Secure Computation of Constant-Depth Circuits with Applications to Database Search Problems*

Omer Barkol and Yuval Ishai

Computer Science Department, Technion
{omerb,yuvali}@cs.technion.ac.il

Abstract. Motivated by database search problems such as partial match or nearest neighbor, we present secure multiparty computation protocols for constant-depth circuits. Specifically, for a constant-depth circuit C of size s with an m-bit input x, we obtain the following types of protocols.

- In a setting where $k \geq \operatorname{poly}\log(s)$ servers hold C and a client holds x, we obtain a protocol in which the client privately learns $C(x)$ by communicating $\tilde{O}(m)$ bits with each server.
- In a setting where x is arbitrarily distributed between $k \geq \operatorname{poly}\log(s)$ parties who all know C, we obtain a secure protocol for evaluating $C(x)$ using $O(m \cdot \operatorname{poly}(k))$ communication.

Both types of protocols tolerate $t = k/\operatorname{poly}\log(s)$ dishonest parties and their computational complexity is nearly linear in s. In particular, the protocols are optimal "up to polylog factors" with respect to communication, local computation, and minimal number of participating parties. We then apply the above results to obtain sublinear-communication secure protocols for natural database search problems. For instance, for the partial match problem on a database of n points in $\{0,1\}^m$ we get a protocol with $k \approx \frac{1}{2}\log n$ servers, $\tilde{O}(m)$ communication, and nearly linear server computation. Applying previous protocols to this problem would either require $\Omega(nm)$ communication, $\tilde{\Omega}(m)$ servers, or super-polynomial computation.

1 Introduction

As networking becomes a common tool and data can be accessible to all, many applications require distributed access of clients to data servers over the web and other network environments. Once the search is not locally performed, privacy might become a major concern. This motivates the problem of *privacy-preserving database search*, allowing clients to search a database without revealing their search queries to the servers storing the database. Since the databases being searched might be very large, it is desirable to obtain privacy-preserving search protocols whose communication complexity is sublinear in the database size.

The above problem was extensively studied within the context of *private information retrieval* (PIR) [11]. The goal of PIR is to allow a client to privately retrieve the ith item (say, a bit) from a database stored in one or more servers.

* Research supported by Israel Science Foundation grant 36/03.

PIR can be used as a building block for more complex database search operations. Using PIR to probe a data structure representing the database, one can obtain sublinear-communication private protocols for problems such as keyword search [10, 13] or approximate nearest neighbor search (e.g., using [21]).[1] Unfortunately, many natural database search problems that arise in practice are not known to have efficient data structures, namely ones which provide the guarantee that each query can be answered by making few probes into the data structure. For instance, all known algorithms for the *partial match* problem (aka "keyword search with wildcards") require either the number of probes to be nearly linear in the database size or the data structure to be exponential in the length of database entries (see, e.g., the best algorithms known for partial match by Charikar et *al.* [9] and the survey by Miltersen [22]). Hence, the generic PIR-based approach is not useful for this problem. The same holds for many other natural and useful search problems, including Boolean information retrieval (supporting "advanced Google search" functionality), exact nearest neighbor search, and others.

Our point of departure is the observation that most practical database search operations can be efficiently implemented using *constant depth circuits.* (By default, we allow circuits to use AND, OR, NOT, and XOR gates with unbounded fan-in and fan-out.) That is, it is possible for the server to represent its database as a (large) constant-depth circuit C and for the client to independently represent its query as a (small) input x, such that $C(x)$ returns the answer to the client's query on the server's database. Given this observation, it suffices to obtain protocols for securely evaluating a constant-depth circuit C held by the server on an input x held by a client, such that the communication complexity of the protocol is dominated by the size of x rather than by the size of C. Unfortunately, no protocols of this type are known. Using the current toolbox of techniques for secure two-party computation, one can either obtain protocols whose communication complexity is (at least) linear in the circuit size [30, 17] or ones whose computational complexity is exponential in the input size [24].

A good solution to the above problem would imply a major breakthrough in the theory of secure computation. (A small step in this direction, resolving the case in which C is a 2-DNF formula, was very recently made in [6].) In the current work we consider a relaxed setting where (few) different servers hold copies of C, and the client's privacy should be protected against every individual server or collusion of servers of some bounded size t. This "data replication" scenario is the one originally considered in the context of PIR [11]. It is arguably becoming more and more relevant to practice, with the widespread use of peer-to-peer networks, distributed file backup and web caching systems, and other forms of replicated data. Moreover, in this setting it is possible to avoid the use of expensive "cryptographic" computations (e.g., modular exponentiations) which would make the protocols computationally infeasible in practice. Our goal in this setting is to simultaneously obtain nearly-optimal communication (of the order

[1] The PIR-based approach was generalized in [24] to turn *arbitrary* sublinear-communication protocols into private ones. However, the resulting protocols generally require a super-polynomial amount of computation.

of $|x|$) and local computation (of the order of $|C|$), while minimizing the number of servers.

In addition to the client-servers scenario discussed above, we also consider the complexity of evaluating constant-depth circuits in a multiparty setting, without any data replication assumptions. Here the circuit C specifies the functionality to be computed (hence it is known to *all* parties) and the input x is arbitrarily partitioned between the parties. In this case, the best known techniques from the multiparty computation literature would either require linear communication in $|C|$ [5] or an exponential amount of computation and $\tilde{\Omega}(|x|)$ parties [3]. Our goal, as before, is to simultaneously obtain nearly optimal communication and computation while minimizing the required number of parties (alternatively, maximizing the security threshold).

1.1 Our Results

We obtain communication-efficient protocols for securely evaluating constant-depth circuits in both the client-servers setting and the multiparty setting discussed above. Our main protocols are optimal "up to polylog factors" with respect to all three parameters of interest: communication, computation, and number of participating parties. Furthermore, the protocols typically require a minimal amount of interaction, consisting of only two communication rounds (a single round of queries and answers in the client-servers case). Since the number of servers or participating parties is the most crucial resource, we also attempt to optimize the multiplicative constants involved, as was done in the context of PIR. In the case of depth 2 circuits (which in particular suffices for capturing general DNF or CNF evaluation and secure partial match), the number of servers can be as low as $\frac{1}{2}\log_2|C|$ while maintaining (essentially) optimal communication and computation.

We now provide a more detailed account of our results. We let C denote a circuit of size s and depth c with an m-bit input x. In the client-servers setting, each of k servers holds C and a client who holds x should privately learn $C(x)$. In the multiparty setting the circuit C is known to all k parties, where x is distributed between them, and (by default) they all privately learn $C(x)$. Let t denote a security threshold ($t = 1$ by default). We obtain the following two main types of protocols:

- In the client-servers setting we obtain protocols where the communication complexity and the client's computation complexity are $\tilde{O}(m)$ per server, the computation complexity of each server is $\tilde{O}(s)$, and $k = O(t \cdot \log^{c-1}(s))$.
- In the multiparty setting we obtain a protocol for $k = O(\log^{c-1}(s))$ parties in which the communication complexity is $O(m \cdot \mathrm{poly}(k))$, the computation complexity of each party is $\tilde{O}(s)$, and the protocol resists $t = \Omega(k/\log^{c-1}(s))$ dishonest parties.

All these protocols are secure in an information-theoretic sense.

We then apply the above general results and optimize them to obtain efficient secure protocols for database search problems. For instance, for the partial match problem on a database of n points in $\{0,1\}^m$, we get a protocol with $k \approx \frac{1}{2}\log n$ servers, $\tilde{O}(m)$ communication, and nearly linear server computation.

1.2 Overview of Techniques

The main technical tool we use in obtaining the above results is a compact representation of constant-depth circuits by probabilistic low-degree multivariate polynomials. We rely on techniques that have been part of a large body of work proving lower bounds on the size of constant-depth circuits (originating from [25, 27]), tailoring them to our different goals. Additional tools we employ are *ϵ-biased generators* [23] and *randomizing polynomials* [19].

Randomizing polynomial provide a secure reduction of a "complex" functionality $f(x)$ to a *low-degree* randomized functionality $p(x,r)$. (Here r represents "private" randomness chosen by the functionality; both x and r count towards the degree.) Such reductions are motivated by the fact that most standard protocols for secure function evaluation can handle low-degree functionalities very efficiently. Specifically, the main motivation for introducing randomizing polynomials in [19] was the fact that evaluating low-degree polynomials requires few *rounds* of interaction. The current motivation is different: We are mainly interested in minimizing the *communication* complexity. We exploit the fact that the amount of communication required for evaluating a vector of low-degree polynomials is dominated by the *length* of the vector (i.e., the number of outputs), rather than by the description size of the polynomials. Thus, our goal is to construct *short* vectors of low-degree randomizing polynomials representing constant-depth circuits.

In all previous constructions of randomizing polynomials from the literature, the *output* length of p is at least linear in the representation size of f, even when f outputs only a single bit. For instance, in [19] it is shown how to construct a vector of degree-3 randomizing polynomials whose length is quadratic in the size of a branching program computing f. In our case, both the output length and the amount of private randomness must be sublinear in the circuit size. To this end we define the more general notion of a *randomizing polynomials collection* (RPC), which introduces *public randomness* in addition to the private randomness r. Specifically, an RPC is defined by a collection of polynomial vectors $p_\rho(x,r)$, where the "key" ρ is viewed as public randomness and thus does not count towards the degree. We say that the RPC $p_\rho(x,r)$ represents the function f if: (1) it is possible to recover $f(x)$ from $p_\rho(x,r)$ with a negligible failure probability (over the choices of ρ and r), and (2) the output of $p_\rho(x,r)$ gives (essentially) no additional information about x, *even given the knowledge of the public randomness ρ*. The usual notion of randomizing polynomials corresponds to the special case in which ρ is empty.

An RPC representation for f naturally gives rise to secure protocols for f, similarly to the case of standard randomizing polynomials. Thus, our goals reduce to constructing "good" RPCs for constant-depth circuits. The *degree* of the RPC corresponds to the minimal number of participating parties (alternatively, maximal security threshold), and thus serves as our main optimization goal. In addition to minimizing the degree, we wish to optimize both the output length and the amount of randomness, and in particular require them to be sublinear in the circuit size.

Our main RPC construction proceeds in three stages. The first and main stage applies a variant of the techniques of Razborov and Smolensky [25, 27] to create a *short* vector of low-degree randomized polynomials that uses a large amount of public randomness (but no private randomness) in order to reduce the degree of f. This representation guarantees that $f(x)$ can be reconstructed from the outputs of the polynomials with overwhelming probability, yet these outputs might reveal additional information about x. In the second stage we reduce the amount of public randomness by using ϵ-biased generators [23, 1]. Finally, we eliminate the extra information about x revealed by the polynomials using previous constructions of degree-3 randomizing polynomials [19]. This stage introduces a small amount of private randomness and only incurs a minor increase to the degree.

Organization. The remainder of the paper is organized as follows. In Section 2 we give some definitions, and in particular define the notion of RPCs. In Section 3 we describe our main RPC construction for constant-depth circuits and in Section 4 we apply it to obtain secure protocols in both the client-servers setting and the multiparty setting. Finally, Section 5 discusses applications to concrete database search problems.

2 Preliminaries

2.1 Circuits

We represent functions using Boolean circuits with unbounded fan-in and fan-out, as defined below. A circuit C is a labelled directed acyclic graph. The nodes with no incoming edges are labelled with variables (x_i), their negations $(\bar{x}_i)$, or constants (0 or 1). All other nodes are called *gates* and are labelled with some operator. Our default basis of operators includes AND, OR, NOT, and XOR. The nodes from which there are edges to a gate g are called the inputs of g. We refer to the number of such inputs as the *fan-in* of g. In the full version we also consider a generalization of XOR gates to MOD_{p^e} gates, where p is prime and $e \geq 1$ is an integer. Such a gate outputs 1 *iff* the sum of its inputs is 0 mod p^e (for $p^e = 2$ this is a $\mathsf{NOT} - \mathsf{XOR}$ gate).

The *size* of a circuit is the number of edges. Its *depth* is the length of the longest path from a variable node to an output node, where intermediate NOT and XOR gates do not count towards the depth. Nodes with no outgoing edges are called the *output* nodes. We denote by $C(x)$ the output of C on input x, and say that C computes a function f if $C(x) = f(x)$ for all inputs x.

We will focus on the case of *constant-depth* circuits. By this we refer to (polynomial-time uniform) families of circuits whose depth is bounded by some constant c, independently of the input length. Such circuits over the basis AND, OR, NOT (resp., $\mathsf{AND}, \mathsf{OR}, \mathsf{NOT}, \mathsf{MOD}_{p^e}$) correspond to the complexity class

AC^0 (resp., $AC^0(p^e)$). Note that, using De-Morgan's law, one can eliminate AND gates without increasing the depth.

The case of depth-2 circuits will be of particular interest. An *n-term DNF* formula is a depth-2 circuit computing the disjunction (OR) of n conjunctions (AND) of literals. For instance, $(x_1 \wedge \bar{x}_2) \vee (x_2 \wedge \bar{x}_3 \wedge x_4)$ is a 2-term DNF formula.

2.2 Secure Computation

We consider two different scenarios for secure computation: a client-servers scenario, which may be viewed as a distributed form of two-party computation, and the standard multi-party scenario. We begin by recalling the latter.

Multi-Party Setting. In the multi-party setting there are k parties, each holding an input x_i to a functionality f. By default, we consider *deterministic, single-output* functionalities; that is, the output $f(x_1, \ldots, x_k)$ should be learned by all parties. Generalization to randomized, multiple-output functionalities is straightforward.

Our protocols in this setting satisfy standard definitions for secure multi-party computation from the literature [7, 8, 18]. In fact, all our protocols are secure in an *information-theoretic* sense, assuming the availability of secure point-to-point channels. More specifically, our protocols will be *statistically secure*, where security is parameterized by a (statistical) security parameter σ which is given to all parties as an additional input.

We will distinguish between security in the *semi-honest* model (capturing "honest-but-curious" players or a passive adversary) and security in the *malicious* model (capturing an active adversary). In the latter case, we assume the availability of broadcast. In both cases, we allow the adversary to adaptively corrupt up to t parties.

Client-Servers Setting. Our client-servers model generalizes the model for information-theoretic PIR introduced in [11]. In this model there is a client (or *user*) $\mathcal{U}$ who holds an input x of length m, and k servers $\mathcal{S}_1, \ldots, \mathcal{S}_k$ who all hold the same input C of length s. The goal is for the client to learn the value $f(C, x)$, for some publicly known function f, while keeping its input x hidden from any collusion of t servers. We will be particularly interested in the case where the servers hold (a description of) a constant-depth circuit C and the client holds an input x to this circuit. In this case, f will be a *universal* function defined by $f(C, x) = C(x)$.

All of our protocols in this setting will only require a single round of interaction in which the client sends a query to each server and receives an answer in return. The protocol is ϵ-correct if the client's output is correct except with error probability bounded by ϵ. By default, ϵ should be exponentially small in the security parameter σ.

Similarly to the case of PIR, our default security requirement only considers the privacy of the client. We say that the protocol is *t-private* if any collusion of t servers can learn nothing about the client's input x.

We will also consider enhanced client-servers protocols that additionally protect the privacy of the servers' input. In such a protocol, the client should learn essentially nothing about C except the output $f(C, x)$. More specifically, the protocol is said to be *δ-server-private* (with respect to f) if the view of the client can be simulated, up to statistical distance of δ, based on its input and output alone. (See full version for a formal definition.) We require $\delta(\sigma) = 2^{-\Omega(\sigma)}$ by default. Following the terminology that was used in the context of PIR [16], we refer to protocols that satisfy this additional server privacy requirement as being *symmetrically private*. To enable server privacy without direct interaction between the servers, it is required to allow the servers to share a common random string (CRS) [16].

The above security requirements induce three levels of security for client-servers protocols: (1) basic security, providing client-privacy only; (2) symmetric privacy with respect to a semi-honest client; and (3) symmetric privacy with respect to a malicious client.

2.3 Randomizing Polynomials

We generalize the notion of randomizing polynomials from [19] and consider what we call *collections* of randomizing polynomials. Before describing our generalization, we review the original notion of randomizing polynomials.

Randomizing polynomials represent a function f using a vector of multivariate polynomials over a finite field. (In this work, the underlying field will be GF(2) by default.) Each polynomial has two types of inputs: ordinary inputs x and random inputs r. A randomizing polynomials vector will usually be denoted by $p(x, r)$. Note that p is a *vector* of polynomials which all act on the same variables x, r. The vector $p(x, r)$ is said to represent a function f if its output distribution is "equivalent" to the output of f in the following sense. First, given $p(x, r)$ it is possible to recover $f(x)$ (without knowing r). In the other direction, given $f(x)$ alone it is possible to sample from the output distribution $p(x, r)$ induced by a uniform choice of r (without knowing x).

For the purpose of allowing more compact representations, we generalize the notion of randomizing polynomials by considering *collections* of randomizing polynomials. Let $p_\rho(x, r)$ denote a collection of polynomial vectors, indexed by a key ρ. When ρ is picked at random, we refer to it as *public randomness*, whereas r is referred to as *private randomness*. We will say that $p_\rho(x, r)$ represents a function $f(x)$ if the following two properties hold: (1) it is possible to recover $f(x)$ from the output of $p_\rho(x, r)$ (except for a negligible failure probability over the choices of ρ and r), and (2) the output of $p_\rho(x, r)$ gives (essentially) no additional information about x *even given the knowledge of the public randomness ρ*. These properties guarantee that the secure computation of f can be reduced to that of p_ρ, where ρ is a *public* random string chosen independently of the inputs.

Definition 1. (Randomizing Polynomials Collection (RPC)) *Let $p_\rho(x,r) = ({p_1}_\rho(x,r),\ {p_2}_\rho(x,r), \ldots, {p_l}_\rho(x,r))$ be a vector of l polynomials over the input $x = (x_1, \ldots, x_m)$, the private random input r, and the public random input ρ. All polynomials are over a finite field F, where $F = \mathrm{GF}(2)$ by default. We say that $p_\rho(x,r)$ is an ϵ-correct, δ-private* randomizing polynomials collection (RPC) *for $f(x)$ if the following holds.*

- *(*ϵ-correctness*) There exists a reconstruction algorithm $\mathcal{R}$ such that for every input x, $Pr_{r,\rho}[\mathcal{R}(p_\rho(x,r)) \neq f(x)] \leq \epsilon$, where r and ρ are chosen uniformly and independently. Note that reconstruction should not depend on ρ. (Intuitively, correctness should hold for all but a negligible fraction of the ρ's.)*
- *(*δ-privacy*) There exists a simulator $\mathcal{M}$ such that for every input x,*

$$SD[(\rho, \mathcal{M}(f(x))), (\rho, p_\rho(x,r))] \leq \delta,$$

where r and ρ are chosen uniformly and independently at random and SD denotes statistical distance. Note that the simulator is not given ρ yet the simulation should also be successful when considered jointly with ρ. This implies that the output distribution of $p_\rho(x,r)$ should be essentially the same given almost any fixed ρ.

The length *of $p_\rho(x,r)$ is l. Its* degree *is the maximal degree of a polynomial in the vector, taking into account only the* input variables x and the private random variables r. *We refer to $|\rho|$ as the* public randomness complexity *and to $|r|$ as the* private randomness complexity.

Universal RPC. We will sometimes want to represent each function f in a class $\mathcal{F}$ by an RPC, such that all RPCs in the class share the same simulator and reconstruction algorithms. In such a case, we say that the class of RPCs is *universal* for the function class $\mathcal{F}$. Our main RPC construcsion will be *fully universal*: the same reconstruction algorithm and simulator can be applied for all functions f. (Of course, the RPC itself varies from one function to another.) This feature will be useful for obtaining protocols in the client-servers model, where a circuit held by the servers is evaluated on an input held by the client.

Polynomial Collection (PC). We will also consider RPCs which do not need to satisfy the privacy requirement. (In fact, such collections will serve as an intermediate step in constructing RPCs.) In this case, there is no need for private randomness. We will refer to this relaxed type of RPC as a *polynomial collection* (PC) and denote it by $p_\rho(x)$. A PC of length 1 will also be referred to as a *randomized polynomial*. Note that the identity function $p(x) = x$ defines a trivial PC with no public randomness. However, we will be interested in constructing *universal* PCs (whose reconstruction algorithm does not depend on f), and in particular ones in which the output length is sublinear in the input length.

Randomizing polynomials for branching programs. We will rely on an efficient representation of branching programs by randomizing polynomials.

Lemma 1. *[20] Suppose $f(x)$ can be computed by a branching program of size ℓ. Then, f can be represented by a vector $p(x, r)$ of degree-3 (perfectly correct and private) randomizing polynomials of length $O(\ell^2)$ and randomness complexity $O(\ell^2)$. Moreover, the degree of p in the x variables is 1.*

2.4 ϵ-Biased Generators

The communication complexity of some of our protocols will depend on the *randomness complexity* of the underlying RPCs. This calls for the use of pseudo-randomness. It turns out that the pseudo-random generators we need are only required to fool linear distinguishers. Thus, we will rely on the following standard notion of ϵ-biased generators [23].

Definition 2. (ϵ-biased generator) *A function $G : \{0,1\}^\ell \to \{0,1\}^{n(\ell)}$ is an* ϵ-biased generator *(for some* bias *function $\epsilon(\ell)$) if for all sufficiently large ℓ and all linear functions $L : \mathrm{GF}(2)^{n(\ell)} \to \mathrm{GF}(2)$, we have*

$$|\Pr[L(G(U_\ell)) = 1] - \Pr[L(U_{n(\ell)}) = 1]| \le \epsilon(\ell)$$

By default, the function $\epsilon(\ell)$ is required to be negligible.

3 Low-Degree RPCs for Constant Depth Circuits

In this section we present our main constructions of low-degree PCs and RPCs for constant-depth circuits. The high level idea is to simulate the given circuit in a gate-by-gate fashion, going from the inputs to the output, where each such simulation step does not add much to the degree and does not create a big error.

For simplicity, we assume that the circuit has a single output and thus computes a boolean function; a generalization to the non-boolean case is straightforward. We also restrict the attention to $AC^0(2)$ circuits (a generalization to $AC^0(p^e)$ circuits appears in the full version). Finally, we may assume without loss of generality that the circuit contains only $\mathsf{OR}, \mathsf{XOR}, \mathsf{NOT}$ gates and that the output gate is OR.

A central "gadget" in the construction is the following representation of the OR function by a single randomized polynomial, namely a PC of length 1.

OR *construction with parameter γ.* Given an OR gate with t inputs, let R be a random $\gamma \times t$ matrix over $\mathrm{GF}(2)$, and define the randomized polynomial:

$$p_R(x_1, \ldots, x_t) = 1 - \prod_{i=1}^{\gamma}(1 - (\sum_{j=1}^{t} R_{i,j} x_j)). \qquad (1)$$

If $\mathsf{OR}(x) = 0$ then so is $p_R(x)$, while if $\mathsf{OR}(x) = 1$ then the probability of every inner product (the sum) to result in 0 or 1 is equal. Thus, $p_R(x) = \mathsf{OR}(x)$ except with probability $2^{-\gamma}$ over the choice of R, and we have the following.

Lemma 2. *An* OR *gate with t inputs has a $2^{-\gamma}$-correct PC representation over* GF(2) *of length* 1 *and degree* γ.

Notice that in this case we only have a one-sided error; however, applying the OR gadget within the general construction will generally result in a two-sided error. We now proceed to the case of a general circuit C.

Basic construction with parameter σ. Given a circuit C of size s, we define a randomized polynomial $p^g_\rho(x)$ for every gate g of C, so that the PC representing the circuit is the polynomial defined for the output gate. The polynomial p^g is defined inductively as follows. An input is represented by a deterministic polynomial corresponding to its straightforward arithmetization (e.g., $\bar{x}_i$ is represented by $1 - x_i$). If g is an OR gate, then p^g is defined by applying the above OR construction with $\gamma = \log s + \sigma$ to the polynomials representing its inputs. This step introduces new public randomness. Finally, if g is a XOR or a NOT gate, then p^g is naturally defined in terms of the polynomials representing the input gates (e.g., their summation in case of XOR). The degree of the output polynomial is bounded by the maximal degree of a polynomial representing a gate (as a function of its inputs) to the power of the depth of the circuit. Using union bound on the error probability of the representation[2] we have the following:

Lemma 3. *Given a circuit C with m inputs, one output, size s and depth c, the* basic construction *with parameter σ produces a $2^{-\sigma}$-correct PC representation for C over* GF(2) *of length* 1, *degree at most* $(\log s + \sigma)^c$, *and public randomness complexity* $O(s(\log s + \sigma))$.

The parameters of the above PC representation leave much to be desired. First, the degree depends on σ which will generally be larger than $\log s$; moreover, even for depth-2 circuits (capturing the important case of DNF) the degree grows *quadratically* with $\log s + \sigma$. As we shall see, one can make the degree linear in $\log s$ (and independent of σ) in the depth-2 case. Finally, the public randomness complexity is very large.

We will start by reducing the amount of randomness via the use of ϵ-biased generators. We use here the *powering construction* by Alon et *al.*:

Lemma 4. *[1] There exists an efficient ϵ-biased generator $G : \{0,1\}^{2\ell} \to \{0,1\}^t$ with $(t-1)2^{-\ell}$-bias.*

We use the above generator to produce the matrix R from the OR construction (1). To ensure independence between rows, we use a separate seed for every row. This gives the following:

Lemma 5. *Let $0 < \epsilon < \frac{1}{2}$. An* OR *gate of t inputs has a PC representation over* GF(2) *of length* 1, *degree* γ, $(\frac{1}{2} + \epsilon)^\gamma$*-correctness, and public randomness complexity* $2\gamma\lceil \log \frac{t}{\epsilon} \rceil$.

[2] The random matrices of the different gates are not considered independent in this analysis and thus the same matrix can be "recycled".

We turn to the question of optimizing the degree, our most crucial parameter. The main observation is that one can reduce the error probability in the basic construction by repeating it σ times *in parallel*, using independent randomness in each copy. This will result in a universal PC of length σ from which the output can be recovered, except with $2^{-\Omega(\sigma)}$ error probability, by applying some fixed *threshold* function. We will then enhance this PC into a universal RPC at a minor additional cost.

Improved construction with parameter σ. The improved construction is similar to the basic construction with the following changes:

– OR gates are represented using the construction of Lemma 5.
– The output gate is assigned a special parameter γ_o when Lemma 5 is applied.
– The representation is repeated σ times in parallel, producing a PC of length σ with threshold as its reconstruction function.
– If privacy is required, a randomizing polynomials representation of a threshold function is applied to the σ outputs of the PC, producing an RPC.

Theorem 1. *Let σ be a security parameter. Given a circuit C with m inputs, one output, size s and depth c, the* improved construction *yields representations of C by:*

– *a PC over* GF(2) *of length σ, degree $d_{PC} = \lceil(\log s + 3)\rceil^{c-1}$, $2^{-\Omega(\sigma)}$-correctness, and public randomness complexity $O(\sigma \log^2 s)$;*
– *an RPC over* GF(2) *of length $O(\sigma^4)$, degree $d_{PC} + 2$, $2^{-\Omega(\sigma)}$-correctness, $2^{-\Omega(\sigma)}$-privacy, public randomness complexity $O(\sigma \log^2 s)$, and private randomness complexity $O(\sigma^4)$.*

The above representations are universal*, i.e., their reconstruction algorithm and simulator do not depend on the circuit C.*

Proof. We instantiate the improved construction outlined above with the following parameters. Consider first the case where no ϵ-biased generators are used to reduce public randomness. In this case, we choose $\gamma = \log s + 2$ and $\gamma_o = 1$. By Lemma 2 each randomized polynomial representing an internal OR gate of C errs with at most $2^{-\gamma}$ probability. Using a union bound, the probability that at least one of them errs is bounded by $s \cdot 2^{-\gamma} \leq \frac{1}{4}$. Since $\gamma_o = 1$, the output will be 1 with probability at most $\frac{1}{4} \cdot \frac{1}{2} = \frac{1}{8}$ if $f(x) = 0$ and at least $\frac{3}{4} \cdot \frac{1}{2} = \frac{3}{8}$ if $f(x) = 1$. (Here we assume that an independent random matrix R is used for the top gate; this has no impact on the asymptotic complexity.)

Consider the PC obtained by concatenating σ copies of the above construction, using independent randomness in each copy. The degree remains as before. By Chernoff's bound, we have a (universal) $2^{-\Omega(\sigma)}$-correct reconstruction algorithm that outputs 1 if at least $\sigma/4$ out of the σ outputs evaluate to 1.

To optimize the amount of public randomness, we choose the bias parameter to be $\epsilon = \frac{1}{s}$. The improved construction is then applied with parameters $\gamma = \left\lceil (\log s + 2)/\log \frac{1}{\frac{1}{2}+\epsilon} \right\rceil$ and $\gamma_o = 1$. The resulting PC has degree $\gamma_o \cdot (\gamma)^{(c-1)}$ which, for sufficiently large s, is bounded by $\lceil(\log s+3)\rceil^{c-1}$. The amount of randomness for each randomized polynomial in the PC is the amount of randomness needed for a single OR gate which, by Lemma 5, is $O(\gamma \log \frac{s}{\epsilon}) = O(\log^2 s)$.

The output of the above PC representation $p_\rho(x)$ might reveal additional information about x, other than what follow from $C(x)$. To this end, we apply a randomizing polynomials representation of a $(\sigma/4)$-threshold function to the σ outputs of p_ρ. Since any threshold function on σ bits can be computed by a branching program of size $O(\sigma^2)$, Lemma 1 guarantees a representation by degree-3, perfectly correct and private randomizing polynomials $P(\tilde{x}, r)$ where both the length and the private randomness complexity are $O(\sigma^4)$. Moreover, the degree in the $\tilde{x}$ variables is 1. Applying this construction with $p_\rho(x)$ as an input produces an RPC $\tilde{P}_\rho(x,r) = P(p_\rho(x), r)$ with the required parameters.

Note that the perfect simulator (resp., reconstruction algorithm) of P can serve as a universal $2^{-\Omega(\sigma)}$-private simulator (resp., $2^{-\Omega(\sigma)}$-correct reconstruction algorithm) for the above RPC. This follows from the fact that the simulation and reconstruction of P are perfect when conditioned on the event that the choice of ρ does not lead $p_\rho(x)$ to err. $\square$

We note that above PC and RPC constructions implicitly define efficient evaluation algorithms whose complexity is nearly linear in the circuit size s (defined as the number of *wires*). Moreover, for the purpose of bounding the degree, one can take s to be the number of OR gates in the circuit. Thus, for the important special case where C is an n-term DNF formula, we get a PC (resp., RPC) of degree $\log n + O(1)$ and length σ (resp., $O(\sigma^4)$).

4 Secure Computation of Constant Depth Circuits

In this section we describe the application of low-degree representations to communication-efficient secure computation. Combined with the results of the previous section, we will get efficient protocols for constant-depth circuits.

We will separately consider the client-servers model and the multiparty model, both defined in Section 2.2. In the basic client-servers setting, where only the privacy of the client is guaranteed, it will suffice to use an underlying PC representation of the function we wish to compute. In multi-party setting, as well as the client-servers setting with server privacy, we will need to rely on the stronger RPC representation.

The protocols we describe rely on standard techniques for securely evaluating low-degree polynomials, previously used in the contexts of information-theoretic secure multi-party computation [5, 2, 12, 19] and private information retrieval [11, 16, 4]. We only sketch the high-level structure of the protocols and the parameters they achieve. Further details can be found in the full version.

Throughout this section, assume F to be an extension field of GF(2) having more than k elements, where k is the number of servers or parties. Most of the protocols will employ Shamir's secret-sharing [26] over F in order to securely compute polynomials over its subfield GF(2). We let t denote a security threshold, where $t = 1$ by default.

4.1 The Client-Servers Setting

In the general definition of client-servers computation given in Section 2.2, the client holds an input x, the servers hold an input C, and the client wishes to learn $f(C,x)$ for some publicly known function f. It will be convenient for our purpose to focus on the case where C represents a circuit, x is an m-bit input to this circuit, and f is the *universal* function defined by $f(C,x) = C(x)$. We assume that C is taken from some known class $\mathcal{C}$, typically the class of depth-c, size-s circuits. Thus, the problem we consider is that of allowing the client to privately learn the value of a circuit $C \in \mathcal{C}$ held by the servers on its secret input x.

We start with the case of client-privacy only. In this case, a universal degree-d PC representation for $\mathcal{C}$ gives rise to the following simple protocol. Suppose $k > dt$. Let $p_\rho(x)$ denote be a PC representing the servers' circuit C. The client secret-shares each of its input bits between the servers, using the t-private Shamir's scheme over F. In parallel, it picks ρ at random and sends it to all k servers. Each server replies to the client with the output of p_ρ on the m-tuple of shares it received.[3] The client recovers $C(x)$ by first recovering the output y of p_ρ (since $k > dt$, this can be done by polynomial interpolation) and then applying the (universal) reconstruction algorithm of $\mathcal{C}$. The error probability of the protocol is the same as that of p_ρ. Combining the above protocol with the representation obtained by Theorem 1 we have the following.

Theorem 2. *Let f be the* universal circuit evaluation function $f(C,x) = C(x)$. *Suppose k servers hold a circuit C of size s, depth c, m inputs and a single output, and the client holds an assignment $x \in \{0,1\}^m$. Then for every integer $t \geq 1$ there exists a t-private client-servers protocol with the following parameters:*

- *The number of servers is $k = t \cdot (\log s + O(1))^{c-1}$.*
- *The communication consists of $O(m \log k + \sigma \log^2 s)$ bits sent from the client to each server and $O(\sigma)$ bits sent in return.*
- *The computation of each party is nearly linear in its input length (up to factors of σ and $\log k$).*

In the special case of n-term DNF we can substitute in the above $c = 2$ and $s = n$ (see remark following Theorem 1). Thus, we get a protocol with $\log n + O(1)$ servers, $\tilde{O}(m)$ communication, and $\tilde{O}(mn)$ server computation.

Using a technique of Woodruff and Yekhanin [29], it is possible to reduce the number of servers by a factor of 2 without substantially increasing the *total* communication. Specifically, in the resulting protocol the total communication is $O(m\sigma \log m \log^{c-1} s)$ bits per server, and the computation of each server is $\tilde{O}(sm)$. The following is implicit in [29]:

Lemma 6. *Let $t \geq 1$ be an integer, and let $p(x)$ be an m-variate polynomial of degree d, over a field of a* prime *order $q > dt + 1$. Then, there exists a t-private*

[3] In fact, it suffices for each server to send back just a single bit by projecting the answers from F back to GF(2) (cf. [4], Lemma 3).

client-servers protocol to compute $p(x)$ (where p is held by the servers and x by the client), with $k = \lceil \frac{dt+1}{2} \rceil$ servers, queries of m field elements per server, and answers of $m+1$ field elements per server.

The protocol from [29] is similar to the basic client-servers protocol described above, and in particular the client's queries have the same structure. The improvement comes from the fact that each server replies not only with the value of p on the point queried by the client, but also with the values of its m partial derivatives. In order to apply Lemma 6 in our context, we should emulate computation of $p_\rho(x)$ over GF(2) using computation over a field of a large prime order q. Since the value of $p_\rho(x)$ when computed *over the integers* is bounded by $2^{O(\log^{c-1} s \log m)}$, it suffices to use a prime q of the latter size. This yields the communication complexity specified above. In particular, in the case of n-term DNF we get a protocol with $\frac{1}{2}\log n + O(1)$ servers and nearly optimal communication and computation.

Symmetrically private client-servers protocols. We now briefly discuss an extension of the above results to the case where server privacy is also a requirement. (Further details can be found in the full version.) In this case, the stronger RPC representation should replace the previous RP representation. If the circuit C held by the servers is represented by a (universal) RPC $p_\rho(x, r)$, the protocol will let the client securely evaluate the polynomial vector $p_\rho(x, r)$, where both r and ρ are taken from the CRS shared by the servers. (Note that here it is crucial that r remain hidden from the client.) Unlike the previous case of client privacy only, here the polynomial evaluation protocol should prevent the client from learning anything other than the output of p_ρ. The simple polynomial evaluation protocol described before fails to achieve this property. In the case of a semi-honest client, it is easy to eliminate the extra information from the servers' answers by masking them with (private) randomness from the CRS. The case of a malicious client is more difficult, since the queries it sends to the servers might not be well formed. For instance, the client might share values from $F \setminus \{0, 1\}$ or send shares that are not of the right degree. In the full version we describe a client-servers protocol for fully secure polynomial evaluation that resists a malicious client without requiring additional rounds of interaction and with a very minor complexity overhead. The protocol relies on the conditional disclosure methodology of Gertner et al. [16] and can also be used to obtain better t-private SPIR protocols than those implied by [16]. Applying this machinery, we get an enhanced version of Theorem 2 which provides server privacy against a malicious client, with the main additional cost of increasing the answer size from σ to $\sigma^{O(1)}$ (reflecting the larger length of the RPC).

4.2 The Multi-Party Setting

Here we consider the standard MPC setting in which an m-bit string x is partitioned between k parties, who all want to learn the output of a publicly known function $f(x)$. Given an RPC $p_\rho(x, r)$ representing f, we view p_ρ as a randomized low-degree ideal functionality in which r is a "private" random input and ρ is a "public" random input. Both r and ρ are independently chosen by the functionality; the difference between them is that ρ is additionally revealed to the adversary. (Recall that the separation between ρ and r is motivated by the goal of minimizing the *degree* of the RPC, which in the current context will correspond to the minimal required number of parties.) The reconstruction algorithm of the RPC now defines a non-interactive reduction from f to p_ρ: to securely compute $f(x)$ the parties first securely compute $p_\rho(x, r)$ and then locally apply its reconstruction algorithm.

We turn to the question of efficiently computing the randomized functionality induced by an RPC p_ρ of degree d. We require the communication complexity to be dominated by the length of the inputs and outputs and not by the description size of p. This can be done by using standard MPC techniques (cf. [5, 3, 15]). In the semi-honest case, a simple two-round protocol for $k > dt$ parties is described in [19]. In the malicious case, we can use the following constant-round protocol for $k > (d+2)t$ parties. First, the parties apply a VSS protocol (e.g., from [5]) to create degree-t shares of x along with shares of *secret* random values r, ρ and shares of 0 (the latter are created by locally adding shares contributed by different players). Using the protocol from [14] the above requires only two rounds. One also needs to ensure that the shared values are elements of the subfield GF(2) rather than general elements of F; this can be done (without additional interaction) using a method from [12]. Next, the players reveal ρ which now defines a *public* polynomial vector p_ρ of degree d. Finally, the players recover $p_\rho(x, r)$: this is done by having each player locally evaluate p_ρ on its shares of x, r, mask the resulting shares with the shares of 0 created in the first phase, and communicate the result to all other players. The players can now recover the output of p_ρ via local error-correction. Combining this protocol with Theorem 1, we have:

Theorem 3. *Let $t \geq 1$ be a security threshold, and C be a circuit of size s and depth c whose m input bits are partitioned between $k \geq \Omega(t \cdot \log^{c-1} s)$ players. Then, there exists a constant-round, statistically t-secure protocol computing $C(x)$ with communication complexity $O(m \cdot poly(k))$.*

5 Constant-Depth Circuits for Search Applications

In this section we demonstrate the usefulness of our general results in the context of database search problems. The goal is to translate the servers' database into a constant-depth circuit and the client's query into a corresponding assignment in a way that would optimize the complexity of our general constructions. For the following problems assume the database consists of n points in $\{0, 1\}^m$:

- *The partial match decision problem* supports queries in $\{0, 1, *\}^m$, where the symbol $*$ is interpreted as "don't care". An answer should be 1 *iff* there is a point in the database that agrees with the query in all indices that are

not $*$. We encode the database as an n-term DNF over the $2m$ variables $x_{1,0}, x_{1,1}, \ldots x_{m,0}, x_{m,1}$. The DNF is constructed by assigning a term for every point in the database (e.g., 1011 translates into $x_{1,1} \wedge x_{2,0} \wedge x_{3,1} \wedge x_{4,1}$). The query is then translated by applying a two-bit encoding for each symbol (0 by 10, 1 by 01 and $*$ by 11). As noted in Section 3, for n-term DNF we get a PC (resp., RPC) of degree $\log n + O(1)$, length σ (resp., $\sigma^{O(1)}$) and $2m$ variables. Using Theorem 2, we get a client-servers protocol with $\log n + O(1)$ servers, $\tilde{O}(m)$ communication and $\tilde{O}(nm)$ server computation. One can reduce the number of servers to $\frac{1}{2}\log n$ as discussed in Section 4.1.
- *The partial match search problem* is similar to the decision version except that a matching point of the database should be retrieved. A naive construction gives a depth-4 circuit, paying a lot on tie breaking between matching points. Using the Valiant-Vazirani lemma [28], the database can be translated to a (probabilistic) circuit with $O(m^2)$ inputs, $\tilde{O}(m)$ outputs, and size $\tilde{O}(nm^2)$.
- *The nearest neighbor search problem* is a problem where the point with the smallest Hamming distance to the query should be retrieved. The database can be translated into an $AC^0(p)$ circuit for $p > m$ with m inputs, $O(m^2)$ outputs, size $O(nm^2)$ and depth 3. The use of MOD_p enables efficient constant-depth computation of Hamming distance.

In the full version we describe optimized constructions of constant-depth circuits for the problems mentioned above as well as for other problems.

Acknowledgement. We would like to thank Ronny Roth for helpful comments.

References

1. [1] N. Alon, O. Goldreich, J. Hastad, and R. Peralta. Simple construction of almost k-wise independent random variables. *Random Structures and Algorithms*, 3(1):289–304, 1992. Preliminary version in FOCS '90.
2. [2] D. Beaver and J. Feigenbaum. Hiding instances in multioracle queries. In *Proc. 7th STACS*, pages 37–48, 1990.
3. [3] D. Beaver, J. Feigenbaum, J. Kilian, and P. Rogaway. Security with low communication overhead. In *Proc. 10th CRYPTO*, pages 62–76, 1990.
4. [4] A. Beimel and Y. Ishai. Information-theoretic private information retrieval: A unified construction. In *Proc. 28th ICALP*, pages 912–926, 2001.
5. [5] M. Ben-Or, S. Goldwasser, and A. Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation. In *Proc. 20th STOC*, 1988.
6. [6] D. Boneh, E.J. Goh, and K. Nissim. Evaluating 2-DNF formulas on ciphertexts. In *Proc. 2nd TCC*, pages 325–341, 2005.
7. [7] R. Canetti. Security and composition of multiparty cryptographic protocols. *J. Cryptology*, 13(1):143–202, 2000.
8. [8] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *Proc. 42st FOCS*, pages 136–145, 2001.
9. [9] M. Charikar, P. Indyk, and R. Panigrahy. New algorithms for subset query, partial match, orthogonal range searching and related problems. In *Proc. 29th ICALP*, pages 451–462, 2002.

[10] B. Chor, N. Gilboa, and M. Naor. Private information retrieval by keywords. Technical report, Department of Computer Science, Technion, 1997.

[11] B. Chor, O. Goldreich, E. Kushilevitz, and M. Sudan. Private information retrieval. In *Proc. 36th FOCS*, pages 41–50, 1995.

[12] R. Cramer, I. Damgård, and U. Maurer. General secure multy-party computation from any linear secret-sharing scheme. In *Proc. of EUROCRYPT*, 2000.

[13] M.J. Freedman, Y. Ishai, B. Pinkas, and O. Reingold. Keyword search and oblivious pseudorandom functions. In *Proc. 2nd TCC*, pages 303–324, 2005.

[14] R. Gennaro, Y. Ishai, E. Kushilevitz and T. Rabin. The Round Complexity of Verifiable Secret Sharing and Secure Multicast. In *Proc. 33rd STOC*, 2001.

[15] R. Gennaro, M.O. Rabin, and T. Rabin. Simplified VSS and fact-track multiparty computations with applications to threshold. In *Proc. 17th PODC*, 1998.

[16] Y. Gertner, Y. Ishai, E. Kushilevitz, and T. Malkin. Protecting data privacy in private information retrieval schemes. *J. of Computer and Systems Sciences*, 60, 2000. Preliminary version in STOC '98.

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

[18] Oded Goldreich. *Foundations of Cryptography: Basic Applications.* Cambridge University Press, 2004.

[19] Y. Ishai and E. Kushilevitz. Randomizing polynomials: A new representation with applications to round-efficient secure computation. In *Proc. 41st FOCS*, pages 294–304, 2000.

[20] Y. Ishai and E. Kushilevitz. Perfect constant-round secure computation via perfect randomizing polynomials. In *Proc. 29th ICALP*, pages 244–256, 2002.

[21] E. Kushilevitz, R. Ostrovsky, and Y. Rabani. Efficient search for approximate nearest neighbor in high dimensional spaces. In *Proc. 30th STOC*, 1998.

[22] P.B. Miltersen. Cell probe complexity–a survey. In *Pre-Conference Workshop on Advances in Data Structures at the 19th Conference on Foundations of Software Technology and Theoretical Computer Science*, 1999.

[23] J. Naor and M. Naor. Small-bias probability spaces: Efficient constructions and applications. *SIAM J. Comput.*, 22(4):838–856, 1993.

[24] M. Naor and K. Nissim. Communication preserving protocols for secure function evaluation. In *Proc. 33rd STOC*, pages 590–599, 2001.

[25] A. Razborov. Lower bounds for the size of circuits of bounded depth with basis (AND, XOR). *Math. Notes of the Academy of Science of the USSR*, 41(4):333–338, 1987.

[26] A. Shamir. How to share a secret. *Communication of the ACM*, 22(11):612–613, 1979.

[27] R. Smolensky. Algebric methods in the theory of lower bound for boolean circuit complexity. In *Proc. 19th STOC*, pages 77–82, 1987.

[28] L.G. Valiant and V.V. Vazirani. NP is as easy as detecting unique solutions. *Theoretical Computer Science*, 47:85–93, 1986. Preliminary version in STOC '85.

[29] D. Woodruff and S. Yekhanin. A geometric approach to information-theoretic private information retrieval. In *Electronic Colloquium on Computational Complexity (ECCC)*, 2005. Report TR05-009. To appear in CCC 2005.

[30] A. C. Yao. How to generate and exchange secrets. In *Proc. 27th FOCS*, 1986.