Optimal Rounds SMPC via Polynomial Representation and Distributed Random Matrix (Preliminary Version)

by

Dor Bitan and Shlomi Dolev

Technical Report #2019-02

July 2019

Optimal Rounds SMPC via Polynomial Representation and Distributed Random Matrix[⋆] (Preliminary Version)

Dor Bitan[1] and Shlomi Dolev[2]

[1] Dept. of Mathematics, Ben-Gurion University of the Negev, Israel, `dorbi@post.bgu.ac.il`,
[2] Dept. of Computer Science, Ben-Gurion University of the Negev, Israel `dolev@cs.bgu.ac.il`

Abstract. We present a novel cryptographic tool, *Distributed Random Matrix Bootstrapping*, and show how it may be used to achieve MPC schemes of arithmetic functions over finite fields with optimal round complexity. All our schemes assume honest but curious parties and are perfectly secure. We describe a new homomorphic secret sharing scheme, *Distributed Random Matrix*, and show how it may be used to achieve 2PC and MPC with dishonest majority in two rounds of communication assuming the parties posses correlated randomness. That scheme gives rise to a scheme which enables a user to outsource the storage of confidential data to n distrusted servers and have the servers perform computations over the encrypted data in a single round of communication against dishonest majority. We then show how these schemes may be bootstrapped by a set of three or four parties to generate the correlated randomness required for the first scheme themselves while maintaining perfect security under an honest majority.

Our approach deviates from standard conventions of MPC that is based on computing circuits with exponential in the depth of the circuit overhead. We consider a representation of f as a multivariate polynomial (rather than an arithmetic circuit). Second, we divide the problem into two scenarios. We begin with solving the *non-vanishing* scenario, in which the secrets are non-zero elements of $\mathbb{F}_p$. Then, we address the general case by showing that f is *q-bounded* by a prime number q. Though all arithmetic functions are q-bounded for some q, in the worst case, q might be exponentially larger than p. The round complexity of our schemes is optimal, and the communication complexity of our schemes is quadratic in the number of parties, linear in the size of the polynomial and independent of its degree.

⋆ The research was partially supported by the Rita Altura Trust Chair in Computer Sciences; the Lynne and William Frankel Center for Computer Science; the Cyber Security Research Center at Ben-Gurion University; the Ministry of Foreign Affairs, Italy; the grant from the Ministry of Science, Technology and Space, Israel; the National Science Council (NSC) of Taiwan; the Ministry of Science, Technology and Space, Infrastructure Research in the Field of Advanced Computing and Cyber Security; and the Israel National Cyber Bureau.

Keywords: Secure multiparty computation, Two-party computation, Perfect security, Optimal round complexity, Correlated randomness, Bootstrapping

1 Introduction

Secure multiparty computation (SMPC) is an extensively studied field in cryptography which discusses the following problem. N parties, $\mathcal{P}_1, \ldots, \mathcal{P}_N$, are holding secret inputs, $s_1, \ldots, s_N$, and wish to evaluate a function $y = f(x_1, \ldots, x_n)$ over their inputs while not revealing any information regarding their inputs to each other (except for what may be deduced from the output). Countless papers were written on that topic in the past four decades, e.g., [Yao82,GMW87,CCD88,BMR90,Riv99,DN03,ABT18], suggesting various solutions to that problem based on different assumptions and approaches. Solutions usually assume that the parties are connected via authenticated point-to-point channels (P2P) and that f is given as an arithmetic or boolean circuit. Primary solutions used *rounds of communication* to reduce the degree of the polynomial that encrypts the data after each multiplication during the computation. A round of communication is a phase in which each party may send at most one message to each of the other parties, perform arbitrary computations and/or receive at most one message from each of the other parties (not necessarily in this order) [KN06]. Recent state-of-the-art solutions also require rounds of communication proportional to the depth of the circuit representing f.

In this work, we assume that parties are *honest but curious*. That is, parties follow the protocol, yet, they may attempt to use the information they receive throughout the execution of the protocol to gain information regarding the secret inputs of other parties. Furthermore, a subset of *corrupted* parties may form a coalition, joining the information they hold in an adversarial attempt to reveal the secret inputs of the *honest* parties. For an MPC protocol π, the maximal size of such a coalition of corrupted parties under which the privacy of the inputs of honest parties is maintained is the *threshold* of π, denoted t. In our research, we seek for MPC schemes which are *perfectly secure*. We assume that the (adversarial) parties are computationally unbounded.

In their seminal work from 1988, Ben-Or et al. [BOGW88] showed that every function of N inputs can be efficiently computed by N parties with a threshold $t < N/2$ in case of honest but curious parties. Their methods are based on Shamir's secret sharing scheme [Sha79], and their protocols require rounds of communication proportional to the depth of the arithmetic circuit. Substantial efforts have been spent on reducing the number of rounds of communication needed in such tasks. Bar-Ilan and Beaver [BIB89] were the first to suggest a way to evaluate functions in a constant number of rounds of communication, followed by further works that attempt to minimize that constant number. Recent works by [ABT18,GIS18,ACGJ18] achieve optimal round complexity and present protocols which enable SMPC of functions in two rounds of communication. Comprehensive coverage of results and bounds on the round complexity of SMPC schemes may be found in [PR18,DLN19].

We now state the main results of our work. Assume that $\{\mathcal{P}_j\}_{j=1}^n$ is a set of $n \geq 2$ honest but curious parties. For $1 \leq j \leq n$, let $\{s_j^{(i)}\}_{i=1}^{k_j} \subseteq \mathbb{F}_p$ a set of k_j private inputs of party $\mathcal{P}_j$, and let $k = \sum_{j=1}^n k_j$. We have

Theorem 1. *Let $f : \mathbb{F}_p^k \to \mathbb{F}_p$. There exists an n-party scheme which realizes f using correlated randomness (CR) as follows:*

- *Its communication complexity is*
  - *exponential in k,*
  - *quadratic in n,*
  - *at most linear in the number of monomials in the polynomial representation of f, and*
  - *independent of the degree of the polynomial representation of f.*
- *Its space complexity is at most linear in the number of monomials in the polynomial representation of f, and linear in n;*
- *The correlated randomness (CR) used in it is independent of f and the inputs. The CR may be obtained in one of the following ways:*
  - *provided by a trusted initializer (before the secrets and f are known), or*
  - *generated by the parties themselves in a perfectly secure offline preprocessing phase.*

  In the first case, the scheme is perfectly secure against attacks of coalitions of up to $n-1$ honest but curious parties. In the second case, the scheme is perfectly secure against any coalition which not contain a predetermined party (who plays a unique role in the generation of the CR). In particular, if $n \in \{3, 4\}$ then, the scheme is perfectly secure assuming honest majority.
- *Either way, the round complexity of the online phase of the scheme is two.*

We note that, in Theorem 1, in the *non-vanishing* case, where the secrets are non-zero elements of $\mathbb{F}_p$, if f is given as a polynomial of constant size then, our scheme is highly efficient regardless of the degree of f. Specifically, in the non-vanishing case, the communication complexity of the scheme is linear in k, which is a significant improvement in comparison to the general case (where the communication is exponential in k).

2 Preliminaries

We recall some linear algebra and MPC notations and definitions. Throughout the paper we use $\mathbb{F}_p$ to denote the finite field containing p elements (where p is prime), $\mathbb{F}_p^k$ to denote the k-dimensional vector space over $\mathbb{F}_p$, $\mathbb{F}_p^\times$ to denote the multiplicative group of $\mathbb{F}_p$, and $(\mathbb{F}_p^\times)^k$ to denote the set of k-tuples over $\mathbb{F}_p^\times$. We use $M_n(\mathbb{F}_p)$ to denote the set of square matrices of order n over $\mathbb{F}_p$. $\mathbb{N}$ is the set of natural numbers. The notation $*$ stands for entrywise multiplication of equally length tuples. If α is a k-tuple then $(\alpha)_i$ is the i'th entry of α. If C is a matrix then $[C]_i$ is the i'th column of C. We denote by $x \overset{R}{\leftarrow} A$ the process of assigning to the variable x a uniformly random element from the set A.

Security of MPC schemes. The security of an MPC protocol is formalized and proved through the *Ideal world vs Real world* paradigm. We briefly overview the general idea and recall standard definitions. Let $\mathcal{P} = \{\mathcal{P}_j\}_{j=1}^n$ a set of n parties and assume that each party $\mathcal{P}_j$ is holding a secret value s_j in some (joint) domain $\mathcal{R}$. Assume that the parties wish to find $f(s_1, \ldots, s_n)$, where $f : \mathcal{R}^n \to \mathcal{R}$, while not revealing to each other any information regarding their secret inputs that could not be deduced from $f(s_1, \ldots, s_n)$. In an ideal world, the parties could have find a trusted entity, *Ted*, to whom they will all tell their private inputs and from whom they will receive the output. Ted will perform the computation on their behalf and promise to keep their secrets safe.

In the real world, such a trusted entity is hard to find, and hence, the parties may attempt to perform the computation themselves by following an MPC protocol π. Informally, to consider π secure for computing f, it should have the property that by following it, the parties gain no information regarding the secret inputs of other parties than they could not have learned by following the ideal world solution. Putting it in other words, an MPC protocol is not meant to provide a way of computing $f(s_1, \ldots, s_n)$ while not revealing *any* information regarding the secret inputs of other parties, but to provide a way of computing $f(s_1, \ldots, s_n)$ while not revealing *any more* information regarding the secret inputs of other parties *than* is revealed by an ideal world solution.

We also consider the case in which a subset of the parties, $\mathcal{T} \subseteq \mathcal{P}$, join forces in an adversarial attempt to gain information regarding the secret inputs of parties in $\overline{\mathcal{T}} = \mathcal{P} - \mathcal{T}$. Informally, we would say that π is secure for computing f *with threshold* t if it holds that, for every $\mathcal{T} \subseteq \mathcal{P}$ with $|\mathcal{T}| \leq t$, the parties in $\mathcal{T}$ gain no more information regarding $\{s_i\}_{\mathcal{P}_i \in \overline{\mathcal{T}}}$ from π than they would have got from the real world solution.

To show that π leaks no more information than Ted, we show that all the information gotten from π may be computed from the information received from Ted only. To this end, we define $view_j$ to be the random variable indicating s_j and all the messages that $\mathcal{P}_j$ receives through the execution of π, including the results of random choices that she makes. We assume that all parties are *honest but curious*. That is, they follow the exact instructions of π, though, they attempt to gain information regarding the secret inputs of other parties through the information they receive. That leads us to

Definition 1. ***Perfect security of an MPC protocol against honest but curious adversaries.*** *Let $\mathcal{P} = \{\mathcal{P}_j\}_{j=1}^n$ a set of n parties, $\{s_j\}_{\mathcal{P}_j \in \mathcal{P}} \subseteq \mathcal{R}$ their corresponding secrets, $f : \mathcal{R}^n \to \mathcal{R}$ a function, and $t \in \mathbb{N}$. Let π a multiparty computation protocol for computing f. We say that π is a perfectly secure MPC protocol for computing f against honest but curious adversaries and with threshold t if the following holds.*

1. *Perfect correctness. By executing π, all parties learn $y = f(s_1, \ldots, s_n)$.*

2. *Perfect privacy. For every adversarial coalition $\mathcal{T} \subseteq \mathcal{P}$ with $|\mathcal{T}| \leq t$ there exists a simulator — a probabilistic algorithm* `Sim` *— which receives inputs y and $\{s_j\}_{\mathcal{P}_j \subseteq \mathcal{T}}$, and its output is identically distributed to*

$$view_T = \{view_j\}_{\mathcal{P}_j \in \mathcal{T}}.$$

Minimal multivariate polynomial representation. Any function $f : \mathbb{F}_p^k \to \mathbb{F}_p$ can be represented as a multivariate polynomial. This representation may be obtained, for example, by using the multivariate interpolation formula over finite fields suggested in [CLF12]. The fact that $x^p \equiv x \pmod{p}$ implies that for a given $f : \mathbb{F}_p^k \to \mathbb{F}_p$ there are infinitely many polynomial representations of it.[3] Given a function f, we assign f with a *minimal multivariate polynomial representation* of it, that is, the representation of f as a multivariate polynomial with the degree of each variable being at most $p-1$. We denote this polynomial by Q_f and assign f with Q_f as its minimal multivariate polynomial representation. Throughout the paper we abuse notation and write f instead of Q_f. Whenever a function $f : \mathbb{F}_p^k \to \mathbb{F}_p$ is discussed we assume that f is given with its minimal multivariate polynomial representation. We note that the total degree[4] of Q_f is at most $k(p-1)$ and write

$$f(x) = \sum_{i=(i_1,\ldots,i_k)\in\mathcal{L}} a_i \cdot x_1^{i_1} \ldots x_k^{i_k},$$

where $\mathcal{L} = \{0, \ldots, p-1\}^k$ and $a_i \in \mathbb{F}_p$. The fact that each variable in each monomial can appear with any exponent between 0 and $p-1$ implies that there are p^k different monomials. For $i = (i_1, \ldots, i_k) \in \mathcal{L}$, we denote $x_1^{i_1} \ldots x_k^{i_k}$ by A_i. We refer to A_i as *the i'th monomial of f*.

q-bounded functions. Let $f : \mathbb{F}_p^k \to \mathbb{F}_p$ and $s = (s_1, \ldots, s_k) \in \mathbb{F}_p^k$. One can compute $f(s)$ by performing operations in $\mathbb{F}_q$ on s according to a representation of f as a multivariate polynomial. The same result is obtained if one computes $f(s)$ over the positive integers and then takes the result modulo p. Formally, for each entry s_j of s let $a_j \in \{1, 2, \ldots, p\} \subseteq \mathbb{N}$ denote the minimal positive integer such that $a_j \equiv s_j \pmod{p}$. Then, performing the computation over the a_j's using integer operations we obtain an integer result $f(s)_{\mathbb{N}}$, such that $f(s)_{\mathbb{N}} \equiv f(s) \pmod{p}$. Assume a function $f : \mathbb{F}_q^k \to \mathbb{F}_q$ is such that for every $s \in \mathbb{F}_p^k$, computation of $f(s)$ over the integers yields an integer result, $f(s)_{\mathbb{N}}$, which is strictly smaller than a large prime q. Such a function is *q-bounded.* In fact, all functions $f : \mathbb{F}_p^k \to \mathbb{F}_p$ are q-bounded for $q \geq p^{kp+1}$. In practice, we are interested in the minimal prime q for which f is q-bounded.

[3] Generally, there are infinitely many polynomials of k variables and only a finite number, namely p^{p^k}, of functions $f : \mathbb{F}_p^k \to \mathbb{F}_p$.

[4] The total degree of a multivariate polynomial is the maximal sum of exponents in a single monomial of it.

3 The Distributed Matrix Secret Sharing Scheme

In this section we describe the basic tool of this work, the *Distributed Matrix* procedure, `Dist.Matrix`, and discuss some of its properties. `Dist.Matrix` employs two other basic procedures — `Mult.split` (which will also be used by parties in our schemes to secret share their private inputs) and `Add.split`. We now describe these schemes and discuss some of their properties.

Multiplicative secret sharing procedure. The following procedure is invoked by party $\mathcal{P}_i$ to split $s_i \in \mathbb{F}_p^\times$ into n multiplicative secret shares. Given a prime number p, an element $s \in \mathbb{F}_p$, a natural number $n \in \mathbb{N}$, and $1 \leq i \leq n$, the procedure `Mult.split` is as follows. Pick $n-1$ uniformly random non-zero elements m_j $(1 \leq j \leq n,\, j \neq i)$ from $\mathbb{F}_p$ and set $m_i \in \mathbb{F}_p$ such that $s = \prod_{j=1}^{n} m_j$. Output $(m_1, \ldots, m_n)$, a sequence of *multiplicative shares* of s. Formally:

```
Mult.split($p, s, n, i$):          # {$p$ is prime, $s \in \mathbb{F}_p$, $n \in \mathbb{N}$, $1 \leq i \leq n$}

    For $1 \leq j \leq n, j \neq i$:

        $m_j \xleftarrow{R} \mathbb{F}_p^\times$;

    $\delta \leftarrow \prod_{j=1, j \neq i}^{n} m_j$;

    $m_i \leftarrow \frac{s}{\delta}$

    return $(m_1, \ldots, m_n)$
```

Procedure 1: `Mult.split`*. Given an element $s \in \mathbb{F}_p$, the procedure returns n elements whose product equals s.*

Remark 1. Joint zeroness of s and m_i. Observe that the assignment $m_i \leftarrow \frac{s}{\delta}$ implies that if $s = 0$ then $m_i = 0$ and if $s \neq 0$ then $m_i \neq 0$. All other entries m_j of the output (with $j \neq i$) are uniformly random non-zero elements of $\mathbb{F}_p^\times$.

Lemma 1. *Procedure* `Mult.split` *is a perfectly secure secret sharing scheme for $\mathbb{F}_p^\times$ elements with a threshold of $n-1$ and which supports homomorphic multiplications.*

Proof. Assume that a user, holding a non-zero element $s \in \mathbb{F}_p^\times$, distributes the output of `Mult.split`(p, s, n, i) (for some i) to a set of parties $\{\mathcal{P}_j\}_{j=1}^{n}$. Any coalition of $n-1$ parties gains absolutely no information regarding s since given their shares, for every non-zero element $s' \in \mathbb{F}_p^\times$ there exists a single n'th share for which the product of the n shares equals s'. Now, assume that s_1 and s_2 are two non-zero elements of $\mathbb{F}_p$ that were secret shared by a user using `Mult.split`. From the definition of `Mult.split`, it immediately follows that

$$\prod_{j=1}^{n} \Big(\texttt{Mult.split}(p, s_1, n, i) * \texttt{Mult.split}(p, s_2, n, i') \Big)_j = s_1 \cdot s_2,$$

where $*$ stands for entrywise multiplication of vectors and the subscript j indicates the j'th entry of the vector. The same holds for any number d of shared secrets $s_1, \ldots, s_d$. Hence, if a user secret shares d non-zero elements of $\mathbb{F}_p^\times$ using `Mult.split`, then, having each of the parties compute the product of her shares (locally), each party obtains a multiplicative share of $\prod_{i=1}^{d} s_i$. $\square$

Additive secret sharing procedure. Similarly to the previous procedure, given a prime number p, an element $s \in \mathbb{F}_p$, and a natural number $n \in \mathbb{N}$, the procedure `Add.split` is as follows. Pick $n-1$ uniformly random elements $a_1, \ldots, a_{n-1}$ from $\mathbb{F}_p$ and set $a_n = s - \sum_{i=1}^{n-1} a_i$. Output $(a_1, \ldots, a_n)$, a sequence of *additive shares* of s. Formally:

```
Add.split(p, s, n):            # {p is prime, s ∈ F_p, n ∈ N}
    For $1 \le i \le n-1$:
        $a_i \overset{R}{\leftarrow} \mathbb{F}_p$;
    $a_n \leftarrow \left(s - \sum_{i=1}^{n-1} a_i\right)$;
    return $(a_1, \ldots, a_n)$
```

Procedure 2: `Add.split`*. Given an element $s \in \mathbb{F}_p$, the procedure returns n elements whose sum equals s.*

Lemma 2. *The procedure* `Add.split` *is a perfectly secure secret sharing scheme for $\mathbb{F}_p$ elements with a threshold of $n-1$ and which supports homomorphic additions.*

Proof. Assume that a user, holding an element $s \in \mathbb{F}_p$, distributes the a_j's to a set of parties $\{\mathcal{P}_j\}_{j=1}^n$. Any coalition of $n-1$ parties gains absolutely no information regarding s since given their shares, for every element $s' \in \mathbb{F}_p$ there exists a single n'th share for which the sum of the n shares equals s'. Now, assume that s_1 and s_2 are two elements of $\mathbb{F}_p$ that were secret shared by a user using `Add.split`. From the definition of `Add.split` it immediately follows that

$$\sum_{j=1}^{n} \left(\texttt{Mult.split}(p, s_1, n, i) + \texttt{Mult.split}(p, s_2, n, i')\right)_j = s_1 + s_2.$$

The same holds for any number d of shared secrets $s_1, \ldots, s_d$. Consequently, if a user secret shares d elements of $\mathbb{F}_p$ using `Add.split`, then, having each of the parties locally compute the sum of the shares received, each party obtains an additive share of $\sum_{i=1}^{d} s_i$. $\square$

Distributed matrix secret sharing procedure. We now define the procedure `Dist.Matrix`. Given a prime number p, an element $x \in \mathbb{F}_p$ and a natural number

$n \in \mathbb{N}$, the procedure `Dist.Matrix` outputs (the columns of) a matrix C, a *matrix-random-split of* x. C is generated by performing additive secret sharing of x, followed by multiplicative secret sharing of each of the additive shares. Formally:

```
Dist.Matrix$(p, s, n)$:        # $\{p$ is prime, $s \in \mathbb{F}_p$, $n \in \mathbb{N}\}$

    $(\gamma_1, \ldots, \gamma_n) \leftarrow$ Add.split$(p, s, n)$;

    For $1 \le i \le n$:

        $(c_{i1}, c_{i2}, \ldots, c_{in}) \leftarrow$ Mult.split$(p, \gamma_i, n, i)$ ;

    $C \leftarrow (c_{ij}) \in M_n(\mathbb{F}_p)$;

    return $\big([C]_1, \ldots, [C]_n\big)$
```

Procedure 3: `Dist.Matrix`*. Given an element $s \in \mathbb{F}_p$, the procedure returns n columns of a matrix C.*

Remark 2. Since the i'th row of C is generated by the call `Mult.split`(p, γ_i, n, i), from Remark 1 it follows that the matrix C may contain zeroes only on its main diagonal, if any. That is, $c_{ij} = 0 \implies i = j$.

Reconstruction of a `Dist.matrix`-secret-shared element x from n shares may be performed by simply multiplying all the elements in each row of C and summing the products. Namely,

$$\sum_{i=1}^{n} \prod_{j=1}^{n} c_{ij} = x.$$

To formalize that, we define the following reconstruction procedure.

```
    Reconstruct$\big(p, (v_1, \ldots, v_n), n\big)$ :        # $\{p$ is prime, $v_i \in \mathbb{F}_p^n$, $n \in \mathbb{N}\}$

        return $\sum_{i=1}^{n} \prod_{j=1}^{n} (v_j)_i$
```

Procedure 4: `Reconstruct`*. The procedures reconstructs x from n columns of a square matrix.*

One may readily verify that, for a prime number p, a natural number $n \in \mathbb{N}$ and an element $s \in \mathbb{F}_p$, it holds that `Reconstruct`$\big(p,$ `Dist.Matrix`$(p, s, n), n\big) = s$.

Lemma 3. *The procedure* `Dist.Matrix` *is a perfectly secure secret sharing scheme for $\mathbb{F}_p$ elements with a threshold of $n - 1$ and which supports homomorphic multiplications by* `Mult.split`*-secret-shared elements.*

Proof. Assume that a user, holding an element $x \in \mathbb{F}_p$, distributes the output of `Dist.Matrix`(p, x, n) to a set of parties $\{\mathcal{P}_j\}_{j=1}^n$ in such a way that $\mathcal{P}_j$ receives $[C]_j$. Any coalition of $n-1$ parties gains absolutely no information regarding x since given their shares, for every element $x' \in \mathbb{F}_p$ there exist exactly $(p-1)^{n-1}$ elements of $\mathbb{F}_p^n$ for which the collection of the n shares is a matrix-random-split of x'. Indeed, an appropriate n'th share for x' may be computed as follows. Without loss of generality, assume that such an adversarial coalition contains $\mathcal{P}_1, \ldots, \mathcal{P}_{n-1}$. Choose $n-1$ uniformly random non-zero elements $c_{1,n}, \ldots, c_{n-1,n}$ of $\mathbb{F}_p^\times$ and (given the $n-1$ shares of the coalition) set

$$c_{nn} = \frac{x' - \sum_{i=1}^{n-1} \prod_{j=1}^{n} c_{ij}}{\prod_{j=1}^{n-1} c_{jn}}. \tag{1}$$

One may readily verify that $(c_{1n}, \ldots, c_{nn})$ is a possible n'th share for x'.

Now, to show that the `Dist.Matrix` secret sharing scheme supports homomorphic multiplications by `Mult.split`-secret-shred non-zero elements, assume that a user, holding $s \in \mathbb{F}_p$ and $s' \in \mathbb{F}_p^\times$, distributes n `Dist.Matrix` shares $\big([C]_1, \ldots, [C]_n\big)$ of s and n `Mult.split` shares $(m_1, \ldots, m_n)$ of s', respectively, to a set of n parties, $\{\mathcal{P}_j\}_{j=1}^n$. Let each party $\mathcal{P}_i$ locally compute the product $m_i[C]_i$. This way, each party obtains a `Dist.Matrix` share of $s \cdot s'$. Indeed,

$$\begin{aligned}\texttt{Reconstruct}\Big(p, \big(m_1[C]_1, \ldots, m_n[C]_n\big), p\Big) &= \sum_{i=1}^{n} \prod_{j=1}^{n} \Big(m_j[C]_j\Big)_i \\ = \sum_{i=1}^{n} \Big(\Big(\prod_{j=1}^{n} m_j\Big) \cdot \Big(\prod_{j=1}^{n} c_{ij}\Big)\Big) &= s' \cdot \sum_{i=1}^{n} \prod_{j=1}^{n} c_{ij} = s \cdot s'.\end{aligned} \tag{2}$$

Observe that, even if s is public, the procedure of homomorphically multiplying s by s' leaks no information regarding s', and if $s \neq 0$ then this procedure leaks no information regarding the result. Similarly, one can multiply the `Dist.Matrix` shares $\big([C]_1, \ldots, [C]_n\big)$ of s by `Mult.split` shares of $\mathbb{F}_p^\times$ elements $s'_1, ..., s'_d$ (for an arbitrary d) and obtain a result which is a `Dist.Matrix`-share of the product $s \cdot \prod_{i=1}^{d} s'_i$. □

Remark 3. Observe that, regarding the second part of the proof of Lemma 3, from (1) it follows that, given the $n^2 - 1$ entries c_{ij} of C with $i, j \neq n$, the entry c_{nn} is uniquely determined by x'. Hence, even if the coalition is given an additional information — the $n-1$ entries of the n'th share not on the main diagonal of C — the coalition remains perfectly oblivious to the actual value of x. This implies the following:

1. The `Dist.Matrix` sharing scheme enables transforming from holding multiplicative shares of x to holding additive shares of x in a single round of communication. Simply let each party $\mathcal{P}_i$ send the j'th entry of $[C]_i$ to $\mathcal{P}_j$ and have each party $\mathcal{P}_i$ compute the product of of the elements $c_{i,j}$ received

to obtain an additive share γ_i of x. Now the parties are ready to perform homomorphic additions with (practically infinite) `Add.split`-shared elements.

2. Transforming from additive shares of x to multiplicative shares of x can be done by letting each party $\mathcal{P}_i$ run $\texttt{Mult.Split}(p, \gamma_i, n, i)$ and send the j'th output entry to $\mathcal{P}_j$. Receiving these outputs from all other parties, each party obtains n elements of $\mathbb{F}_p$ which constitute a `Dist.Matrix` share of x.

4 Two-round MPC protocol with correlated randomness

In this section we make the first step approaching a proof of Theorem 1. We suggest schemes which enable MPC of arithmetic functions in two rounds of communication with perfect security using correlated randomness (CR). We begin with the *non-vanishing* case and then address the general *q-bounded* case. Assume that $\mathcal{P} = \{\mathcal{P}_j\}_{j=1}^n$ is a set of n parties, where each party $\mathcal{P}_j$ is holding a private input $s_j \in \mathbb{F}_p$. We assume that the parties are connected by point-to-point authenticated secure channels. Assume the parties wish to realize $f : \mathbb{F}_p^k \to \mathbb{F}_p$ over their inputs, where the minimal multivariate polynomial representation of f is

$$f(x_1, \ldots, x_k) = \sum_{l=(l_0,\ldots,l_k)\in\mathcal{L}} l_0 \cdot x_1^{l_1} \ldots x_k^{l_k}$$

and $\mathcal{L} = \{0, \ldots, p-1\}^{k+1}$. For $l \in \mathcal{L}$, the l'th monomial is $l_0 \cdot x_1^{l_1} \ldots x_k^{l_k}$.
The non-vanishing case.

In this case we assume that $s_i \neq 0$. The scheme Stages are as follows. We begin with the offline phase.

Preprocessing. For each monomial of f, each party $\mathcal{P}_j$ obtains a `Dist.Matrix` share $[C]_j^{(l)}$ of $1 \in \mathbb{F}_p$ ($l \in \mathcal{L}$). For ease of presentation, we occasionally omit the superscript (l).

Secret sharing. Each party $\mathcal{P}_i$ secret-shares s_i using `Mult.split` to obtain multiplicative shares $s_{i1}, \ldots, s_{in}$ of s_i and distributes s_{ij} to $\mathcal{P}_j$.

We regard the secret sharing stage as a part of the offline phase since we consider the case in which the secrets may be known long before the function, such as in the case of secret shared database. The Secret sharing phase can be squeezed into the first evaluation phase in a function-dependent way as explained below while maintaining two rounds of communication in the online phase. We continue with the online phase of the scheme.

Evaluation 1. For each monomial of f, each party $\mathcal{P}_j$ computes:

$$\alpha_j := \prod_{i=1}^{n} s_{ij}^{l_i} \cdot [C]_j.$$

Communication 1. For $1 \leq i, j \leq n$, $\mathcal{P}_j$ sends the i'th entry of α_j to $\mathcal{P}_i$.

Evaluation 2. Party $\mathcal{P}_i$ computes: $U_i = l_0 \cdot \prod_{j=1}^{n} (\alpha_j)_i$.

Communication 2. Each party $\mathcal{P}_i$ sends $\sum_l U_i^{(l)}$ to all other parties.

Output reconstruction. Each party computes

$$\sum_{i=1}^{n} \sum_{l} U_i^{(l)}.$$

Observe that,

$$\sum_{i=1}^{n} \sum_{l} U_i^{(l)} = \sum_{l} \sum_{i=1}^{n} U_i^{(l)} = \sum_{l} \sum_{i=1}^{n} l_0 \cdot \prod_{j=1}^{n} (\alpha_j)_i = \sum_{l} l_0 \cdot \sum_{i=1}^{n} \prod_{j=1}^{n} \left(s_{1j}^{l_1} \dots s_{nj}^{l_n} \cdot [C]_j \right)_i$$

$$= \sum_{l} l_0 \cdot \sum_{i=1}^{n} \left(s_1^{l_1} \dots s_n^{l_n} \cdot \prod_{j=1}^{n} c_{ij} \right) = \sum_{l} l_0 \cdot s_1^{l_1} \dots s_n^{l_n} \cdot \sum_{i=1}^{n} \gamma_i,$$

where γ_i denotes the product $c_{i1} \dots c_{in}$. Since C is a matrix-random-split of 1, we have $\sum_{i=1}^{n} \gamma_i = 1$ and hence,

$$\sum_{i=1}^{n} \sum_{l} U_i^{(l)} = f(s_1, \dots, s_n).$$

The q-bounded case. In the non-vanishing case, there is a limitation on the possible values that the s_j's may take. Namely, they cannot be zero. Can we avoid that limitation? We now consider a scenario in which some of the s_j's may be zero and suggest a way to overcome these limitations using a simple adjustment of the scheme assuming f is *q-bounded* for small enough q. That adjustment costs in a possible exponential blowup of the communication complexity. The general idea is to take the integer correspondents of the $\mathbb{F}_p$ inputs and run the same procedure as in the previous case while performing the computations in $\mathbb{F}_q$. Formally, for s_j let $a_j \in \{1, 2, \dots, p\} \subseteq \mathbb{N}$ denote the minimal positive integer such that $a_j \equiv s_j \pmod{p}$. For $1 \leq j \leq k$, let $\tilde{s}_j := a_j \pmod{q}$ and let $\tilde{s} = (\tilde{s}_1, \dots, \tilde{s}_k) \in \mathbb{F}_q^k$ denote the element of $\mathbb{F}_q^k$ corresponding to s in the q-world. Similarly, for $f : \mathbb{F}_p^k \to \mathbb{F}_p$, let $\tilde{f} : \mathbb{F}_q^k \to \mathbb{F}_q$ denote the function corresponding to f in the q-world. Now, all inputs are non-zero elements of $\mathbb{F}_q$ and the parties may invoke the same scheme of the non-vanishing case to evaluate $\tilde{f}$ over the inputs $\tilde{s}_j$.

Complexity analysis. We now analyze the communication complexity of the schemes suggested above. We begin with an important observation regarding representations of functions. Typically, SMPC protocols evaluate functions assuming that the function is described using an arithmetic circuit. One of the benefits of using arithmetic circuits is that they allow re-using intermediate values. The communication complexity of such schemes is analyzed in reference to the size and depth of the circuit. Under such considerations, addition gates are typically cheap (communication wise), and multiplication gates are costly.

In this work, we take a different approach and assume that the function is described by a multivariate polynomial and analyze the communication complexity of our schemes in reference to the *size of the polynomial* (the number of monomials in it). Our constructions induce free multiplications and a single round of communication in which all the additions are performed. Using a polynomial representation instead of an arithmetic circuit might induce some overhead. What is the exact relation between the number of monomials in the polynomial representation of an arbitrary function f and the size and depth of an arithmetic circuit which describes f? Giving a full answer to this question is beyond the scope of this paper. This question has roots in the algebraic analog of the problem $P \overset{?}{=} NP$, suggested by Valiant in [Val79].

Denote by k the number of monomials in f. In the preprocessing stage, each party obtains correlated randomness in the form of an independent `Dist.Matrix` share of $1 \in \mathbb{F}_p$ for each monomial of f. That share is an n-long vector of $\mathbb{F}_p$ elements. Hence, the space complexity induced by this stage is linear in n, linear in k, and linear in $\log p$.

In the secret sharing stage, each party obtains a single field element for each secret of every other party. Hence, the space complexity of this stage is linear in n and in $\log p$.

In the first communication stage, each party sends n messages for each monomial of f, where each message is single field element. Hence, we have a total of $n^2 \cdot k \cdot \log p$ bits of communication. Namely, the communication complexity induced by this stage is quadratic in n, linear in k and linear in $\log p$. Observe that the degree of f does not play a role here. In other words, the depth of the circuit neither affects the communication complexity of the scheme nor its round complexity.

In the second communication stage each party sends a single field element to every other party. That implies sending a total of n^2 messages, regardless of k.

Evaluation of boolean formulas. The scheme suggested above may be used to perform SMPC of boolean formulas by working in $\mathbb{F}_2$. A *True* boolean value is $1 \in \mathbb{F}_2$ and a *False* boolean value is $0 \in \mathbb{F}_2$. Boolean operations may be identified with field operations in the following way. The $\wedge$ operation is identified with $\mathbb{F}_2$ multiplication, the $\oplus$ operation with $\mathbb{F}_2$ addition, and the $\neg$ operation with adding 1 in $\mathbb{F}_2$. The $\vee$ operation of two literals is identified with $x + y + x \cdot y$, where x and y are the elements of $\mathbb{F}_2$ corresponding to the literals. Then, given a boolean formula C over boolean literals $b_1, \ldots, b_k \in \{True, False\}$, one can use the scheme suggested above for q-bounded functions to perform SMPC of C by taking $s_1, \ldots, s_k \in \mathbb{F}_2$, where the s_i's are the $\mathbb{F}_2$ correspondents of the b_i's. The boolean formula $C : \{True, False\}^k \rightarrow \{True, False\}$ will be taken as a polynomial function $\tilde{C} : \mathbb{F}_2^k \rightarrow \mathbb{F}_2$.

5 The client-servers model

Cloud services have become very popular in recent years. Companies like Amazon, Google, Microsoft, IBM, etc., are offering storage devices and computing engines to both private users and organizations. The usage of clouds for storage and computing has significant benefits in price, speed, and manageability. Nonetheless, it requires users to send their information to an untrusted party. In some cases, the information held by a user is confidential, and hence, the distribution of the information to untrusted parties cannot be considered. One solution to this problem may be a cryptographic scheme that enables a user to upload encrypted data to the cloud, perform computations in the cloud over the encrypted data and retrieve the encrypted version of the desired result. Such an encryption scheme enables the user to take advantage of the storage and computing services provided by the cloud without compromising the confidentiality of the data. Existing fully homomorphic encryption schemes suggest a centralized (rather than distributed) computationally secure solutions to the above mentioned problem [Gen09,SV10,VDGHV10,GHS12,GHS16,BP16,AGHL18]. Unfortunately, beyond being only computationally secure (rather than information-theoretically secure), they are currently too slow to be used in practice.
We now suggest a solution to this problem based on the scheme suggested in the previous section. Assume that a user has a private connection channel with n honest-but-curious servers, denoted $\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_n$. The scheme we suggest now enables a user to secret share $s_1, \ldots, s_k$ amongst the servers in a way that allows the user to evaluate $f(s_1, \ldots, s_k)$ using computing engines provided by the servers, where $f : \mathbb{F}_p^k \rightarrow \mathbb{F}_p$.

Secret sharing. The user secret-shares each s_i using `Mult.split` to obtain multiplicative shares $s_{i1}, \ldots, s_{in}$ of s_i and distributes the shares to the servers.
Preprocessing. For each monomial of f, the user uses the `Dist.Matrix` secret sharing scheme to generate and distribute $[C]_j^{(l)}$, shares of $1 \in \mathbb{F}_p$, $(l \in \mathcal{L})$ amongst the servers. Again, for ease of presentation, we occasionally omit the superscript (l).

Evaluation 1. For each monomial of f, each server $\mathcal{P}_j$ computes:

$$\alpha_j := \prod_{i=1}^{n} s_{ij}^{l_i} \cdot [C]_j.$$

Communication 1. For $1 \leq i, j \leq n$, $\mathcal{P}_j$ sends the i'th entry of α_j to $\mathcal{P}_i$.

Evaluation 2. $\mathcal{P}_i$ computes: $U_i = l_0 \cdot \prod_{j=1}^{n} (\alpha_j)_i$.

Sending shares of output to the user. Each server $\mathcal{P}_i$ sends $\sum_l U_i^{(l)}$ to the user.

Output reconstruction. The user computes

$$\sum_{i=1}^{n} \sum_{l} U_i^{(l)}.$$

6 Bootstrapping the Distributed Matrix

The schemes described above may be bootstrapped to enable the users generate the correlated randomness (CR) them selves. To jointly generate a matrix random split of $1 \in \mathbb{F}_p$ we use the following simple observations. By (1), given x, the last entry of the matrix is a function of the other entries. Let each party $\mathcal{P}_i$, $\leq i \leq n-1$ randomly choose $n-1$ uniformly random non-zero elements $c_{1,i}, \ldots, c_{i-1,i}, c_{i+1,i}, \ldots, c_{n,i}$ of $\mathbb{F}_p$ and another uniformly element c_{ii} of $\mathbb{F}_p$ (which may be zero). Let $\mathcal{P}_n$ choose $n-1$ uniformly random non-zero elements $c_{1,n}, \ldots, c_{n-1,n}$ of $\mathbb{F}_p$. Now, let $f : \mathbb{F}_p^{n^2-1} \to \mathbb{F}_p$ such that

$$f(c_{11}, \ldots, c_{n,n-1}) = \frac{1 - \sum_{i=1}^{n-1} \prod_{j=1}^{n} c_{ij}}{\prod_{j=1}^{n-1} c_{jn}}.$$

For $1 \leq j \leq n-1$ let $c'_{jn} = \frac{1}{c_{jn}}$. Replacing the c with the c' elements when necessary, f is an n-monomials polynomial. Let the parties engage in the client-servers scheme where $\mathcal{P}_n$ plays the role of the user and $\mathcal{P}_1, \ldots, \mathcal{P}_{n-1}$ play the role of the servers. At the end of the protocol the user obtains the n'th entry of his correlated randomness vector. This way the parties may generate the CR themselves.

7 Conclusions

In this paper, we have suggested schemes for perfectly secure multiparty computation of any arithmetic function with optimal round complexity, both in the

CR model and the client-servers model. These schemes were established based on four other SMPC schemes, presented in [BD18]. The schemes that appear in [BD18] use two sets of participants and require the elimination of the previous set whenever there is a switch between operations. That need for ongoing elimination of parties is solved here, where we present schemes for SMPC of arithmetic functions using one set of n parties for both operations with no need to eliminate parties. This solution costs in communication complexity. Particularly, in the schemes presented in [BD18], some of the procedures have communication complexity which is linear in the number of parties, and some of the procedures require sending a total of $n_1 \cdot n_2$ messages between the participants, where $n = n_1 + n_2$. In the schemes suggested here, the number of messages sent between participants is quadratic in the number of parties.

We have suggested schemes for performing computations in a finite field $\mathbb{F}_p$ in two different cases. The first case assumes that the secrets are non-zero elements in the field. In the second case, the secrets may vanish in $\mathbb{F}_p$, and we assume that f is q-bounded. We solve the second case using the first case by embedding $\mathbb{F}_p$ in $\mathbb{F}_q$ for large enough q. Such a q always exists. Nevertheless, in some cases, q must be so large that the resulting scheme is impractical.

To emphasize the importance of round-efficiency, we note that, while processing information becomes faster as technology improves, the time that it takes to transmit information between two distant places is strongly limited by the speed of light. One may consider a future need to perform SPMC over inputs held by parties which reside in distant places outside of earth.

Lastly, we believe that our new approach and techniques may be used to securely outsource computations in a reduced cost of communication complexity, and may be found to have further uses in many other scopes.

Acknowledgments. We thank Dani Berend for being involved during the entire research providing original ideas throughout, in particular suggesting to use polynomial representation instead of circuits.

References

[ABT18] Benny Applebaum, Zvika Brakerski, and Rotem Tsabary. Perfect secure computation in two rounds. In *Theory of Cryptography Conference*, pages 152–174. Springer, 2018.

[ACGJ18] Prabhanjan Ananth, Arka Rai Choudhuri, Aarushi Goel, and Abhishek Jain. Round-optimal secure multiparty computation with honest majority. In *Annual International Cryptology Conference*, pages 395–424. Springer, 2018.

[AGHL18] Adi Akavia, Craig Gentry, Shai Halevi, and Max Leibovich. Setup-free secure search on encrypted data: Faster and post-processing free. Technical report, Cryptology ePrint Archive Report, 2018.

[BD18] Dor Bitan and Shlomi Dolev. One-round secure multiparty computation of arithmetic streams and functions. In *International Symposium on Cyber Security Cryptography and Machine Learning*, pages 255–273. Springer, 2018.

[BIB89] Judit Bar-Ilan and Donald Beaver. Non-cryptographic fault-tolerant computing in constant number of rounds of interaction. In *Proceedings of the eighth annual ACM Symposium on Principles of distributed computing*, pages 201–209. ACM, 1989.

[BMR90] Donald Beaver, Silvio Micali, and Phillip Rogaway. The round complexity of secure protocols. In *Proceedings of the twenty-second annual ACM Symposium on Theory of Computing*, pages 503–513. ACM, 1990.

[BOGW88] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation. In *Proceedings of the twentieth annual ACM symposium on Theory of computing*, pages 1–10. ACM, 1988.

[BP16] Zvika Brakerski and Renen Perlman. Lattice-based fully dynamic multi-key fhe with short ciphertexts. In *Annual Cryptology Conference*, pages 190–213. Springer, 2016.

[CCD88] David Chaum, Claude Crépeau, and Ivan Damgard. Multiparty unconditionally secure protocols. In *Proceedings of the twentieth annual ACM symposium on Theory of computing*, pages 11–19. ACM, 1988.

[CLF12] Yaotsu Chang, Chong-Dao Lee, and Keqin Feng. Multivariate interpolation formula over finite fields and its applications in coding theory. *arXiv preprint arXiv:1209.1198*, 2012.

[DLN19] Ivan Damgård, Kasper Green Larsen, and Jesper Buus Nielsen. Communication lower bounds for statistically secure mpc, with or without preprocessing. *IACR Cryptology ePrint Archive*, 2019:220, 2019.

[DN03] Ivan Damgård and Jesper Buus Nielsen. Universally composable efficient multiparty computation from threshold homomorphic encryption. In *Annual International Cryptology Conference*, pages 247–264. Springer, 2003.

[Gen09] Craig Gentry. *A fully homomorphic encryption scheme.* Stanford University, 2009.

[GHS12] Craig Gentry, Shai Halevi, and Nigel P Smart. Fully homomorphic encryption with polylog overhead. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 465–482. Springer, 2012.

[GHS16] Craig B Gentry, Shai Halevi, and Nigel P Smart. Homomorphic evaluation including key switching, modulus switching, and dynamic noise management, 2016. US Patent 9,281,941.

[GIS18] Sanjam Garg, Yuval Ishai, and Akshayaram Srinivasan. Two-round mpc: information-theoretic and black-box. In *Theory of Cryptography Conference*, pages 123–151. Springer, 2018.

[GMW87] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game. In *Proceedings of the nineteenth annual ACM symposium on Theory of computing*, pages 218–229. ACM, 1987.

[KN06] Eyal Kushilevitz and Noam Nisan. *Communication Complexity.* Cambridge University Press, United Kingdom, 2006.

[PR18] Arpita Patra and Divya Ravi. On the exact round complexity of secure three-party computation. In *Annual International Cryptology Conference*, pages 425–458. Springer, 2018.

[Riv99] Ronald Rivest. Unconditionally secure commitment and oblivious transfer schemes using private channels and a trusted initializer. *Unpublished manuscript*, 1999.

[Sha79] Adi Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979.

[SV10] Nigel P Smart and Frederik Vercauteren. Fully homomorphic encryption with relatively small key and ciphertext sizes. In *International Workshop on Public Key Cryptography*, pages 420–443. Springer, 2010.

[Val79] Leslie G Valiant. Completeness classes in algebra. In *Proceedings of the eleventh annual ACM symposium on Theory of computing*, pages 249–261. ACM, 1979.

[VDGHV10] Marten Van Dijk, Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. Fully homomorphic encryption over the integers. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 24–43. Springer, 2010.

[Yao82] Andrew Chi-Chih Yao. Protocols for secure computations. In *FOCS*, volume 82, pages 160–164, 1982.