined as the special case in which for every input x, the out-degree of every vertex in G_x is at most 1 (note that in this case, the choice of p does not make a difference, and it can be fixed to be as small as 2). A definition of standard *non-deterministic* branching programs may be obtained from the above by counting the number of accepting paths over the integers, rather than modulo p. Thus, when p is larger than the possible number of paths, mod-p computation and non-deterministic computation coincide (see [34, 8] for better simulations of non-deterministic branching programs by modular branching programs).

In contrast to constant-degree polynomials, branching programs are quite powerful. In particular, every function can be computed by a branching program, and many "natural" function families (in particular regular languages) by linear-size branching programs. The branching program size of any function f is no larger than the *Boolean formula* size of f.[6] However, as the power of (uniformly constructible) polynomial-size branching programs coincides with that of different variants of *log-space* computation, they are probably not powerful enough to efficiently compute all polynomial-time computable functions.

3. Constructing Low-Degree Randomizing Polynomials

Generalizing Example 2.3 from the previous section, any linear function over $\mathcal{K}$ is trivially evaluated (hence also randomized) by degree-1 polynomials. A more interesting construction is obtained by the following easy generalization of Example 2.4.

[6] The parity function provably demonstrates a quadratic separation between the formula complexity and the branching program complexity. Assuming the existence of functions in NL $\setminus$ NC^1, the separation is superpolynomial.

Theorem 3.1 *For a matrix* $A \in \mathcal{K}^{w,n}$ *and vector* $b \in \mathcal{K}^w$*, the Boolean function* $f_{A,b}$ *testing whether* $Ax = b$ *can be randomized by degree-2 polynomials, with output complexity 1 and randomness complexity* w*.*

Proof. The function $f_{A,b}$ is randomized by the polynomial vector $p(x, r) = r^t(Ax - b)$. Indeed, if $Ax = b$ then $p(x, r)$ is deterministically zero, and otherwise it is uniformly distributed over $\mathcal{K}$. □

Some useful special cases of Theorem 3.1 are the OR function, the AND function, and testing equality between two or more substrings of the input. For instance, the case of the AND function is obtained by letting $\mathcal{K} = \mathrm{GF}(2)$, $A = I_n$, and $b = (1, 1, \ldots, 1)$.

In contrary to what one might expect, being able to randomize the unbounded fan-in AND function using degree-2 polynomials does not imply that degree-2 polynomials are sufficient to randomize all functions. While standard polynomial representations of f_1, f_2 can always be added together to obtain a representation of $f_1 + f_2$, this is not the case for randomizing polynomials. In Section 5 we show that, indeed, randomizing degree-2 polynomials are not enough to represent all Boolean functions.

We now turn to our main construction, of degree-3 randomizing polynomials, which may apply to any Boolean function f. In the heart of our construction lies the following reduction: given a modular branching program of size $\ell + 1$ computing f, an input x can be transformed via a simple *affine transformation* into an $\ell \times \ell$ matrix M_x,[7] such that the output value $f(x)$ directly corresponds to the rank of M_x. This is formalized by the following lemma, which is implicit in [22] (a similar, though less efficient, transformation can be derived from Lemma 2.3 in [1]). A proof of the lemma appears in Appendix A.

Lemma 3.2 *[22] Let* $\mathcal{K} = \mathrm{GF}(p)$*, where* p *is prime, and suppose that BP is a mod-*p *branching program of size* $\ell + 1$ *computing a Boolean function* $f : \{0,1\}^n \to \{0,1\}$*. Then, there exists an affine (degree-1) transformation* $L : \mathcal{K}^n \to \mathcal{K}^{\ell,\ell}$ *such that for any* $x \in \{0,1\}^n$*:*

- *If* $f(x) = 1$ *then the matrix* $M_x \stackrel{\text{def}}{=} L(x)$ *is of full rank (i.e.,* $\mathrm{rank}(M_x) = \ell$*).*
- *If* $f(x) = 0$ *then the rank of* M_x *is one less than full (i.e.,* $\mathrm{rank}(M_x) = \ell - 1$*).*

Next, we show that multiplying a matrix M by two independently random matrices, one from the left and one from the right, produces a random matrix whose distribution depends only on $\mathrm{rank}(M)$.

Lemma 3.3 *Let* R_1, R_2 *be two independent random matrices, each uniformly distributed over* $\mathcal{K}^{\ell,\ell}$*. Then, for any* $M, M' \in \mathcal{K}^{\ell,\ell}$ *such that* $\mathrm{rank}(M) = \mathrm{rank}(M')$*, we have:* $R_1 M R_2 \equiv R_1 M' R_2$*.*

[7] More specifically, each entry of M_x can be taken to be a degree-1 polynomial in a *single* input variable x_i (or a constant).

Proof. First observe that since $\mathrm{rank}(M) = \mathrm{rank}(M')$ we can write $M = Q_1 M' Q_2$ where both Q_1, Q_2 are invertible matrices in $\mathcal{K}^{\ell,\ell}$. (This follows from the fact that every matrix of degree ρ can be transformed via elementary row and column operations into some canonic matrix C_ρ; hence we can write $C_\rho = QMQ'$ for some invertible Q, Q' and the observation follows.) Now, since a random matrix times an invertible matrix yields a random matrix, we have: $R_1 M R_2 \equiv (R_1 Q_1) M' (Q_2 R_2) \equiv R_1 M' R_2$ and the lemma follows. □

Lemma 3.4 *Let* R_1, R_2 *be distributed as above, and suppose that* $\mathrm{rank}(M) > \mathrm{rank}(M')$. *Then,* $\mathrm{SD}(R_1 M R_2, R_1 M' R_2) > 0.08$.

Proof. Let $\rho = \mathrm{rank}(M)$. We argue that a distinguisher A which outputs 1 iff its input matrix is of rank ρ has the required advantage. Note that $\mathrm{rank}(R_1 M' R_2) \leq \mathrm{rank}(M')$, implying that

$$\Pr[A(R_1 M' R_2) = 1] = 0. \tag{1}$$

On the other hand, conditioned on the event that both R_1 and R_2 happen to be invertible, we have $\mathrm{rank}(R_1 M R_2) = \mathrm{rank}(M)$ implying that $A(R_1 M R_2) = 1$. The probability of a random matrix over a finite field being invertible is greater than 0.288.[8] Hence:

$$\Pr[A(R_1 M R_2) = 1] > 0.288^2 > 0.08. \tag{2}$$

Combining equations (1) and (2), the distinguishing advantage of A is larger than 0.08, as required. □

Combining the previous two lemmas, we get:

Theorem 3.5 *Let* $f : \{0,1\}^n \to \{0,1\}$ *be a Boolean function. If* f *can be computed by a mod-*p *branching program of size* ℓ*, where* p *is a prime, then it can be randomized by degree-3 polynomials over* $\mathcal{K} = \mathrm{GF}(p)$ *with complexity* $O(\ell^2)$.

Proof. Consider the polynomial vector

$$p(x, R_1, R_2) = R_1 L(x) R_2.$$

p has degree 3, output complexity $(\ell-1)^2$ and randomness complexity $2(\ell-1)^2$. By Lemmas 3.2, 3.3 it satisfies the privacy property. By Lemma 3.4 the statistical distance between the induced output distributions D_0 and D_1 is lower-bounded by some constant. Hence, a constant number of repetitions of p suffices to amplify the statistical distance between D_0 and D_1 to the level required by the definition. □

Remark 3.6 The output distributions of the randomizing polynomials of Theorem 3.5 can be *efficiently* distinguished with a constant *one-sided* by a distinguisher A which output 1 iff its inputs form an invertible matrix.

[8] This constant reflects the case $\mathcal{K} = \mathrm{GF}(2)$; over a larger field, the probability of getting an invertible matrix is even higher.

4. Application to Round-Efficient Secure Computation

In this section we show how to securely evaluate a function $f(x)$ by using its randomizing polynomials representation $p(x, r)$ and *any* secure protocol for evaluating polynomials of the same degree as p. In the following we refer to the notion of *privacy* defined in Section 2.2, but the same ideas apply to any reasonable notion of security.

In a nutshell, our means for obtaining the above goal is a concatenation of two reductions: the ϵ-correct private computation of $f(x)$ reduces to a perfectly-correct private computation of the *randomized* function $P(x)$, which in turn reduces to a perfectly-correct private computation of a vector of *deterministic* polynomials of the same degree as p. While all relevant arguments are rather standard (in particular, the second reduction may be viewed as a special case of a reduction from [20]) we prefer a completely self-contained treatment, which does not rely on any additional definitions.

Our starting point is the following observation. Suppose that we could sample an element z from the distribution $P(x)$, and let all parties learn the value of z alone; then, by the privacy property of randomizing polynomials, the parties learn nothing except $f(x)$. By the correctness property of randomizing polynomials, all parties can reconstruct $f(x)$ from z with high probability.

The next observation is that to emulate the above random sampling process by a multi-party protocol, the random inputs r should be generated by the parties; however, this should be done while keeping r secret. Learning *both* the randomized output and the random inputs r may give more information than just $f(x)$. For instance, in Example 2.4 from Section 2.1, knowledge of both $p(x, r)$ and r gives information on whether or not the OR of certain subsets of the variables is 1.

To achieve this goal, without assuming anything about the particular protocol employed for securely evaluating low-degree polynomials, we define a new k-argument function $f'(y'_1, \ldots, y'_k)$ as follows. Suppose that $p(x, r)$ is a degree-d polynomial vector which randomizes f with ϵ-correctness. Let y_j denote the inputs to f held by party $\mathcal{P}_j$ (i.e., y_j is a vector of elements from $\mathcal{K}$, and $y_1, \ldots, y_k$ include a total of n field elements). Moreover, let r_j, $1 \leq j \leq k$, be a vector of input variables of length m, where m is the randomness complexity of p. (Note that here r_j is not a single entry of r, but rather each r_j is a "copy" of r on a disjoint set of variables). Now, let $y'_j = (y_j, r_j)$ for $1 \leq j \leq t+1$, and $y'_j = y_j$ for $t+1 \leq j \leq k$, where t is the desirable privacy threshold, and define:

$$f'(y'_1, \ldots, y'_k) \stackrel{\mathrm{def}}{=} p(x, r_1 + \ldots + r_{t+1}).$$

Note that f' may be evaluated by first applying a *linear* (degree 1) transformation, mapping its inputs to $(x, r_1 + \ldots + r_{t+1})$, and then applying p to the result; hence the degree of f' is d, the same as that of p. Moreover, f' has the same number of outputs as p, and a total number of inputs which is comparable to that of p. We now argue that a private evaluation of f reduces to a private evaluation of the degree-d polynomial vector f'; that is, *any* t-private perfectly-correct protocol for computing f' immediately translates into a t-private ϵ-correct protocol for computing f. Let $\mathcal{F}'$ be a t-private k-party protocol for computing f'. Define a protocol $\mathcal{F}$ for computing f as follows:

1. Each of the first $t+1$ parties $\mathcal{P}_j$ picks a random vector $r_j \in \mathcal{K}^m$, and lets $y'_j = (y_j, r_j)$; the remaining parties $\mathcal{P}_j, t+2 \leq j \leq k$, let $y'_j = y_j$.
2. The parties invoke $\mathcal{F}'$, where each party $\mathcal{P}_j$ uses y'_j as its private input; at the end of the interaction, all parties learn a single output vector $z = f'(y')$.
3. Each party applies a distinguisher A between D_0 and D_1 to z (such that $\Pr[A(D_1) = 1] - \Pr[A(D_0) = 1] \geq 1 - \epsilon$, and $\Pr[A(D_0) = 1] = 0$ in the one-sided case) and outputs $A(z)$.

Theorem 4.1 *The protocol $\mathcal{F}$ described above is a t-private ϵ-correct protocol for computing f. Furthermore, if p is ϵ-correct with a one-sided error then so is $\mathcal{F}$.*

Proof. See Appendix A. □

In different security settings than the one considered here, the randomizing polynomials p and their output distributions should meet additional computational efficiency requirements. However, the same reasoning as above applies in these different settings as well. Combining Theorem 4.1, Theorem 3.5, and Remark 3.6, we obtain:

Corollary 4.2 *Suppose that $f : \{0,1\}^n \to \{0,1\}$ can be computed by a mod-p branching program of size ℓ. Then, a t-private k-party evaluation of f (with a constant one-sided error) is no more expensive than a t-private k-party evaluation of $O(\ell^2)$ degree-3 polynomials on $n + O(t\ell^2)$ inputs over* $\mathrm{GF}(p)$.

Finally, we rely on well-known techniques for privately evaluating polynomials to obtain our improved constant-round protocols. Optimized variants of standard information-theoretically private protocols (e.g. [16], or [9] with optimization from [18]) may be used to obtain:

Lemma 4.3 *A degree-3 polynomial vector with n inputs and s outputs over a field $\mathcal{K}$ can be computed t-privately by k parties with either:*

- *2 rounds, $t = \lfloor \frac{k-1}{3} \rfloor$, and communication of $O(k(n + ks))$ field elements;*[9]
- *3 rounds, (optimal) privacy threshold of $t = \lfloor \frac{k-1}{2} \rfloor$, and communication of $O(k^2(n^2 + s))$ field elements.*

Moreover, for the specific degree-3 polynomials constructed in Section 3, the 3-round communication bound can be brought down to $O(k(n + ks))$ as well.

[9] This assumes that $|\mathcal{K}| > k$; otherwise, one can use a sufficiently large extension field of $\mathcal{K}$, paying at most a $\log k$ multiplicative factor.

Protocols realizing Lemma 4.3 are described in Appendix A. Combining Lemma 4.3 with Corollary 4.2 we get the following concrete application of randomizing polynomials.

Corollary 4.4 *Let f be a k-argument function which can be computed by a mod-p branching program of size ℓ. Then, two (respectively, three) communication rounds are sufficient to $\lfloor \frac{k-1}{3} \rfloor$-privately (resp., $\lfloor \frac{k-1}{2} \rfloor$-privately) compute f with a one-sided error. This can be done while communicating $O(k^2\ell^2 \log 1/\epsilon)$ field elements ($O(\log(\max\{k, p\}))$-bits each), where ϵ is the one-sided error probability.*

Remark 4.5 Instead of ϵ-correct protocols, one can obtain zero-error $\lfloor \frac{k-1}{3} \rfloor$-private (resp., $\lfloor \frac{k-1}{2} \rfloor$-private) protocols with *expected* $2 + \epsilon$ (resp., $3 + \epsilon$) rounds, for an arbitrarily small $\epsilon > 0$. One way of achieving this is by repeatedly running one-sided $O(\epsilon)$-correct protocols for f and its complement in parallel, until a conclusive answer is obtained.

5. A Negative Result for Degree-2 Randomizing Polynomials

In this section we characterize the exact power of degree-2 randomizing polynomials, and in particular prove that they cannot represent all functions. Combining this with the results of Section 3, we conclude that 3 is the minimal degree which is sufficient for randomizing all functions. For simplicity of exposition, we address here only the case $\mathcal{K} = \mathrm{GF}(2)$. An outline of the proof for general finite fields, which involves some additional technical complications, is given in Appendix A. We start by defining the following natural relaxation of randomizing polynomials.

Definition 5.1 *A polynomial vector $p(x, r)$ is said to* weakly randomize *a Boolean function $f : \{0,1\}^n \to \{0,1\}$ if there exist* distinct *probability distributions D_0, D_1 such that for any input $x \in \{0,1\}^n$, $P(x) \equiv D_{f(x)}$.*

Note that the usual notion of randomizing polynomials strengthens the above by requiring D_0 and D_1 to be statistically far from each other. We use the following basic fact about probability distributions. A proof for the case $\mathcal{K} = \mathrm{GF}(2)$ appears in [13].

Fact 5.2 *Let $D_0 = (D_0^1, \ldots, D_0^s), D_1 = (D_1^1, \ldots, D_1^s)$ be distinct distributions over $\mathrm{GF}(2)^s$. Then, there exists a vector $w \in \mathrm{GF}(2)^s$ such that $\sum_{j=1}^s w_j D_0^j \not\equiv \sum_{j=1}^s w_j D_1^j$ (where summations are carried over $\mathrm{GF}(2)$).* □

The next lemma, which follows easily from Fact 5.2, allows to focus on the expressiveness of a *single* weakly randomizing polynomial, rather than that of a *vector* of polynomials.

Lemma 5.3 *Let f be a Boolean function. If f can be randomized by a vector of degree-d polynomials, then it can be weakly randomized by a single degree-d polynomial.*

Definition 5.4 *Let $\mathcal{C}_n^{\mathcal{K}}$ be the class of Boolean functions $f : \{0,1\}^n \to \{0,1\}$ that satisfy either of the following:*

1. *f or its negation test a linear condition over $\mathcal{K}$ (as in Theorem 3.1). That is, either $f^{-1}(0)$ or $f^{-1}(1)$ are equal to the intersection of some affine subspace of $\mathcal{K}^n$ with $\{0,1\}^n$.*
2. *f can be evaluated by a deterministic degree-2 polynomial.*

In the rest of the section we characterize the class of Boolean functions which can be weakly randomized by a degree-2 polynomial, and prove that it coincides with $\mathcal{C}_n^{\mathcal{K}}$. We start with degree-2 polynomials of a restricted form. The next lemma and its proof apply to any field $\mathcal{K}$.

Lemma 5.5 *Suppose that $f : \{0,1\}^n \to \{0,1\}$ is weakly randomized by a degree-2 polynomial $p(x,r)$ over $\mathcal{K}$ whose canonical representation does not include monomials of the form $r_i r_j$. Then $f \in \mathcal{C}_n^{\mathcal{K}}$.*

Proof. Write p in the form: $p(x,r) = r_1 L_1(x) + r_2 L_2(x) + \ldots + r_m L_m(x) + q(x)$, where each $L_j(x)$ is a degree-1 polynomial in x (e.g., $x_1 + x_3 + 1$), and $q(x)$ is a degree-2 polynomial in x. Now, consider the set of linear equations:

$$L_j(x) = 0, \quad j = 1, 2, \ldots, m. \tag{3}$$

Recall that $P(x)$ denotes the probability distribution of $p(x,r)$, induced by a uniform choice of r from $\mathcal{K}^m$. Observe that if x does not solve (3), then $P(x)$ is uniform over $\mathcal{K}$. Otherwise, $P(x)$ is deterministically equal to $q(x)$. Now, consider the following cases.

1. At most one x in $\{0,1\}^n$ solves (3): then f tests for a linear condition;
2. Every $x \in \{0,1\}^n$ solves (3): then f is equal to q, and hence can be evaluated by a degree-2 polynomial.
3. Otherwise (i.e., at least two but not all x in $\{0,1\}^n$ solve (3)), let S be the set of x's in $\{0,1\}^n$ solving (3), and consider the two subcases:
   (a) q has the same value on every $x \in S$: in that case, f determines whether $x \in S$, which again amounts to testing for a linear condition.
   (b) q has different values on S: in this case, there are at least three possible distributions $P(x)$ (the uniform one, corresponding to an $x \notin S$, and two different distributions with a singleton support). This contradicts the assumption that f is Boolean. □

It remains to show that monomials of the form $r_i r_j$ can also be handled. The next lemma applies to any field $\mathcal{K}$.

Lemma 5.6 *Suppose that $p(x, r_1, r_2, \ldots, r_m)$ is a degree-2 polynomial over $\mathcal{K}$, whose representation as a sum of monomials includes $r_1 r_2$ (with a nonzero coefficient) but not r_1^2. Then, there exists a degree-2 polynomial $p'(x, r_3, \ldots, r_m)$ over $\mathcal{K}$ which weakly randomizes the same function as p.*

Proof. Assume without loss of generality that the coefficient of the monomial $r_1 r_2$ in p is 1. By the assumptions on p, it may be written in the form:

$$\begin{aligned} p(x, r_1, r_2, \ldots, r_m) &= r_1(r_2 - L(x, r_3, \ldots, r_m)) + \\ &\quad q(x, r_2, r_3, \ldots, r_m) \end{aligned} \tag{4}$$

where L is a degree-1 polynomial and q is a degree-2 polynomial. Now, define

$$p'(x, r_3, \ldots, r_m) = q(x, L(x, r_3, ..., r_m), r_3, ..., r_m). \tag{5}$$

Note that p' is also a degree-2 polynomial. We claim that p' weakly randomizes the same function as p. Fix $x \in \{0,1\}^n$, and consider the joint random variables $P(x), P'(x)$ induced by the uniform choice of $(r_1, r_2, \ldots, r_m)$ from $\mathcal{K}^m$. Let A denote the event that $r_2 = L(x, r_3, \ldots, r_m)$. The following facts are easy to verify:

1. Given A, the random variables $P(x)$ and $P'(x)$ are always identical (since the first term in the r.h.s. of (4) vanishes, and the expression substituted for r_2 in (5) is equal to r_2);
2. Given $\overline{A}$, $P(x)$ is uniformly distributed over $\mathcal{K}$ (since so is $r_1(r_2 - L(x, r_3, \ldots, r_m))$ and since q does not depend on r_1);
3. $\Pr[A] = 1/|\mathcal{K}|$;
4. $P'(x)$ is independent of A (since p' does not depend on r_2).

Using the above facts, we have that for any $y \in \mathcal{K}$,

$$\begin{aligned} \Pr[P(x) = y] &= \Pr[A] \cdot \Pr[P(x) = y \mid A] + \\ &\quad \Pr[\overline{A}] \cdot \Pr[P(x) = y \mid \overline{A}] \\ &= (1/|\mathcal{K}|) \cdot \Pr[P'(x) = y \mid A] + \\ &\quad (1 - 1/|\mathcal{K}|) \cdot (1/|\mathcal{K}|) \\ &= (1/|\mathcal{K}|) \cdot \Pr[P'(x) = y] + \\ &\quad (1 - 1/|\mathcal{K}|) \cdot (1/|\mathcal{K}|). \end{aligned}$$

Hence, $\Pr[P(x) = y]$ may be obtained by applying a fixed invertible transformation to $\Pr[P'(x) = y]$ (intuitively, the distribution $P(x)$ is a "flattened" version of $P'(x)$). It follows that for any $x, x' \in \{0,1\}^n$, $P(x) \equiv P(x')$ iff $P'(x) \equiv P'(x')$, implying that p' weakly randomizes the same function as p. □

We are now ready to prove the main characterization result, for general degree-2 polynomials over $\mathrm{GF}(2)$.

Theorem 5.7 *Suppose that $p(x,r)$ is a degree-2 polynomial over $\mathcal{K} = \mathrm{GF}(2)$ which weakly randomizes a Boolean function $f : \{0,1\}^n \to \{0,1\}$. Then $f \in \mathcal{C}_n^{\mathcal{K}}$.*

Proof. We proceed by induction on m, the randomness complexity of p. The case $m = 0$ is immediate (i.e., f satisfies condition 2 in the definition of $C_n^{\mathcal{K}}$). Since $\mathcal{K} = \mathrm{GF}(2)$, the standard representation of p includes no monomials of the form r_i^2 (each such monomial can be simply replaced by r_i). Now, if p does not include a monomial of the form $r_i r_j$, then Lemma 5.5 applies. Otherwise, by appropriately renaming the random variables, Lemma 5.6 implies the existence of a degree-2 polynomial p' with $m-2$ random variables which weakly randomizes f as well. By the induction's hypothesis, $f \in \mathcal{C}_n^{\mathcal{K}}$. $\square$

Note that the above proof crucially relies on the fact that over $\mathrm{GF}(2)$, monomials of the form r_i^2 can be trivially eliminated. In Appendix A we show how to deal with such monomials over general fields.

For completeness, we include the general version of Theorem 5.7, whose proof is sketched in Appendix A.

Theorem 5.8 *Suppose that $p(x,r)$ is a degree-2 polynomial over a finite field $\mathcal{K}$, which weakly randomizes a Boolean function $f : \{0,1\}^n \to \{0,1\}$. Then $f \in \mathcal{C}_n^{\mathcal{K}}$.*

Using Theorem 3.1 and Lemma 5.3, we get a precise characterization of the power of randomizing degree-2 polynomials.

Corollary 5.9 *The class of functions $f : \{0,1\}^n \to \{0,1\}$ which can be randomized by degree-2 polynomials over a finite field $\mathcal{K}$ coincides with $\mathcal{C}_n^{\mathcal{K}}$. Hence, "most" Boolean functions cannot be randomized by degree-2 polynomials over any finite field.*

6. Concluding Remarks

Randomizing polynomials introduce a simple and powerful paradigm for the design of round-efficient secure computation protocols. A long-standing open question is whether every polynomial-time computable function can be computed by an *efficient* constant-round protocol with information-theoretic privacy and linear privacy threshold [6, 2, 5]. This question is open even in a setting where the parties may be computationally unbounded. While general interactive protocols are difficult to analyze, there seems to be much better hope for characterizing the power of the more restricted model of randomizing polynomials, for example by relating it to well-studied complexity models. Gaining a better understanding of randomizing polynomials may be a first step towards resolving the above question.

An open question of a somewhat different flavor is that of finding the *exact* number of rounds required for privately evaluating an arbitrary (i.e., a worst-case) function f with an *optimal* privacy threshold. Using randomizing polynomials, an upper bound of 3 was obtained. If this bound is tight (i.e., 2 rounds are not enough) then, in a very crude sense, the randomizing polynomials approach is non-restrictive.

Finally, there are several natural variants of the randomizing polynomials model which were not studied in this work. These include a statistically-private relaxation and a perfectly-correct restriction of the model.

Acknowledgments: We wish to thank Richard Beigel, Ivan Damgård, Nick Howgrave-Graham, and Dieter van-Melkebeek for helpful discussions and suggestions.

References

[1] E. Allender, R. Beals, and M. Ogihara. The complexity of matrix rank and feasible systems of linear equations. In *Proc. of 28nd STOC*, pages 503–513, 1996.

[2] J. Bar-Ilan and D. Beaver. Non-cryptographic fault-tolerant computing in a constant number of rounds. In *Proc. of 8th PODC*, pages 201–209, 1989.

[3] D. Barrington, R. Beigel, and S. Rudich. Representing Boolean functions as polynomials modulo composite numbers. In *Proc. of 24th STOC*, pages 455–461, 1992.

[4] D. Beaver. Minimal-latency secure function evaluation. In *Proc. EUROCRYPT 2000*, pages 335–350.

[5] D. Beaver, J. Feigenbaum, J. Kilian, and P. Rogaway. Security with low communication overhead. In *Proc. of CRYPTO '90*, pages 62–76.

[6] D. Beaver, S. Micali, and P. Rogaway. The round complexity of secure protocols (extended abstract). In *Proc. of 22nd STOC*, pages 503–513, 1990.

[7] R. Beigel. The polynomial method in circuit complexity. In *Proc. of structure in complexity theory*, pages 82–95, 1993.

[8] A. Beimel and A. Gal. On arithmetic branching programs. In *Proc. of the 13th Annu. IEEE Conf. on Computational Complexity*, pages 68–80, 1998.

[9] M. Ben-Or, S. Goldwasser, and A. Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation. In *Proc. of 20th STOC*, pages 1–10, 1988.

[10] C. Cachin, J. Camenisch, J. Kilian, and J. Muller. One-round secure computation and secure autonomous mobile agents. In *ICALP 2000*.

[11] R. Canetti. Security and composition of multi-party cryptographic protocols. *J. Cryptology*, 13(1):143–202, 2000.

[12] D. Chaum, C. Crépeau, and I. Damgård. Multiparty unconditionally secure protocols (extended abstract). In *Proc. of 20th STOC*, pages 11–19, 1988.

[13] B. Chor, O. Goldreich, J. Hastad, J. Friedman, S. Rudich, and R. Smolensky. The bit extraction problem of t-resilient functions (preliminary version). In *Proc. of 26th FOCS*, pages 396–407, 1985.

[14] R. Cramer and I. Damgård. Secure distributed linear algebra in a constant number of rounds. Manuscript, 2000.

[15] R. Cramer, I. Damgård, S. Dziembowski, M. Hirt, and T. Rabin. Efficient multiparty computations secure against an adaptive adversary. In *Proc. EUROCRYPT '99*, pages 311–326.

[16] R. Cramer, I. Damgård, and U. Maurer. General secure multi-party computation from any linear secret-sharing scheme. In *Proc. of EUROCRYPT 2000*, pages 316–334.

[17] U. Feige, J. Kilian, and M. Naor. A minimal model for secure computation (extended abstract). In *Proc. of 26th STOC*, pages 554–563, 1994.

[18] R. Gennaro, M. O. Rabin, and T. Rabin. Fact-track multiparty computations with applications to threshold cryptography. In *Proc. of 17th PODC*, pages 101–111, 1998.

[19] O. Goldreich. *Modern Cryptography, Probabilstic Methods and Pseudo-Randomness*. Springer-Verlag, 1999.

[20] O. Goldreich. Secure multi-party computation (working draft). Manuscript, 2000.

[21] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game (extended abstract). In *Proc. of 19th STOC*, pages 218–229, 1987.

[22] Y. Ishai and E. Kushilevitz. Private simultaneous messages protocols with applications. In *Proc. of 5th ISTCS*, pages 174–183, 1997.

[23] J. Kilian. Founding cryptography on oblivious transfer. In *Proc. of 20th STOC*, pages 20–31, 1988.

[24] R. Lipton. New directions in testing. In *Proc. of DIMACS workshop on distributed computing and cryptography*, 1989.

[25] S. Micali and P. Rogaway. Secure computation. In *Proc. of CRYPTO '91*, pages 392–404.

[26] N. Nisan and M. Szegedy. On the degree of Boolean functions as real polynomials. *Computational Complexity*, 4:301–313, 1994.

[27] T. Rabin and M. Ben-Or. Verifiable secret sharing and multiparty protocols with honest majority. In *Proc. of 21st STOC*, pages 73–85, 1989.

[28] A. A. Razborov. Lower bounds for the size of circuits of bounded depth with basis $\{\wedge, \oplus\}$. *Math. notes of the Academy of Science of the USSR*, 41(4):333–338, 1987.

[29] T. Sandler, A. Young, and M. Yung. Non-interactive cryptocomputing for NC^1. In *Proc. of 40th FOCS*, pages 554–566, 1999.

[30] A. Shamir. How to share a secret. *Commun. ACM*, 22(6):612–613, June 1979.

[31] R. Smolensky. Algebraic methods in the theory of lower bounds for Boolean circuit complexity. In *Proc. of 19th STOC*, pages 77–82, 1987.

[32] J. Tarui. Randomized polynomials, threshold circuits, and the polynomial hierarchy. In *Proc. of 8th Symp. on Theoretical Aspects of Computer Science*, pages 238–250, 1991.

[33] S. Tsai. Lower bounds on representing Boolean functions as polynomials in Z_m. In *Proceedings of the 8th Annual Structure in Complexity Theory*, pages 96–101, 1993.

[34] A. Wigderson. $\mathrm{NL}/poly \subseteq \oplus\mathrm{L}/poly$. In *Proc. of the 9th Annual Structure in Complexity Theory*, pages 59–62, 1994.

[35] A. C. Yao. Protocols for secure computations (extended abstract). In *Proc. of 23rd FOCS*, pages 160–164, 1982.

[36] A. C. Yao. How to generate and exchange secrets. In *Proc. of 27th FOCS*, pages 162–167, 1986.

A. Proofs

Proof of Lemma 3.2.

Let $BP = (G, \phi, s, t)$ be a branching program of size $\ell + 1$ computing f. For any input assignment x, let H_x be the $(\ell+1) \times (\ell+1)$ adjacency matrix of the induced subgraph G_x. The number of $s-t$ paths in G_x modulo p can then be presented as

$$(I + H_x + H_x^2 + \cdots)_{s,t} = (I - H_x)^{-1}_{s,t}$$

where all matrices are over $\mathcal{K} = \mathrm{GF}(p)$. (In particular, when G_x is acyclic, the expansion of $(I - H_x)^{-1}$ has finitely many nonzero terms, and $I - H_x$ is invertible.) Now, let M_x denote the submatrix of $I - H_x$ obtained by deleting the s-th column and t-th row. By a well known formula for computing matrix inverse,

$$(I - H_x)^{-1}_{s,t} = \frac{\det M_x}{\det(I - H_x)}.$$

Hence, the number of accepting paths is nonzero modulo p if and only if $\det(M_x)$ is nonzero, or equivalently the matrix M_x is of full rank over $\mathcal{K}$. Also note that since $I - H_x$ is invertible and M_x is obtained from it by removing a single row and a single column, the rank of M_x can never be smaller than $(\ell + 1) - 2 = \ell - 1$, and in particular it must be exactly $\ell - 1$ when it is not full.

Finally, since each entry of M_x is a degree-1 polynomial in the input variables, we have that the mapping $L(x) \stackrel{\text{def}}{=} M_x$ satisfies the required properties. □

Proof of Theorem 4.1.

The (probabilistic) correctness of $\mathcal{F}$ follows immediately from the observation that the intermediate output z resulting from the invocation of $\mathcal{F}'$ is distributed identically to $D_{f(x)}$.

To argue that $\mathcal{F}$ is t-private, consider the view of a (passive) adversary corrupting a set of parties B, where $|B| \leq t$. This view includes the inputs $y'_B \stackrel{\text{def}}{=} (y'_j)_{j \in B}$ (including the original inputs y_B and the randomly chosen vectors r_B), and additional messages sent during the invocation of $\mathcal{F}'$.

The intuition is very simple: since the adversary can learn nothing on $r = r_1 + \ldots + r_{t+1}$ from its partial view r_B, it has no advantage over learning $f'(y') = p(x, r)$ alone, which gives no additional information other than the value of f.

To formalize this intuition, we need to show that the adversary's entire view can be *simulated* based on y_B and $f(y)$ alone (without knowing the inputs of parties not in B). Such a simulator, on inputs $y_B, f(y)$, may proceed as follows:

- sample a string z from $D_{f(y)}$;
- sample a $|B|$-tuple r_B of independent vectors from $\mathcal{K}^m$;
- invoke a simulator for $\mathcal{F}'$ on inputs (y_B, r_B) and output z to produce a view v;
- output (y_B, r_B, v).

We omit an analysis of the correctness of the above simulator. □

Proof Sketch of Lemma 4.3.

We first address the case $k > 3t$, describing a 2-round protocol for t-privately evaluating a *single* degree-3 polynomial $p(x)$, where the input $x \in \mathcal{K}^n$ is arbitrarily partitioned among the k parties.

Round 1. Each party shares each of its inputs among the k parties, using t-private Shamir's secret-sharing [30] (this is where it is required that $|\mathcal{K}| > k$). In addition, the parties distribute random "pads" to allow a non-interactive private computation of linear functions in the second round. Specifically, let each of the first $t+1$ parties "additively share 0" among all parties (i.e., send independent random field elements to all other parties, and keep the negation of their sum as their own share).

Round 2. Each party locally evaluates the polynomial p on the input shares it received. By the homomorphic property of Shamir's scheme, the k local outputs lie on a degree-$3t$ polynomial with free coefficient $p(x)$. Hence $p(x)$ can be reconstructed by applying a fixed linear functional to the local outputs. A private computation of a linear functional can be easily realized using the random pads distributed in Round 1: Each party first multiplies its share of $p(x)$ by the appropriate reconstruction coefficient, then adds the $t+1$ pads it received in Round 1, and sends the result to all other parties.

Output. Each party locally outputs the sum of the k field elements received in Round 2.

To evaluate a vector of s degree-3 polynomials the protocol is essentially repeated s times in parallel, once for each polynomial, except that each input should only be shared once. The entire communication includes $O((k(n+ks))$ field elements.

A 3-round protocol for the case $k > 2t$ is obtained by letting Round 2 produce shares of intermediate degree-2 polynomials, and Round 3 combine the intermediate shares with the input shares to yield the final degree-3 polynomials. The communication complexity will depend on the number of intermediate degree-2 polynomials which have to be computed, which is always less than n^2 (since it clearly suffices to use all monomials $x_i x_j$ as intermediate values). Substituting $s + n^2$ for s in the complexity of the 3-round protocol, we get a worst-case communication of $O(k^2(n^2+s))$ field elements. Luckily, for the specific degree-3 polynomial vectors constructed in Section 3 (and their modification f' described above), only $O(s)$ intermediate values have to be stored; specifically, since the polynomials are defined by a product of three matrices, the intermediate values can consist of the entries of the product of the first two matrices. Hence, it is possible to evaluate these degree-3 polynomials communicating only $O(k(n+ks))$ field elements. □

Proof Sketch of Theorem 5.8.

We start by making two simplifying observations. A first observation is that it suffices to deal with fields of a prime order. Viewing an extension field $\mathcal{K}' = \mathrm{GF}(p^w)$ as a vector space over $\mathcal{K} = \mathrm{GF}(p)$, a degree-2 polynomial over $\mathcal{K}'$ can be "simulated" by a vector of degree-2 polynomials over $\mathcal{K}$. A second observation is that Fact 5.2 generalizes to finite fields of a prime order, as follows from the invertibility of the Discrete Fourier Transform.

From the above two observations, we may conclude the following generalization of Lemma 5.3: If f can be randomized by a vector of degree-d polynomials over $\mathrm{GF}(p^w)$, where p is prime, then it can be weakly randomized by a single degree-d polynomial over $\mathrm{GF}(p)$. It thus remains to deal with a *single* randomizing polynomial $p(x, r)$ over a field $\mathcal{K}$ of an odd prime order.

The proof of Theorem 5.7 does not show how to deal with monomials of the form r_i^2; moreover, the random input elimination procedure described in the proof of Lemma 5.6 may create such monomials even if they did not exist in p. To make the proof go through, we replace the original induction step as follows. Without loss of generality, p may be written as

$$p(x, r) = r_1 L(x, r) + q(x, r_2, \ldots, r_m)$$

where L is a degree-1 polynomial which depends on at least one r_i, and q is a degree-2 polynomial. Now, if L does not depend on r_1, we may proceed as in the original proof. Otherwise we may write

$$\begin{aligned} p(x, r) &= (r_1 + L'(x, r_2, \ldots, r_m))^2 + \\ &\quad q'(x, r_2, \ldots, r_m) \end{aligned} \qquad (6)$$

where $L' = (L - r_1)/2$ is a degree-1 polynomial and $q' = q - L'^2$ is a degree-2 polynomial (note that division by 2 is well defined, since the order of the field is odd). Since neither L' nor q' depend on r_1, it may be deduced from (6) that $P(x)$ is identically distributed to $P'(x)$ where

$$p'(x, r) = r_1^2 + q'(x, r_2, \ldots, r_m).$$

Since q' contains one less random variable than p, to complete the induction step it suffices to show that p' weakly randomizes the same function as q'. This follows from the following lemma, whose proof is omitted from this version:

Lemma A.1 *Let $\mathcal{K}$ be a field of prime order, and let $\mathcal{D}_{\mathcal{K}}$ be the set of all probability distributions over $\mathcal{K}$. Define a mapping $\mu : \mathcal{D}_{\mathcal{K}} \to \mathcal{D}_{\mathcal{K}}$ as follows. On input D, μ returns the distribution obtained by adding a field element randomly selected according to D with the square of an independent random field element. Then, the mapping μ is injective.* □