Parallel Homomorphic Encryption

Seny Kamara
Microsoft Research

Mariana Raykova[1]
IBM Research

Abstract. In the problem of private outsourced computation, a client wishes to delegate the evaluation of a function f on a private input x to an untrusted worker without the latter learning anything about x and $f(x)$. This problem occurs in many applications and, most notably, in the setting of cloud computing.

In this work, we consider the problem of privately outsourcing computation to a *cluster* of machines, which typically happens when the computation needs to be performed over massive datasets, e.g., to analyze large social networks or train machine learning algorithms on large corpora. At such scales, computation is beyond the capabilities of any single machine so it is performed by large-scale clusters of workers.

To address this problem, we consider *parallel* homomorphic encryption (PHE) schemes, which are encryption schemes that support computation over encrypted data through the use of an evaluation algorithm that can be efficiently executed in parallel. More concretely, we focus on the MapReduce model of parallel computation and show how to construct PHE schemes that can support various MapReduce operations on encrypted datasets including element testing and keyword search. More generally, we construct schemes that can support the evaluation of functions in NC^0 with locality 1 and $\mathsf{polylog}(k)$ (where k is the security parameter).

Underlying our PHE schemes are two new constructions of (local) randomized reductions (Beaver and Feigenbaum, *STACS '90*) for univariate and multivariate polynomials. Unlike previous constructions, our reductions are not based on secret sharing and are *fully-hiding* in the sense that the privacy of the input is guaranteed even if the adversary sees *all* the client's queries.

Our randomized reduction for univariate polynomials is information-theoretically secure and is based on permutation polynomials, whereas our reduction for multivariate polynomials is computationally-secure under the multi-dimensional noisy curve reconstruction assumption (Ishai, Kushilevitz, Ostrovsky, Sahai, *FOCS '06*).

1 Introduction

In the problem of private outsourced computation, a client wishes to delegate the evaluation of a function f on a private input x to an untrusted worker without the latter learning anything about x and $f(x)$. This problem occurs in many applications and, most notably, in the setting of cloud computing, where a provider makes its computational resources available to clients "as a service".

One approach to this problem is via the use of homomorphic encryption (HE). An encryption scheme is homomorphic if it supports computation on encrypted data, i.e., in addition to the standard encryption and decryption algorithms it also has an evaluation algorithm that takes as input an encryption of some message x and a function f and returns an encryption of $f(x)$. If a HE scheme supports both addition and multiplication,

[1] Supported by NSF Grant No.1017660. Work done while at Microsoft Research.

then it can evaluate any arithmetic circuit over encrypted data and we say that it is a fully homomorphic encryption (FHE) scheme [10].

The problem of outsourced computation occurs in various forms. For instance, in addition to the simple client/worker setting described above, clients often wish to outsource their computation to *clusters* of workers. This typically occurs when the computation is to be performed over massive datasets, e.g., to analyze large social networks or train machine learning algorithms on large corpora. At such scales, computation is beyond the capabilities of any single machine so it is performed on clusters of machines, i.e., large-scale distributed systems often composed of low-cost unreliable commodity hardware. For our purposes, we will view such a cluster as a system composed of w workers and one controller. Given some input, the controller generates n jobs (where typically $n \gg w$) which it distributes to the workers. Each worker executes its job in parallel and returns some value to the controller who then decides whether to continue the computation or halt.

In this work, we consider the problem of *privately* outsourcing computation to a cluster of machines. To address this, we introduce *parallel* homomorphic encryption (PHE) schemes, which are encryption schemes that support computation over encrypted data through the use of an evaluation algorithm that can be efficiently executed in parallel. Using a PHE scheme, a client can outsource the evaluation of a function f on some private input x to a cluster of w machines as follows. The client encrypts x and sends the ciphertext and f to the controller. Using the ciphertext, the controller generates n jobs that it distributes to the workers and, as above, the workers execute their jobs in parallel. When the entire computation is finished, the client receives a ciphertext which it decrypts to recover $f(x)$.

Applications of PHE. As discussed above, the most immediate application of PHE is to the setting of outsourced computation where a weak computational device wishes to make use of the resources of a more powerful cluster. Clearly, to be useful in this setting it is crucial that either: (1) running the encryption and decryption operations of the PHE scheme take less time than evaluating f on the input x directly; or (2) the PHE scheme is *multi-use* in the sense that the evaluations of several (different) functions can be done on a single ciphertext (this is also referred to as the online/offline setting). In this work we focus on the latter and present several multi-use PHE schemes. Using our schemes a client can encrypt a large database during an offline phase and then, have the workers evaluate many different functions on its data during the online phase. In particular, at the time of encryption, the client does not need to know the functions it will want to evaluate during the online phase.

Parallel computation. Most computations are not completely parallelizable and require some amount of communication between machines. The specifics of how the computation and communication between processors are organized leads to particular architectures, each having unique characteristics in terms of computational and communication complexity. This has motivated the design of several architecture-independent models of parallel computation, including NC circuits [5], the parallel RAM (PRAM) [8,14], Valiant's bulk synchronous parallel (BSP) model [18], LogP [6] and, more recently, the MapReduce [7] and Dryad models [12]. It follows that an important consideration in the design of PHE schemes is the parallel model in which the function will be evaluated. In

this work, we focus on the MapReduce model (which we describe below) but note that our choice is due mainly to practical considerations (e.g., the emergence of cloud-based MapReduce services such as Amazon's Elastic MapReduce) and that PHE can also be considered with respect to other models of parallel computation. As an example, note that any FHE scheme yields an NC-parallel HE scheme for any function f in NC.

1.1 Overview of Techniques

Designing PHE schemes. We propose a general approach to designing PHE schemes. Roughly speaking, our approach yields PHE schemes for any function f that can be randomly reduced to another function g. A randomized reduction (RR) [2,3] from a function f to a function g transforms an input x in the domain of f to a set of n inputs $S = (s_1, \ldots, s_n)$ in the domain of g such that $f(x)$ can be efficiently reconstructed from $(g(s_1), \ldots, g(s_n))$. In addition, a RR guarantees that no information about x or $f(x)$ can be recovered from any subset of $t \leq n$ elements of S.

A natural approach to constructing a PHE scheme (ignoring the particular model of parallel computation) is therefore to encrypt x by using a RR to transform it into a set $(s_1, \ldots, s_n)$ and have each worker i evaluate g on s_i independently. The results can then be sent back to the client who can recover $f(x)$ using the reduction's reconstruction algorithm. As long as at most t workers collude, the RR will guarantee the confidentiality of x and $f(x)$. Unfortunately, there are two problems with this approach. First, as far as we know, the best hiding threshold achieved by any RR is $t \leq (n-1)/q$, which is for univariate polynomials of degree q [2,3]. In the context of cloud computing, however, this is not a reasonable assumption as the cloud provider owns *all* the machines in the cluster. [2] Another limitation is that the client has to run the RR's reconstruction algorithm which can represent a non-trivial amount of work depending on the particular scheme and the parameters used.

We address these limitations in the following way. First, we show how to construct *fully-hiding* RRs, i.e., reductions with a hiding threshold of $t = n$. Our first construction is for the class of univariate polynomials while the second is for multivariate polynomials with a "small" (i.e., poly-logarithmic in the security parameter) number of variables. As far as we know, these are the first RRs to achieve a threshold of $t = n$. Towards handling the second limitation, we observe that if the recovery algorithm of the RR can be evaluated homomorphically, then the reconstruction step can also be outsourced to the workers. Clearly, using FHE any recovery algorithm can be outsourced, but our goal here is to avoid the use of FHE so as to have practical schemes. Our approach therefore will be to design RRs with recovery algorithms that are either (1) simple enough to be evaluated without FHE; or (2) efficient enough to be run by the client. We note that in cases where the reconstruction algorithm can be outsourced to the workers, we can make use of RRs with reconstruction algorithms that are more expensive than evaluating $f(x)$ directly.

Designing fully-hiding RRs. The best known RRs for polynomials [2,3] work roughly as follows. Let $\mathbf{Q}$ be the polynomial of degree q that we wish to evaluate and $\mathbf{x} \in \mathbb{F}^m$ be the input. First, each element of $\mathbf{x}$ is shared into $q \cdot t + 1$ shares using Shamir secret sharing with a sharing polynomial of degree t (i.e., the hiding threshold). This yields m sets of

[2] Of course one could use the above approach with more than one cloud providers if they do not collude.

shares $(\mathbf{s}_1, \ldots, \mathbf{s}_m)$, where $\mathbf{s}_i = (\mathbf{s}_i[1], \ldots, \mathbf{s}_i[q \cdot t + 1])$. Each worker $j \in [q \cdot t + 1]$ is then given $(\mathbf{s}_1[j], \ldots, \mathbf{s}_m[j])$ and evaluates $\mathbf{Q}$ on his shares. Given the results of all these evaluations, the client interpolates at 0 to recover $\mathbf{Q}(\mathbf{x})$. This approach yields a hiding threshold of up to $t = (n-1)/q$. Note that this construction works equally as well for $m = 1$. As shown in [2,3], this can be improved to $t = n \cdot c\log(m)/m$ for any constant $c > 0$ and $m > 1$.

Due to their reliance on secret sharing, it is not clear how to extend the techniques from [2,3] to achieve $t = n$ and (informally) it seems hard to imagine using any technique based on secret sharing to achieve full hiding. Instead, we introduce two new techniques for designing RRs. The first works for univariate polynomials and makes use of permutation polynomials over finite fields (i.e., bijective families of polynomials). The resulting RR is information-theoretically secure and very efficient. Our second approach is only computationally-secure but works for multivariate polynomials. The security of the RR is based on the multi-dimensional noisy curve reconstruction assumption [13,17].

Resulting PHE schemes. Using our fully-hiding RRs we get PHE schemes for univariate and multi-variate polynomials (with a small number of variables). We stress, however, that PHE schemes for univariate polynomials can be constructed without going through our RR-based approach. In fact, in the full version of this work we give an example of such a construction based only on HE schemes that support addition and a single multiplication [4,11]. This particular construction is very simple and slightly more efficient (i.e., by a constant factor) with respect to client-side work than our PHE scheme for univariate polynomials. We stress, however, that our RR-based approach is more general and yields schemes for more than just univariate polynomials. Since the focus of our work is on our RR-based approach to PHE, we only describe here the construction that results from our RR for univariate polynomials and omit the "simple" construction.

1.2 Our Contributions

While (sequential) homomorphic encryption constitutes an important step towards private outsourced computation, an increasing fraction of the computations performed "in the cloud" is on massive datasets and therefore requires the computation to be performed on clusters of machines. To address this, we make the following contributions:

1. We initiate the study of PHE . In particular, we consider the MapReduce model of parallel computation and formalize MapReduce-parallel HE schemes. Given the practical importance of the MapReduce model and the emergence of cloud-based MapReduce clusters, we believe the study of MapReduce-parallel HE to be important and well motivated.
2. We construct new RRs for univariate and multivariate polynomials with a small number of variables (i.e., polylogarithmic in the security parameter). Our reduction for univariate polynomials is information theoretically secure while our reduction for multivariate polynomials is secure based on the multi-dimensional noisy curve reconstruction assumption [13]. Both our constructions achieve a hiding threshold of $t = n$ and are, as far as we know, the first constructions to do so.
3. We give a general transformation from any RR to a MR-parallel HE scheme given any public-key HE scheme that can evaluate the reductions' recovery algorithm.

If the RR works for any function within a class $\mathcal{C}$, then the resulting MR-parallel scheme is $\mathcal{C}$-homomorphic.

Due to space limitations, we are not able to include all our results. In the full version of this work, we also consider and formalize the notion of *delegated* PHE (which also hides the function being evaluated) and give a delegated construction for any function with output values that can be computed by evaluating a (fixed) univariate polynomial over the input values. We also give optimized variants of our (non-delegated) MR-parallel HE schemes for both univariate and multi-variate polynomials. Finally, we show how, using techniques from [15] and [9], our MR-PHE schemes can be used to perform various queries over encrypted databases like set membership testing, disjunctions queries and keyword search.

2 Preliminaries and Notation

Polynomials. If p is a univariate polynomial of degree d over a field $\mathbb{F}$, then it can be written as $p(x) = \sum_{\alpha \in S} p(\alpha) \cdot \mathcal{L}_\alpha(x)$, where S is an arbitrary subset of $\mathbb{F}$ of size $d+1$ and $\mathcal{L}_\alpha$ is the Lagrangian coefficient defined as $\mathcal{L}_\alpha(x) = \prod_{i \in S, i \neq \alpha}(x-i)/(\alpha - i)$. A permutation polynomial $p \in \mathbb{F}[x]$ is a bijection over $\mathbb{F}$. One class of permutation polynomials which will make use of in this work are the Dickson polynomials (of the first kind) which are a family of polynomials $\mathbf{D} = \{\mathbf{D}_{d,\beta}\}$ over a finite field $\mathbb{F}$ indexed by a degree $d > 0$ and a non-zero element $\beta \in \mathbb{F}$. If $|\mathbb{F}|^2 - 1$ is relatively prime to d and if $\beta \neq 0$, then the Dicskon polynomial $\mathbf{D}_{d,\beta}$ defined as

$$\mathbf{D}_{d,\beta}(x) \stackrel{def}{=} \mathbf{D}_d(x,\beta) = \sum_{\lambda=0}^{\lfloor d/2 \rfloor} \frac{d}{d-\lambda} \cdot \binom{d-\lambda}{\lambda} \cdot (-\beta)^\lambda x^{d-2\lambda},$$

is a permutation over $\mathbb{F}$. For $d = 2$ and any $\beta \neq 0$, we have $\mathbf{D}_{2,\beta}(x) = x^2 - 2\beta$ which is a permutation over any $\mathbb{F}$ such that $|\mathbb{F}|^2 - 1$ is odd.

Homomorphic encryption. Let $\mathcal{F}$ be a family of n-ary functions. A $\mathcal{F}$-homomorphic encryption scheme is a set of four polynomial-time algorithms $\mathsf{HE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{Dec})$ such that Gen is a probabilistic algorithm that takes as input a security parameter k and outputs a secret key K; Enc is a probabilistic algorithm that takes as input a key K and an n-bit message m and outputs a ciphertext c; Eval is a (possibly probabilistic) algorithm that takes as input a function $f \in \mathcal{F}$ and n encryptions $(c_1, \ldots, c_n)$ of messages $(m_1, \ldots, m_n)$ and outputs an encryption c of $f(m_1, \ldots, m_n)$; and Dec is a deterministic algorithm takes as input a key K and a ciphertext c and outputs a message m. In this work, we make use of 2DNF-HE schemes which support an arbitrary number of additions and a single multiplication. Concrete instantiations of such schemes include [4] and [11].

3 MapReduce-Parallel Homomorphic Encryption

In this section, we first give an overview of the MapReduce model of computation together with an example of a simple MapReduce algorithm. We refer the reader to [7,16] for a more detailed exposition. After formalizing the MapReduce model, we define MapReduce-parallel HE schemes and present our security definitions for standard and delegated MR-parallel HE schemes.

3.1 The MapReduce Model of Computation

At a high level, MapReduce works by applying a map operation to the data which results in a set of label/value pairs. The map operation is applied in *parallel* and the resulting pairs are routed to a set of reducers. All pairs with the same label are routed to the same reducer which is then tasked with applying a reduce operation that combines the values into a single value for that label.

A MapReduce algorithm $\Pi = (\mathsf{Parse}, \mathsf{Map}, \mathsf{Red}, \mathsf{Merge})$ is executed on a cluster of w workers and one controller as follows. The client provides a function f and an input x to the controller who runs Parse on (f, x), resulting in a sequence of input pairs $(\ell_i, v_i)_i$. Each pair is then assigned by the controller to a worker that evaluates the Map algorithm on it. This results in a sequence of intermediate pairs $\{(\lambda_j, \gamma_j)\}_j$. Note that since the Map algorithm is stateless, it can be executed in parallel. Typically the number of input pairs is much larger than the number of workers so this stage may require several rounds. When all the input pairs have been processed, the controller partitions all the intermediate pairs and each set of the partition is then assigned to a worker that applies the Red algorithm on it. Again, since Red is stateless it can be executed in parallel (though it can be sequential on its own partition). The outputs of all these Red executions are then processed using Merge and the final result is returned to the client. At any time, a worker is either executing the Map algorithm (in which case it is a *mapper*) or the Red algorithm (in which case it is a *reducer*).

An example. A simple example of a MapReduce algorithm is to determine frequency counts, i.e., the number times a keyword occurs in a document collection. The parse algorithm takes the document collection $(D_1, \ldots, D_n)$ as input and outputs a set of input pairs $(i, D_i)_i$. Each mapper receives an input pair (i, D_i) and outputs a set of intermediate pairs $(w_j, 1)_j$ for each word w_j found in D_i. All the intermediate pairs are then partitioned by the partition operation into sets $\{P_l\}$, where P_l consists of all the intermediate pairs with label w_l. The reducers receive a set P_l of intermediate pairs and sum the values of each pair. The result is a count of the number of times the word w_l occurs in the document collection. The merge algorithm then concatenates all these counts and returns the result.

3.2 Syntax and Security Definitions

An MR-parallel HE scheme is a HE whose evaluation operation can be computed using a MapReduce algorithm.

Definition 1 (MR-parallel HE). *A private-key MR-parallel $\mathcal{F}$-homomorphic encryption scheme is a tuple of polynomial-time algorithms* $\mathsf{PHE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{Dec})$*, where* $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *are as in a private-key encryption scheme and* $\mathsf{Eval} = (\mathsf{Parse}, \mathsf{Map}, \mathsf{Red}, \mathsf{Merge})$ *is a MapReduce algorithm. More precisely we have:*

$K \leftarrow \mathsf{Gen}(1^k)$*: is a probabilistic algorithm that takes as input a security parameter k and that returns a key K.*

$c \leftarrow \mathsf{Enc}(K, x)$*: is a probabilistic algorithm that takes as input a key K and an input x from some message space X, and that returns a ciphertext c. We sometimes write this as $c \leftarrow \mathsf{Enc}_K(x)$.*

$(\ell_i, v_i)_i \leftarrow \mathsf{Parse}(f, c)$*: is a deterministic algorithm that takes as input a function $f \in \mathcal{F}$ and a ciphertext c, and that returns a sequence of input pairs.*

$(\lambda_j, \gamma_j)_j \leftarrow \mathsf{Map}(\ell, v)$*: is a (possibly probabilistic) algorithm that takes an input pair* (ℓ, v) *and that returns a sequence of intermediate pairs.*

$(\lambda, z) \leftarrow \mathsf{Red}(\lambda, P)$*: is a (possibly probabilistic) algorithm that takes a label* λ *and a partition* P *of intermediate values and returns an output pair* (λ, z)*.*

$c' \leftarrow \mathsf{Merge}\big((\lambda_t, z_t)_t\big)$*: is a deterministic algorithm that takes as input a set of output pairs and returns a ciphertext* c'*.*

$y \leftarrow \mathsf{Dec}(K, c')$*: is a deterministic algorithm that takes a key* K *and a ciphertext* c' *and that returns an output* y*. We sometimes write this as* $y \leftarrow \mathsf{Dec}_K(c')$*.*

We say that PHE *is correct if for all* $k \in \mathbb{N}$*, for all* $f \in \mathcal{F}_k$*, for all* K *output by* $\mathsf{Gen}(1^k)$*, for all* $x \in \mathsf{X}$*, for all* c *output by* $\mathsf{Enc}_K(x)$*,* $\mathsf{Dec}_K\big(\mathsf{Eval}(f, c)\big) = f(x)$*.*

To be usable in the setting of private outsourced computation, a PHE scheme should guarantee that its ciphertexts reveal no useful information about the input x or the output $f(x)$. We note that in this setting it is sufficient for this to hold with respect to a *single* input. In the context of outsourced computation, as opposed that of secure communication, the cost of generating a new key per input is negligible. As such, our security definitions only guarantee security for a single input (which could be, e.g., a massive dataset).

Definition 2 **(CPA^1-security).** *Let* $\mathsf{PHE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Parse}, \mathsf{Map}, \mathsf{Red}, \mathsf{Merge}, \mathsf{Dec})$ *be a MR-parallel* $\mathcal{F}$*-homomorphic encryption scheme and consider the following probabilistic experiments where* $\mathcal{A}$ *is an adversary and* $\mathcal{S}$ *is a simulator:*

$\mathbf{Real}_{\mathsf{PHE},\mathcal{A}}(k)$*: the challenger begins by running* $\mathsf{Gen}(1^k)$ *to generate a key* K*.* $\mathcal{A}$ *outputs an input* x *and receives a ciphertext* $c \leftarrow \mathsf{Enc}_K(x)$ *from the challenger.* $\mathcal{A}$ *returns a bit* b *that is output by the experiment.*

$\mathbf{Ideal}_{\mathsf{PHE},\mathcal{A},\mathcal{S}}(k)$*:* $\mathcal{A}$ *outputs an input* x*. Given* $|x|$*,* $\mathcal{S}$ *generates and returns a ciphertext* c *to* $\mathcal{A}$*.* $\mathcal{A}$ *returns a bit* b *that is output by the experiment.*

We say that PHE *is secure against a single-message chosen-plaintext attack if for all* PPT *adversaries* $\mathcal{A}$*, there exists a* PPT *simulator* $\mathcal{S}$ *such that*

$$|\Pr[\,\mathbf{Real}_{\mathsf{PHE},\mathcal{A}}(k) = 1\,] - \Pr[\,\mathbf{Ideal}_{\mathsf{PHE},\mathcal{A},\mathcal{S}}(k) = 1\,]| \leq \mathsf{negl}(k),$$

where the probabilities are over the coins of Enc*,* $\mathcal{A}$ *and* $\mathcal{S}$*.*

4 Randomized Reductions for Polynomials

In this section, we formally define randomized reductions [1,2,3] and then present our fully-hiding constructions for univariate and multivariate polynomials. Our definitions follow closely the ones given by Beaver, Feigenbaum, Killian and Rogaway [3].

Let $t, n \in \mathbb{N}$ such that $t \leq n$. A function $f : X \to Y$ is (t, n)-locally random reducible to a function $g : \widetilde{X} \to \widetilde{Y}$ if there exists two polynomial-time algorithms $\mathsf{RR} = (\mathsf{Scatter}, \mathsf{Recon})$ that work as follows. Scatter is a probabilistic algorithm that takes as input an element $x \in X$ and a parameter $n \in \mathbb{N}$, and returns a sequence $\mathbf{s} \in \widetilde{X}^n$ and some state information st. Recon is a deterministic algorithm that takes as input some state st and a sequence $\mathbf{y} \in \widetilde{Y}^n$ and returns an element $y \in Y$. In addition, we require that RR satisfy the following properties:

– (Correctness) for all $x \in X$,

$$\Pr\left[\,\mathsf{Recon}\big(st, g(s_1), \ldots, g(s_n)\big) = f(x) : (\mathbf{s}, st) \leftarrow \mathsf{Scatter}(x, n)\,\right] \geq 3/4,$$

where the probability is over the coins of Scatter. We depart slightly from the original definition [3] in that here Recon does not need to take x as input.

– (t-hiding) for all $I \subseteq [n]$ such that $|I| = t$, and all x_1 and x_2 in X such that $|x_1| = |x_2|$,

$$\left\{\langle s_i\rangle_{i\in I} : (\mathbf{s}, st) \leftarrow \mathsf{Scatter}(x_1, n)\right\} \approx \left\{\langle s_i\rangle_{i\in I} : (\mathbf{s}, st) \leftarrow \mathsf{Scatter}(x_2, n)\right\}$$

where the distributions are over the coins of Scatter. If $t = n$, we sometimes say that f is *fully* hiding. If the distributions are identically distributed we say that f is *perfectly* hiding, and if the distributions are computationally indistinguishable we say f is *computationally* hiding.

– (Efficiency) for all $x \in X$ and all $\mathbf{s}$ and st output by $\mathsf{Scatter}(x, n)$, the time to evaluate $\mathsf{Recon}(st, g(s_1), \ldots, g(s_n))$ is less than the time to evaluate $f(x)$.

If $g \neq f$ then RR is a *local random reduction* (LRR). If $g = f$, then RR is a *randomized self reduction* (RSR). Furthermore, if there exists a pair of algorithms $\mathsf{RSR} = (\mathsf{Scatter}, \mathsf{Recon})$, such that for every function f in some class $\mathcal{C}$, RSR is a random self reduction for f, then we say that RSR is a *universal* random self reduction over $\mathcal{C}$. All of our constructions are universal.

A note on efficiency. For our purposes, the efficiency requirement is not necessary. This is because in our MR-PHE constructions, the Recon algorithm is not executed by the client but, instead, is executed homomorphically by the cluster. As such, a more important requirement for us is that Recon to be "simple" enough so that it can be evaluated homomorphically without making use of FHE.

4.1 A Perfect Randomized Self Reduction for Univariate Polynomials

In this section, we present a *fully*-hiding randomized reduction for univariate polynomials. As far as we know, the best hiding threshold previously achieved by any RR for univariate polynomials is $t \leq (n-1)/q$ which is achieved by the construction of Beaver, Feigenbaum, Killian and Rogaway [2,3]. Like the construction presented in [2,3], our randomized reduction is *universal* and *self-reducing*.

Let $\mathbf{Q}$ be a degree q univariate polynomial over a finite field $\mathbb{F}$ such that $|\mathbb{F}| \geq 2q + 1$ and $|\mathbb{F}|^2 - 1 \equiv 1 \pmod 2$, and let $\delta[\mathbb{F}^n] \stackrel{def}{=} \big\{\mathbf{v} \in \mathbb{F}^n : v_i \neq v_j \text{ for all } i, j \in [n]\big\}$. Consider the random self reduction $\mathsf{Poly}^1_{\mathsf{q}} = (\mathsf{Scatter}_q, \mathsf{Recon}_q)$ for $\mathbf{Q}$ defined as follows:

– $\mathsf{Scatter}_q(x)$: let $n = 2q+1$ and sample a vector $\boldsymbol{\alpha}$ uniformly at random from $\delta[\mathbb{F}^n]$. For all $i \in [n]$, compute $s_i := \mathbf{D}_2(\alpha_i, -x/2) = \alpha_i^2 + x$. Output $(s_1, \ldots, s_n)$ and $st = \boldsymbol{\alpha}$.

– $\mathsf{Recon}_q(st, y_1, \ldots, y_n)$: output $y = \sum_{i=1}^{n} y_i \cdot \mathcal{L}_{\alpha_i}(0)$.

Theorem 1. $\mathsf{Poly}^1_{\mathsf{q}}$ *is a perfect and fully-hiding randomized self reduction.*

Proof. Towards showing correctness, let $\widehat{\mathbf{Q}}(\alpha) \stackrel{def}{=} \mathbf{Q}\big(\mathbf{D}_2(\alpha, -x/2)\big)$ (for some $x \in \mathbb{F}$) and note that $\widehat{\mathbf{Q}}(0) = \mathbf{Q}(x)$. We therefore have:

$$y = \sum_{i=1}^{2q+1} y_i \cdot \mathcal{L}_{\alpha_i}(0) = \sum_{i=1}^{2q+1} \mathbf{Q}\big(\mathbf{D}_2(\alpha_i, -x/2)\big) \cdot \mathcal{L}_{\alpha_i}(0) = \sum_{i=1}^{2q+1} \widehat{\mathbf{Q}}(\alpha_i) \cdot \mathcal{L}_{\alpha_i}(0) = \widehat{\mathbf{Q}}(0) = \mathbf{Q}(x),$$

since $\deg(\widehat{\mathbf{Q}}) = 2q$. We now consider perfect hiding. Let $n = 2q + 1$ and note that for fixed $q \in \mathbb{N}$ and $x \in \mathbb{F}$, $\mathsf{Scatter}$ evaluates the vector-valued function $f_{x,q} : \delta[\mathbb{F}^n] \to \delta[\mathbb{F}^n]$ defined as

$$f_{x,q}(\boldsymbol{\alpha}) = \Big(\mathbf{D}_2\big(\alpha_1, -x/2\big), ..., \mathbf{D}_2\big(\alpha_n, -x/2\big)\Big),$$

for a random $\boldsymbol{\alpha}$. Note that $f_{x,q}$ is a permutation over $\delta[\mathbb{F}^n]$ since $\mathbf{D}_2(\alpha, \beta)$ is a permutation over $\mathbb{F}$ for any β (this follows from the fact that $|\mathbb{F}|^2 - 1 \equiv 1 \pmod 2$). Let $\mathcal{U}$ be the uniform distribution over $\delta[\mathbb{F}^n]$. In the following, for visual clarity we drop the subscript q and denote $f_{x,q}$ by f_x. For all x_1 and x_2 in $\mathbb{F}$,

$$\begin{aligned}
\mathsf{SD}\big(f_{x_1}(\mathcal{U}), f_{x_2}(\mathcal{U})\big) &= \max_{S \subset \delta[\mathbb{F}^n]} |\Pr\left[\, f_{x_1}(\mathcal{U}) \in S \,\right] - \Pr\left[\, f_{x_2}(\mathcal{U}) \in S \,\right]| \\
&= \max_{S \subset \delta[\mathbb{F}^n]} \left|\Pr\left[\, \mathcal{U} \in f_{x_1}^{-1}(S) \,\right] - \Pr\left[\, \mathcal{U} \in f_{x_2}^{-1}(S) \,\right]\right| \\
&\leq \max_{V, V' \subset \delta[\mathbb{F}^n]} |\Pr\left[\, \mathcal{U} \in V \,\right] - \Pr\left[\, \mathcal{U} \in V' \,\right]| \\
&= 0
\end{aligned}$$

where the last equality follows from the fact that $|V| = |V'|$ since f_{x_1} and f_{x_2} are permutations over $\delta[\mathbb{F}^n]$.

4.2 A Computational Randomized Self Reduction for Multivariate Polynomials

We now present a fully-hiding RSR for multi-variate polynomials. The best known hiding threshold previously achieved is from a construction of [2,3] which achieves $t \leq n \cdot c\log(m)/m$ for c and m greater than 1. Our construction is universal and self-reducing.

Let $\mathbf{Q}$ be a m-variate degree q polynomial over a finite field $\mathbb{F}$ such that $|\mathbb{F}| \geq n + 1$, for $n \in \mathbb{N}$. Consider the randomized self reduction $\mathsf{Poly}^{\mathsf{m}}_{\mathsf{q}} = (\mathsf{Scatter}_q, \mathsf{Recon}_q)$ defined as follows:

- $\mathsf{Scatter}_q(\mathbf{x})$: let $n = 2q + 1$ and sample m univariate polynomials $(p_1, \ldots, p_m)$ of degree 2 such that $p_i(0) = x_i$ for all $i \in [m]$. Let $N = \omega(n \cdot (n/q)^m)$ and $\boldsymbol{\alpha} \stackrel{\$}{\leftarrow} \delta[\mathbb{F}^n]$. For all $j \in [n]$, set $\mathbf{z}_j := \big(p_1(\alpha_j), \ldots, p_m(\alpha_j)\big)$ and for all $j \in [n+1, n+N]$ set $\mathbf{z}_j \stackrel{\$}{\leftarrow} \mathbb{F}^m$. Let $\mathbf{S} = (\mathbf{s}_1, \ldots, \mathbf{s}_{n+N})$ be the sequence that results from permuting the elements of $\mathbf{Z} = (\mathbf{z}_1, \ldots, \mathbf{z}_{n+N})$ at random and let Γ be the locations in $\mathbf{S}$ of the elements in $\mathbf{Z}$ that were chosen at random in $\mathbb{F}^m$. Output $\mathbf{S}$ and $st = (\pi(\boldsymbol{\alpha}), \Gamma)$, where π denotes the (random) permutation used to permute $\mathbf{Z}$.
- $\mathsf{Recon}_{m,q}(st, y_1, \ldots, y_{n+N})$: parse st as $(\boldsymbol{\alpha}, \Gamma)$ and output $y = \sum_{i \notin \Gamma} y_i \cdot \mathcal{L}_{\alpha_i}(0)$.

The security of our randomized reduction is based on the multi-dimensional noisy curve reconstruction assumption from Ishai, Kushilevitz, Ostrovsky and Sahai [13], which extends the polynomial reconstruction (PR) assumption from Naor and Pinkas [17].

Assumption 2 (Multi-dimensional noisy curve reconstruction [13,17]) *The multi dimensional noisy curve reconstruction (CR) assumption is defined in terms of the following experiment where* $\mathbf{x}$ *is a* m*-dimensional vector over a finite field* $\mathbb{F}$*,* $d > 1$*, and* $t = t(k)$ *and* $z = z(k)$ *are functions of* k*:*

$\mathbf{CurveRec}(k, \mathbf{x}, d, n, N, m)$*: sample a vector* $\boldsymbol{\alpha} \xleftarrow{\$} \mathbb{F}^n$ *and a random subset of* N *indices* Γ *chosen from* $[n+N]$*. Choose* m *random univariate polynomials* $(p_1, \ldots, p_m)$ *such that each* p_i *is of degree at most* d *and that* $p_i(0) = x_i$*. For all* $j \in [n]$*, set* $\mathbf{z}_j = (p_1(\alpha_j), \ldots, p_m(\alpha_j))$ *and for all* $j \in [n+1, n+N]$ *set* $\mathbf{z}_j \xleftarrow{\$} \mathbb{F}^m$*. Let* $\mathbf{S} = (\mathbf{s}_1, \ldots, \mathbf{s}_{n+N})$ *be the sequence that results from permuting the elements of* $\mathbf{Z} = (\mathbf{z}_1, \ldots, \mathbf{z}_{n+N})$ *uniformly at random. The output of the experiment is* $(\mathbf{s}_1, \ldots, \mathbf{s}_{n+N})$*.*

We say that the CR assumption holds over $\mathbb{F}$ *with parameters* (d, n, N, m) *if for all* $\mathbf{x}_1$ *and* $\mathbf{x}_2$ *in* $\mathbb{F}^m$*,*

$$\left\{\mathbf{CurveRec}(k, \mathbf{x}_1, d, n, N, m)\right\} \stackrel{c}{\approx} \left\{\mathbf{CurveRec}(k, \mathbf{x}_2, d, n, N, m)\right\}$$

We note that the CR assumption is believed to hold when N *is* $\omega(n \cdot (n/d)^m)$ *and* $|\mathbb{F}| = N$ *[13].*

Remark. Setting n and N to be polynomial in k, the CR assumption is believed to hold as long as $m = \mathsf{polylog}(k)$. We note, however, that the parameters provided in [13] and used in this work are for the *stronger* "augmented CR" assumption which outputs, in addition to the vectors $(\mathbf{s}_1, \ldots, \mathbf{s}_{n+N})$, the evaluation points $(\alpha_1, \ldots, \alpha_n)$ together with N random values. It is therefore plausible that the CR assumption could hold for a wider range of parameters and, in particular, for $m = \mathsf{poly}(k)$.

In the following theorem, we show that $\mathsf{Poly}_{\mathsf{q}}^{\mathsf{m}}$ is a fully-hiding and universal RSR for the class of multivariate polynomials with a poly-logarithmic number of variables.

Theorem 3. $\mathsf{Poly}_{\mathsf{q}}^{\mathsf{m}}$ *is a computational and fully-hiding random self reduction.*

The proof follows almost directly from Assumption 2, so due to space limitations, it is deferred to the full version of this work.

5 MR-Parallel HE from Randomized Reductions

We now show how to construct a MR-parallel HE scheme from any $\mathcal{F}$-homomorphic encryption scheme and any fully-hiding RR between functions f and g whose reconstruction algorithm is in $\mathcal{F}$. At a high-level, the construction works as follows.

The RR's scatter algorithm is applied to each element x_i of the input $\mathbf{x}$. This results in a sequence $\mathbf{s}_i$ and a state st_i. The latter is encrypted using the $\mathcal{F}$-homomorphic encryption scheme and each mapper receives a pair composed of a label $\ell = i$ and a value v of the form $(s_i[j], e_i)$ for $i \in [\#\mathbf{x}]$ and $j \in [n]$ and where e_i is an $\mathcal{F}$-homomorphic encryption of st_i. The mapper evaluates g on $s_i[j]$ and returns an intermediate pair with label

Let $\mathsf{HE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{Dec})$ be a public-key $\mathcal{F}$-homomorphic encryption scheme and let $\mathsf{RR} = (\mathsf{Scatter}, \mathsf{Recon})$ be a $\mathcal{C}$-universal (t, n)-local randomized reduction from f to g such that $\mathsf{Recon} \in \mathcal{F}$. Consider the multi-use MR-parallel $\mathcal{C}$-homomorphic encryption scheme $\mathsf{PHE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{Dec})$, where $\mathsf{PHE.Eval} = (\mathsf{Parse}, \mathsf{Map}, \mathsf{Red}, \mathsf{Merge})$, defined as follows:

- $\mathsf{Gen}(1^k)$: compute $(pk, sk) \leftarrow \mathsf{HE.Gen}(1^k)$. Output $K = (sk, pk)$.
- $\mathsf{Enc}(K, \mathbf{x})$: for all $i \in [\#\mathbf{x}]$, compute $(\mathbf{s}_i, st_i) \leftarrow \mathsf{Scatter}(x_i)$ and $e_i \leftarrow \mathsf{HE.Enc}_{pk}(st_i)$. Output $\mathbf{c} = (pk, \mathbf{s}_1, \ldots, \mathbf{s}_{\#\mathbf{x}}, e_1, \ldots, e_{\#\mathbf{x}})$.
- $\mathsf{Parse}(f, \mathbf{c})$: for all $i \in [\#\mathbf{x}]$ and $j \in [n]$, set $\ell_{i,j} := i$ and $v_{i,j} := (f, pk, \mathbf{s}_i[j], e_i)$. Output $(\ell_{i,j}, v_{i,j})_{i,j}$.
- $\mathsf{Map}(\ell, v)$: parse v as (f, s, e) and compute $a \leftarrow \mathsf{HE.Enc}_{pk}(g(s))$. Output $\lambda := \ell$ and $\gamma := (a, e)$.
- $\mathsf{Red}(\lambda, P)$: parse P as $(a_r, e_r)_r$ and compute $z \leftarrow \mathsf{HE.Eval}(\mathsf{Recon}, e_r, (a_r)_r)$. Output (λ, z).
- $\mathsf{Merge}\big((\lambda_t, z_t)_t\big)$: output $\mathbf{c}' := (z_t)_t$.
- $\mathsf{Dec}(K, \mathbf{c}')$: for all $i \in [\#\mathbf{c}']$, compute $y_i := \mathsf{HE.Dec}_{sk}(z_i)$. Output $\mathbf{y} = (y_1, \ldots, y_{\#\mathbf{c}'})$.

Fig. 1. MR-parallel HE from RR and HE

$\lambda = i$ and value $\gamma = \big(g(s_i[j]), e_i\big)$. After the shuffle operation, each reducer receives a pair composed of a label i and a partition

$$P = \Big(\big(y_{i,j}, e_i\big), \ldots, \big(y_{i,n}, e_i\big) \Big),$$

where $y_{i,j} = g(s_i[j])$ for $j \in [n]$. Since Recon is in $\mathcal{F}$, the reducer can evaluate $\mathsf{Recon}(e_i, y_{i,1}, \ldots, y_{i,n})$ homomorphically which results in an encryption of $f(x_i)$.

Theorem 4. *If* HE *is CPA-secure and if* RR *is fully-hiding, then* PHE *as described in Figure 1 is secure against single-message chosen-plaintext attacks.*

We sketch a proof of Theorem 4 and leave a full proof to the full version of this work. Consider the simulator $\mathcal{S}$ that simulates ciphertexts in an $\mathbf{Ideal}(k)$ experiment as follows. Given $\#\mathbf{x}$ it generates $(pk', sk') \leftarrow \mathsf{Gen}(1^k)$ and, for all $i \in [\#\mathbf{x}]$, it computes $(\mathbf{s}'_i, st'_i) \leftarrow \mathsf{Scatter}(0)$ and $e'_i \leftarrow \mathsf{HE.Enc}_{pk'}(st'_i)$. It outputs $\mathbf{c}' = (pk', \mathbf{s}'_1, \ldots, \mathbf{s}'_{\#\mathbf{x}}, e'_1, \ldots, e'_{\#\mathbf{x}})$. The fully-hiding property of RR guarantees that the $\mathbf{s}'_i$'s are indistinguishable from the $\mathbf{s}_i$'s generated in a $\mathbf{Real}(k)$ experiment. Similarly, the CPA-security of HE guarantees that the e'_i's are indistinguishable from the e_i's generated in a $\mathbf{Real}(k)$ experiment.

Direct constructions. By instantiating the RR and the HE scheme in our general construction with our fully-hiding RSR for univariate polynomials (from section 4.1) and an FHE scheme, we get a multi-use MR-parallel HE scheme for the class of functions whose output values can be computed by evaluating a (fixed) univariate polynomial of the inputs. In addition, the resulting construction can be made delegated by encrypting the coefficients of the polynomial using the FHE scheme and having the mappers perform their computations homomorphically. Current FHE constructions, however, are not

yet practical enough for our purposes so, in the full version, we present a direct construction based only on additively homomorphic encryption. The construction can be made delegated if we use 2DNF-HE. The direct construction also has the advantage that the input pairs sent to the mappers are smaller than what would result from our general construction.

Similarly, if we instantiate our general construction with our RR for multi-variate polynomials (from Section 4.2) and an FHE scheme, we get an MR-parallel HE scheme for the class of functions whose output values can be computed by evaluating a (fixed) multi-variate polynomial on the inputs (with small number of variables). To avoid the use of FHE, however, we present in the full version of this work a direct construction that only makes use of additively HE.

References

1. D. Beaver and J. Feigenbaum. Hiding instances in multioracle queries. In *Symposium on Theoretical aspects of Computer Science (STACS '90)*, 1990.
2. D. Beaver, J. Feigenbaum, J. Kilian, and P. Rogaway. Security with low communication overhead. In *Advances in Cryptology - CRYPTO '90*, 1991.
3. D. Beaver, J. Feigenbaum, J. Kilian, and P. Rogaway. Locally random reductions: Improvements and applications. *Journal of Cryptology*, 10(1), 1997.
4. D. Boneh, E.-J. Goh, and K. Nissim. Evaluating 2-dnf formulas on ciphertexts. In *Theory of Cryptography Conference (TCC '05)*, volume 3378 of *Lecture Notes in Computer Science*, 2005.
5. A. Borodin. On relating time and space to size and depth. *SIAM J. of Comp.*, 6(4), 1977.
6. D. Culler, R. Karp, D. Patterson, A. Sahay, E. Santos, K. Schauser, R. Subramonian, and T. von Eicken. Logp: a practical model of parallel computation. *Comm. of the ACM*, 39(11), 1996.
7. J. Dean and S. Ghemawat. Mapreduce: simplified data processing on large clusters. In *Symposium on Opearting Systems Design & Implementation*, 2004.
8. S. Fortune and J. Wyllie. Parallelism in random access machines. In *ACM Symposium on Theory of Computing (STOC '78)*, 1978.
9. M. Freedman, Y. Ishai, B. Pinkas, and O. Reingold. Keyword search and obliviosu pseudorandom functions. In *Theory of Cryptography Conference (TCC '05)*, 2005.
10. C. Gentry. Fully homomorphic encryption using ideal lattices. In *ACM Symposium on Theory of Computing (STOC '09)*, 2009.
11. C. Gentry, S. Halevi, and V. Vaikuntanathan. A simple bgn-type cryptosystem from lwe. In *Advances in Cryptology - EUROCRYPT '10*, 2010.
12. M. Isard, M. Budiu, Y. Yu, A. Birrell, and D. Fetterly. Dryad: distributed data-parallel programs from sequential building blocks. In *ACM SIGOPS/EuroSys European Conference on Computer Systems (EuroSys '07)*, 2007.
13. Y. Ishai, E. Kushilevitz, R. Ostrovsky, and A. Sahai. Cryptography from anonymity. In *IEEE Symposium on Foundations of Computer Science (FOCS '06)*, 2006.
14. R. M. Karp and V. Ramachandran. Parallel algorithms for shared-memory machines, 1990.
15. L. Kissner and D. Song. Privacy-preserving set operations. In *Advances in Cryptology - CRYPTO '05*, 2005.
16. J. Lin and C. Dyer. *Data-Intensive Text Processing with MapReduce*. M. & C., 2010.
17. M. Naor and B. Pinkas. Oblivious polynomial evaluation. *SIAM J. of Comp.*, 35(5), 2006.
18. L. Valiant. A bridging model for parallel computation. *Comm. of the ACM*, 33(8), 1990.