Optimal-Round Preprocessing-MPC via Polynomial Representation and Distributed Random Matrix
(Extended abstract)

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 preprocessing-MPC schemes of arithmetic functions with optimal round complexity, function-independent correlated randomness, and communication and space complexities that grow linearly with the size of the function. We extend our results to the client-server model and present 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. We further extend our results to handle Boolean circuits. All our schemes have perfect passive security against coalitions of up to $N-1$ parties. Our schemes are based on a novel secret sharing scheme, *Distributed Random Matrix* (DRM), which we present here. The DRM secret sharing scheme supports homomorphic multiplications, and, after a single round of communication, supports homomorphic additions.
Our approach deviates from standard conventions of MPC. First, we consider a representation of the function f as a multivariate polynomial (rather than an arithmetic circuit). Second, we divide the problem into two cases. We begin with solving the *non-vanishing* case, in which the inputs are non-zero elements of $\mathbb{F}_p$. In this case, our schemes have space complexity $\mathcal{O}(Nnk)$ and communication complexity $\mathcal{O}(N^2nk)$, where n is the size of the input, and k is the number of monomials of the function. Then, we present several solutions for the general case, in which some of the secrets can be zero. In these solutions, the space and communication complexities are either $\mathcal{O}(N^2n2^nk)$ and $\mathcal{O}(N^3n2^nk)$, or $\mathcal{O}(NnK)$ and $\mathcal{O}(N^2nK)$, respectively, where K is the size of a modified version of f. K is bounded by the square of the maximal possible size of k.

Keywords: MPC with preprocessing, Correlated randomness, Optimal round complexity, Homomorphic secret sharing, Perfect security

1 Introduction

Secure multiparty computation (MPC) is an extensively studied field in cryptography, which discusses the following problem. $\mathcal{P}_1, \ldots, \mathcal{P}_N$ are N parties, $s_1, \ldots, s_N$ are their corresponding secret inputs, and f is some function of N inputs. The parties wish to find $f(s_1, \ldots, s_n)$, while not revealing to each other any information regarding their secret inputs (except for what may be deduced from the output). A vast number of papers were written on that topic in the past four decades based on various assumptions [Yao82,GMW87,BOGW88,CCD88,DI05,IKM$^+$13,DNNR17,ABT18].

The scope of this work is perfectly secure schemes. We assume that the parties are honest-but-curious. In their seminal work, Ben-Or et al. [BOGW88] showed that, in the plain model, every function can be efficiently computed with perfect passive security by N parties if and only if an honest majority is assumed. Nevertheless, perfect passive security against dishonest majority is achievable in *the preprocessing model*. That model, first suggested in [Bea97], assumes that the parties engage in an offline preprocessing phase before the secret inputs are known, at which they obtain (preferably function-independent) correlated randomness (CR), which is then consumed at the online phase of the protocol.

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]. Substantial effort has been spent on finding the minimal number of rounds required for perfectly secure MPC, both theoretically and practically. Bar-Ilan and Beaver [BIB89] were the first to suggest MPC schemes with a constant number of rounds, followed by further works that attempt to lower bound that constant number. Theoretically, two rounds are now known to be optimal for MPC — in the plain or preprocessing model [PR18,DLN19]. Recent works by [ABT18,GIS18,ACGJ18] present plain-model protocols which enable MPC in two rounds and which have perfect passive security against honest majority. Ishai et al. suggested in [IKM$^+$13] preprocessing-model protocols (hereafter, P-MPC protocols) with two rounds and perfect passive security against dishonest majority.

Though theoretically, the problem of finding perfectly secure optimal-round optimal-threshold MPC schemes was resolved both in the plain ([ABT18]) and preprocessing ([IKM$^+$13]) models, practically, the race is far from being over since there is yet a lot to improve. Particularly, in the preprocessing model, the space complexity of known solutions is ex-

ponential in the size of the input and N. Reducing the space complexity of these schemes (even for specific cases) costs in increasing round complexity (or other compromises). To the best of our knowledge, there is no known P-MPC scheme which enables reducing the space complexity in a non-trivial case while maintaining two-rounds of communication, perfect security against dishonest majority, and f-independent CR.

Our main results. We construct the first P-MPC schemes for arithmetic functions that have communication and space complexity linear in the size of the function, two-rounds of communication, perfect security against dishonest majority, and f-independent CR. Our *Distributed Random Matrix (DRM) two-round P-MPC schemes* presented here assume that f is a polynomial with k monomials,[3] and are highly efficient in the *non-vanishing case* (NV), in which we assume that all secret inputs are non-zero elements of $\mathbb{F}_p$. Our NV schemes achieve the following properties.

- Perfect passive security against coalitions of up to $N-1$ parties.
- Space complexity $\mathcal{O}(kNn)$, where $n = \lceil \log p \rceil$ is the size of the input.
- Total communication complexity $\mathcal{O}(kN^2n)$.
- Function-independent correlated randomness.
- Optimal round complexity. I.e., two rounds of communication.

We adjust the NV schemes to achieve solutions for the general case, in which some of the inputs may be zero. Our schemes are based on the *DRM secret sharing scheme*, our new homomorphic secret sharing scheme presented here. We also extend our results to the client-server model and to Boolean circuits.

Remark 1. **Polynomials instead circuits and non-zero inputs.** Most MPC schemes assume that f is given as a circuit, and their communication and space complexities are analyzed with respect to the size s and depth d of the circuit. Our schemes assume that f is a polynomial with k monomials and evaluate each monomial independently. On first sight, this choice may be unclear since it may induce some computational overhead comparing to circuits, which have the benefit of enabling re-use of computed mid-values. In order to (asymptotically) compare the performances of our schemes to those of standard schemes, one should write k in terms of s and d. Finding the relation between the number of monomials of a general function and the size and depth of a circuit which computes the same function has roots in the algebraic analog of the $P \stackrel{?}{=} NP$ problem (suggested by Valiant in [Val79]) and is beyond the scope of this paper.

[3] Over finite fields, every function has a polynomial representation.

However, we show here that the polynomials approach may resolve us from being concerned with d, which is one of the main complexity bottlenecks in MPC. Second, it is common that MPC schemes assume specific conditions over the functions (e.g., NC^1), but conditions over the inputs are hardly ever discussed. However, we address the case of non-zero inputs and achieve a solution for this case with remarkable performances.

Related work. Rivest [Riv99] presented perfectly secure two-party commitment and oblivious transfer (OT) schemes which use CR. Since OT is known to be complete for MPC [Kil88], combining these results, one obtains a statistically secure MPC scheme. A similar approach was used in [IPS08] to construct a statistically secure P-MPC scheme, which realizes any circuit of size s and depth d with communication complexity linear in s (and polynomial in other parameters), and round complexity $\mathcal{O}(d)$.

Ishai et al. suggested in [IKM$^+$13] P-MPC schemes which are based on One-Time Truth Tables (OTTT), require two-rounds of communication, use f-dependent CR, and have perfect passive security against dishonest majority. The communication complexity of their schemes grows linearly with the input size and the number of parties and is independent of the function. The space complexity of their schemes, however, is exponential in the number of parties and the size of the input, *regardless of* f. This makes their schemes impractical for large inputs or a large number of parties, even when considering simple functions. They also suggested schemes with reduced space complexity, but these schemes require $\mathcal{O}(d)$ rounds.

Damgård et al. [DZ13] presented MiniMac, a P-MPC protocol for *well-formed* Boolean circuits. Their schemes have statistical active security against dishonest majority, negligible error probability, constant computational overhead, and communication complexity linear in s and N. Similar performances for general circuits are achieved by the TinyTable protocol, suggested in [DNNR17]. Both the MiniMac and TinyTable protocols have round complexity $\mathcal{O}(d)$. Recent work by Couteau [Cou19] presents P-MPC schemes for *layered Boolean circuits*. Their schemes have perfect passive security against dishonest majority and communication complexity sublinear in s. However, the round complexity of their schemes is $\mathcal{O}(d/\log\log s)$.

To the best of our knowledge, there is no known P-MPC scheme that enables evaluation of any function in two rounds of communication with perfect passive security against dishonest majority and communication and space complexities that grow linearly with the size of the function. Several versions of our new DRM two-round MPC schemes are the first to achieve all these attributes.

Paper organization. In Section 2, we present the DRM secret sharing scheme and discuss its homomorphic properties. Our DRM two-round MPC schemes are presented in Section 3. In Section 4, we discuss the evaluation of Boolean circuits and the client-server model. Conclusions appear in Section 5. Preliminaries and some extra material appear in Appendix.

2 Distributed Random Matrix Secret Sharing Scheme

In this section, we describe the basic tool of this work, the *Distributed Random Matrix* procedure, `DRM`. `DRM` employs two other procedures — `Mult.split` (which is also used in our schemes to secret-share the inputs) and `Add.split`. We begin with `Mult.split`.

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.

`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 2. Joint zeroness of s and m_i. 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. `Mult.split` *is a perfectly-secure multiplicatively-homomorphic secret sharing scheme for $\mathbb{F}_p^\times$ elements with threshold $N - 1$.*

Additive secret sharing procedure. Similarly, given a prime p, an element $s \in \mathbb{F}_p$, and $N \in \mathbb{N}$, `Add.split` returns $(\gamma_1, \ldots, \gamma_N)$, a sequence of additive secret shares of s, as follows: $\gamma_1, \ldots, \gamma_{N-1}$ are uniformly randomly chosen from $\mathbb{F}_p$, and γ_N is determined to satisfy $\sum_{i=1}^{N} \gamma_i = s$. While `Mult.split` takes an input $i \in [N]$, `Add.split` takes no such input.

Lemma 2. `Add.split` *is a perfectly-secure additively-homomorphic secret sharing scheme for $\mathbb{F}_p$ elements with threshold $N - 1$.*
The proofs of Lemma 1 and 2 appear in the full version of the paper.

Distributed Random Matrix (DRM) secret sharing procedure. We now define the procedure `DRM`. Given a prime p, an element $x \in \mathbb{F}_p$ and a natural number $N \in \mathbb{N}$, the procedure `DRM` outputs (the columns of) a matrix C, a *matrix-random-split of* x.

```
DRM(p, s, N):                                  # {p is prime, s ∈ F_p, N ∈ N}
    (γ_1, ..., γ_N) ← Add.split(p, s, N);
    For 1 ≤ i ≤ N:
        (c_i1, c_i2, ..., c_iN) ← Mult.split(p, γ_i, N, i);
    C ← (c_ij)_{i,j∈[N]} ∈ M_N(F_p);
    return ([C]_1, ..., [C]_N)
```

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

Remark 3. Since the i'th row of C is the output of `Mult.split` on inputs p, γ_i, N and i, from Remark 2 it follows that the matrix C may contain zeroes only on its main diagonal, if any. Namely, $c_{ij} = 0 \implies i = j$.

One may readily verify that, $\sum_{i=1}^{N} \prod_{j=1}^{N} c_{ij} = x$. Hence, reconstruction of a `DRM`-secret-shared element x from N shares may be performed by multiplying all the elements in each row of C and summing the products.

We claim that the `DRM` secret sharing scheme supports an arbitrary number of homomorphic multiplications by `Mult.split`-secret-shared non-zero elements, and, after a single round of communication, it supports an arbitrary number of homomorphic additions with `Add.split` secret-shared elements. To make this claim precise, we define the procedure `M2A`.

`M2A`

The procedure is invoked by N parties, $\mathcal{P}_1, \ldots, \mathcal{P}_N$.
Let $s \in \mathbb{F}_p$ and $([C]_1, \ldots, [C]_N) = \texttt{DRM}(p, s, N)$.
Each party $\mathcal{P}_j$ is holding $[C]_j$.

Communication round:

For $1 \leq i, j \leq N$: $\mathcal{P}_j$ sends the i't entry of $[C]_j$ to $\mathcal{P}_i$.

Output computation:

For $1 \leq i \leq N$: $\mathcal{P}_i$ computes $\gamma_i = \prod_{j=1}^{N} c_{ij}$.

`M2A`. *Transforming from supporting multiplications to supporting additions.*

One may readily verify that, following `M2A`, each party obtains an additive share of s. Namely, $\sum_{i=1}^{N} \gamma_i = s$.

Theorem 1. *The procedure* `DRM` *is an N-party secret sharing scheme for $\mathbb{F}_p$ elements which has perfect passive security and threshold $N-1$.* `DRM` *supports homomorphic multiplications by* `Mult.split`*-secret-shared non-zero elements. Have N parties hold* `DRM`*-shares of $s \in \mathbb{F}_p$, executing* `M2A`*, the parties obtain additive shares of s. These additive shares of s enable homomorphic additions with* `Add.split`*-secret-shared elements.*

The proof of Theorem 1 is omitted from this extended abstract version.

3 The DRM Two-Round MPC Schemes

In this section, we present two-round perfectly secure P-MPC schemes for arithmetic functions, based on the DRM secret sharing scheme. We assume that $\mathcal{P} = \{\mathcal{P}_j\}_{j=1}^N$ is a set of $N \geq 2$ honest-but-curious parties which are connected via point-to-point authenticated secure channels. For ease of presentation, we assume that each party $\mathcal{P}_j$ holds a single input $s_j \in \mathbb{F}_p$. In general, each party may hold an arbitrary number of secrets. Let $s = (s_1, \ldots, s_N)$. The function to be evaluated is $f : \mathbb{F}_p^N \to \mathbb{F}_p$, and its *minimal multivariate polynomial representation* (see Preliminaries) is:

$$f(x_1, \ldots, x_N) = \sum_{l=(l_1,\ldots,l_N)\in\mathcal{L}} a_l \cdot x_1^{l_1} \ldots x_N^{l_N},$$

where $\mathcal{L} = \{0, \ldots, p-1\}^N$ and $a_l \in \mathbb{F}_p$. For $l \in \mathcal{L}$, let $A_l = x_1^{l_1} \ldots x_N^{l_N}$. The l'th monomial of f is $a_l A_l$. The size of the function (i.e., the number of monomials with $a_l \neq 0$) is k, and the size of the input is $n = \lceil \log p \rceil$.

We begin with the *database-oriented* (DBO) versions of the DRM two-round MPC scheme, and then present the *one-time secrets* (OTS) versions. The DBO versions involve a secret sharing stage as a part of the preprocessing phase. This stage creates a virtual database shared among the parties and enables any function to be evaluated over that database. The result of the evaluation may be added to the database for future use. The OTS versions of the scheme involve no secret sharing stage. Each of the DBO and OTS versions includes four schemes. The first scheme solves the *non-vanishing* (NV) case, in which the secrets are assumed to be non-zero. The second solves the general case, where some of the inputs may be zero, by embedding $\mathbb{F}_p$ in a larger field, $\mathbb{F}_q$. The larger field is taken such that f is *q-bounded* (qB, see Preliminaries). The last two schemes are the *input-splitting* (IS) schemes, which solve the general case for $p \neq 2$ and $p = 2$, by splitting each secret to a sum of two elements. Overall, we obtain eight DRM two-round MPC schemes: DBO-NV, DBO-qB, DBO-IS, DBO-IS$_2$, OTS-NV, OTS-qB, OTS-IS, and OTS-IS$_2$. We begin with DBO-NV.

The DBO-NV scheme. DBO-NV may be invoked by N parties to find $f(s)$ assuming that all inputs are non-zero.

DBO-NV

Preprocessing phase.

Correlated randomness. For each non-zero monomial $a_l A_l$ of f, each party $\mathcal{P}_j$ obtains a `DRM`-share $[C^{(l)}]_j$ of $1 \in \mathbb{F}_p$. Each $C^{(l)}$ is a matrix-random-split of $1 \in \mathbb{F}_p$.

Secret sharing. Each party $\mathcal{P}_i$ secret-shares s_i using `Mult.split`. The shares $s_{i1}, \ldots, s_{iN}$ of s_i are distributed such that $\mathcal{P}_j$ receives s_{ij}.

Online phase.

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

$$\alpha_j^{(l)} = \prod_{i=1}^{N} s_{ij}^{l_i} \cdot [C^{(l)}]_j.$$

Com. 1. For $i, j \in [N]$, $\mathcal{P}_j$ sends the i'th entry of each $\alpha_j^{(l)}$ to $\mathcal{P}_i$.

Eval. 2. For each monomial $a_l A_l$ of f, each party $\mathcal{P}_i$ computes:

$$U_i^{(l)} = a_l \prod_{j=1}^{N} \left(\alpha_j^{(l)}\right)_i.$$

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

Output reconstruction. Each party computes $\sum_{i=1}^{N} y_i$.

Theorem 2. *DBO-NV is a two-round N-party P-MPC scheme for arithmetic functions over non-zero inputs which has perfect correctness, perfect passive security, threshold $N-1$, communication complexity $\mathcal{O}(N^2 nk)$, space complexity $\mathcal{O}(Nnk)$, and f-independent CR.*
The proof of Theorem 2 appears in Appendix B.

The DBO-qB scheme. DBO-qB solves the general case by embedding $\mathbb{F}_p$ in a larger field, $\mathbb{F}_q$, and using DBO-NV as a subroutine. The larger field $\mathbb{F}_q$ is chosen to satisfy the condition that f is q-*bounded* (see Preliminaries). The embedding is performed as follows. For $s_j \in \mathbb{F}_p$, let σ_j denote the minimal positive integer such that $\sigma_j \equiv s_j \pmod{p}$. Let $\tilde{s}_j \equiv \sigma_j \pmod{q}$ the $\mathbb{F}_q$ correspondent of s_j in the q world. Let $\tilde{s} = (\tilde{s}_1, \ldots, \tilde{s}_N)$. Now, let $\tilde{f} : \mathbb{F}_q^N \to \mathbb{F}_q$ denote the function corresponding to f in the q-world. That is, $\tilde{f}$ is obtained from f by replacing the leading coefficients of the (non-zero) monomials with their q-world correspondents. DBO-qB may be invoked by the parties to find $f(s)$.

DBO-qB

Calling DBO-NV. Use DBO-NV to find $\tilde{y} = \tilde{f}(\tilde{s}) \in \mathbb{F}_q$.

Computing p-world output. Let σ denote the minimal positive integer such that $\sigma \equiv \tilde{y} \pmod{q}$, and let $y \equiv \sigma \pmod{p}$. Output y.

Theorem 3. *DBO-qB is a two-round N-party P-MPC scheme for arithmetic functions which has perfect correctness, perfect passive security, threshold $N-1$, communication complexity $\mathcal{O}(kN^3n2^n)$, space complexity $\mathcal{O}(kN^2n2^n)$, and f-independent CR.*

The proof of Theorem 3 appears in Appendix C. DBO-qB solves the general case by replacing $\mathbb{F}_p$ elements with $\mathbb{F}_q$ elements, hence the factor 2^n. DBO-IS and DBO-IS$_2$, which we now present, avoid the 2^n factor. Instead, these schemes replace f with a K-monomials version of it.

The DBO-IS scheme. DBO-IS solves the general case by splitting each input to a sum of two non-zero elements, and replacing f with *the split-inputs version* of f (see Preliminaries). Let $\varphi : \mathbb{F}_p^{2N} \to \mathbb{F}_p$ the split-inputs version of f. If $p \neq 2$, DBO-IS may be invoked by the parties to find $f(s)$.

DBO-IS

Each party $\mathcal{P}_j$ arbitrarily picks $\alpha_j, \beta_j \in \mathbb{F}_p^{\times}$ such that $s_j = \alpha_j + \beta_j$. Use DBO-NV to find $y = \varphi(\alpha_1, \ldots, \alpha_N, \beta_1, \ldots, \beta_N)$. Output y.

Theorem 4. *DBO-IS is a two-round N-party P-MPC scheme for $\mathbb{F}_p$ functions ($p \neq 2$) which has perfect correctness, perfect passive security, threshold $N-1$, f-independent CR, communication complexity $\mathcal{O}(N^2nK)$, and space complexity $\mathcal{O}(NnK)$, where K is the number of monomials of the split-inputs version of f.*

The proof of Theorem 4 appears in the full version of the paper. We note that, since f is a polynomial of N variables, $k \leq p^N$, and since φ is a polynomial of $2N$ variables, $K \leq p^{2N} = (p^N)^2$.

The DBO-IS$_2$ scheme. In $\mathbb{F}_2$, 1 cannot be written as a sum of two non-zero elements. This is the reason for the requirement $p \neq 2$ in Theorem 4. DBO-IS$_2$ solves the case $p = 2$ by embedding $\mathbb{F}_2$ in $\mathbb{F}_3$ and using DBO-IS as a subroutine. The embedding is performed as follows. The elements $0, 1 \in \mathbb{F}_2$ are identified with $0, 1 \in \mathbb{F}_3$. For $s_j \in \mathbb{F}_2$ let $\overline{s_j}$ denote the $\mathbb{F}_3$ correspondent of s_j. $\mathbb{F}_2$ operations are identified with $\mathbb{F}_3$ operations as follows. $\mathbb{F}_2$-multiplication is identified with $\mathbb{F}_3$-multiplication, and $\mathbb{F}_2$-addition is identified with $Add : \mathbb{F}_3^2 \to \mathbb{F}_3$, $Add(x, y) = x + y + xy$. Let

$\overline{f} : \mathbb{F}_3^N \to \mathbb{F}_3$ denote the $\mathbb{F}_3$ correspondent of f. That is, $\overline{f}$ is obtained from f by replacing the $\mathbb{F}_2$-operations '$\cdot$' and '$+$' of f with the $\mathbb{F}_3$-operations '$\cdot$' and '*Add*'. The parties may invoke DBO-IS$_2$ to find $f(s)$.

DBO-IS$_2$

Use DBO-IS to find $y = \overline{f}(\overline{s_1}, \ldots, \overline{s_N})$. Output y.

Theorem 5. *DBO-IS*$_2$ *is a two-round N-party P-MPC scheme for arithmetic functions over $\mathbb{F}_2$ which has perfect correctness, perfect passive security, threshold $N-1$, f-independent CR, communication complexity $\mathcal{O}(N^2nK)$, and space complexity $\mathcal{O}(NnK)$, where K is the number of monomials of the split-input version of the $\mathbb{F}_3$ correspondent of f.*
The proof of Theorem 5 appears in the full version of the paper.

The OTS versions of the DRM two-round MPC schemes. We give a general description of the OTS versions of the DRM two-round MPC schemes. A full description of the schemes appears in the full version of the paper. The OTS-NV scheme is obtained from DBO-NV by omitting the secret sharing stage, and replacing Eval. 1 of DBO-NV with:
Eval. 1. For each monomial of f, each party $\mathcal{P}_j$ computes $\alpha_j^{(l)} = s_j^{l_j}[C^{(l)}]_j$.

The schemes OTS-qB, OTS-IS and OTS-IS$_2$ are constructed by using OTS-NV as a subroutine, similarly to the way DBO-qB, DBO-IS and DBO-IS$_2$ are constructed (by using DBO-NV as a subroutine), and have similar attributes and performances (see Appendix D).

4 Extensions

The client-server model. Assume $N \geq 2$ honest-but-curious servers and $m \geq 1$ users, with a fully connected network of servers, and a connection channel between each user to each server. Let each of the users and servers hold an arbitrary number of secret inputs in $\mathbb{F}_p$. The *DRM single-round client-server schemes* enable the users to securely outsource the storage of their private inputs to the servers and have the servers evaluate any function over the entire collection of inputs in a single round of communication. The client-server-NV scheme solves the non-vanishing case as follows. At the preprocessing phase, each input-holder `Mult.split`-shares her inputs among all servers, and the users provide the servers with N `DRM`-shares of $1 \in \mathbb{F}_p$ for each monomial of the function. The online phase is invoked by the servers and is similar to that of DBO-NV, except for *Com. 2*, in which the servers send the additive shares of the result to

the users (and not to one another). The general case is solved using the qB or IS approach. Our client-server schemes are perfectly secure against coalitions of up to $N-1$ honest-but-curious servers. Full description of them appears in the full version of the paper. In Appendix E, we discuss a method for harnessing the client-server scheme for self-generation of CR.

Precision-Complexity tradeoff. It is well known that circuits may be approximated using polynomials [DOdS15]. These approximation polynomials may be generated in such a way that they disagree with the original function on a small set of inputs, and have a relatively small size. Approximation polynomials suggest a tradeoff in our schemes between precision and complexity. Assume N parties wish to evaluate f over their private inputs, where the size of the minimal multivariate polynomial representation of f is k. The parties may replace f with a polynomial g which approximates f, where g has k' monomials, $k' < k$. Evaluating g on their inputs, the parties obtain an approximated result in reduced communication and space costs. Note that, in some scenarios, approximate results are used to preserve data privacy (e.g., in differential privacy), and hence, the reduction in precision may be desired due to other considerations.

Evaluation of Boolean formulas. Our schemes may be used to perform MPC 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 $x+y+xy$, where x and y are the elements of $\mathbb{F}_2$ corresponding to the literals. Then, given a Boolean formula φ over Boolean literals $b_1, \ldots, b_M \in \{True, False\}$, one can use DBO-$q$B to perform MPC of φ by taking the $\mathbb{F}_2$ correspondents $s_1, \ldots, s_M \in \mathbb{F}_2$ of $b_1, \ldots, b_M$. Evaluation of the Boolean formula $\varphi : \{True, False\}^M \to \{True, False\}$ will be implemented using the polynomial $\tilde{\varphi} : \mathbb{F}_2^M \to \mathbb{F}_2$, obtained by replacing Boolean operations and literals with their $\mathbb{F}_2$ correspondents.

For example, consider the following scenario. Three millionaires wish to find out which of them is the wealthiest, while not revealing to each other any further information regarding the number of millions they possess. Denote the three millionaires by $\mathcal{P}_1, \mathcal{P}_2, \mathcal{P}_3$, and the number of millions they possess by s_1, s_2, s_3, respectively. For simplicity, assume each s_i is a positive integers between 1 and 10 (other cases may be solved similarly). Let $f : \mathbb{F}_{11}^3 \to \mathbb{F}_{11}$ the function that returns (a) 0, if $x_1 = x_2 = x_3$; (b) i, if x_i is larger (as an integer) than the two other variables; (c) $i+j+1$,

if $x_i = x_j$ (where $i \neq j$) and x_i, x_j are both larger (as integers) than the other variable. The minimal multivariate polynomial representation of f is given in Appendix F. To find who of them is the wealthiest, the parties may invoke DBO-NV to evaluate $f(x_1, x_2, x_3)$ at (s_1, s_2, s_3).

5 Conclusions

In this paper, we have suggested P-MPC schemes that support evaluation of any function with perfect passive security against dishonest majority, optimal round complexity, and space complexity that grows linearly with the size of the function. We began with the construction of an $N-1$-out-of-N perfectly secure secret sharing scheme which supports homomorphic multiplications with non-zero elements of a finite field $\mathbb{F}_p$. We showed how the parties may efficiently generate additive shares of the secret from multiplicative shares of it in a single round of communication, thus enabling homomorphic additions with elements of $\mathbb{F}_p$. This secret sharing scheme was then used to construct a perfectly secure two-round P-MPC scheme for arithmetic functions, assuming the inputs are non-zero elements of $\mathbb{F}_p$. We suggested solutions for the general case based on two different approaches. The first approach looks on the inputs as elements of a larger field, $\mathbb{F}_q$, and replace zero inputs with $p \in \mathbb{F}_q$. The second approach splits each secret to a sum of two non-zero elements. We extended our schemes to the client-server model. Solutions for the case of evaluating Boolean formulas were also suggested here. The case of active security was not discussed in this work, and is to be adequately addressed in future work.

The round complexity of our schemes is optimal. 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. Assume a need to perform MPC over inputs held by parties which reside in distant places, perhaps outside of earth, and let T denote the time it takes to process the computations for evaluation of f using our schemes. If the distance between parties is such that, sending messages between parties takes more time than T, then our optimal-round schemes will outperform any scheme with non-optimal round complexity.

Lastly, we believe that our new methods and construction suggest an alternative approach to MPC, which may inspire other works in this field and may be found to have further implications in other fields as well.

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*. Springer, 2018.

[Bea97] Donald Beaver. Commodity-based cryptography. In *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, pages 446–455. ACM, 1997.

[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.

[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.

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

[Cou19] Geoffroy Couteau. A note on the communication complexity of multiparty computation in the correlated randomness model. In *Advances in Cryptology - EUROCRYPT 2019 - 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Darmstadt, Germany, 2019, Proceedings, Part II*, pages 473–503, 2019.

[DI05] Ivan Damgård and Yuval Ishai. Constant-round multiparty computation using a black-box pseudorandom generator. In *Annual International Cryptology Conference*, pages 378–394. Springer, 2005.

[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.

[DNNR17] Ivan Damgård, Jesper Buus Nielsen, Michael Nielsen, and Samuel Ranellucci. The tinytable protocol for 2-party secure computation, or: Gate-scrambling revisited. In *Advances in Cryptology - CRYPTO 2017 - 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, 2017, Proceedings, Part I*, pages 167–187, 2017.

[DOdS15] Hélio M De Oliveira and RM de Souza. Introducing an analysis in finite fields. *arXiv preprint arXiv:1501.07502*, 2015.

[DZ13] Ivan Damgård and Sarah Zakarias. Constant-overhead secure computation of boolean circuits using preprocessing. In *Proceedings of Theory of Cryptography 2013 - The 10th Theory of Cryptography Conference TCC.*, pages 621–641, 2013.

[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.

[IKM+13] Yuval Ishai, Eyal Kushilevitz, Sigurd Meldgaard, Claudio Orlandi, and Anat Paskin-Cherniavsky. On the power of correlated randomness in secure computation. In *Theory of Cryptography Conference*. Springer, 2013.

[IPS08] Yuval Ishai, Manoj Prabhakaran, and Amit Sahai. Founding cryptography on oblivious transfer - efficiently. In *Advances in Cryptology - CRYPTO 2008, 28th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2008. Proceedings*, pages 572–591, 2008.

[Kil88] Joe Kilian. Founding crytpography on oblivious transfer. In *Proceedings of the twentieth annual ACM symposium on Theory of computing*, 1988.

[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.

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

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

A Preliminaries

We recall some linear algebra and MPC notations and definitions and define several terms and concepts used 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 $\mathbb{F}_p$, and $\mathbb{F}_p^\times$ to denote the multiplicative group of $\mathbb{F}_p$. $(\mathbb{F}_p^\times)^k$ is the set of k-tuples over $\mathbb{F}_p^\times$, with the operations '$+$' and '$\cdot$' for $\mathbb{F}_p$ entry-wise addition and scalar multiplication, respectively. The notation '$*$' stands for entrywise multiplication, and '$||$' for concatenation. 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. For $n \in \mathbb{N}$, $[n] = \{1, \ldots, n\}$. 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 $x \xleftarrow{R} A$ the process of assigning to the variable x a uniformly random element of 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. 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 domain $\mathcal{R}_j$. Assume that the parties wish to find $f(s_1, \ldots, s_n)$, where $f : \mathcal{R}_1 \times \cdots \times \mathcal{R}_N \to \mathcal{R}$, while not revealing to each other any information regarding their secret inputs, except for what may be deduced from $f(s_1, \ldots, s_N)$. In an ideal world, the parties could have found 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 that they could not have learned by following the ideal world solution. We also consider the case in which a subset $\mathcal{T} \subseteq \mathcal{P}$ of the parties 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. π leaks no more information than Ted if all the information obtained from π may be computed from the information received from Ted only. To formalize that, 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 $\mathcal{P}_j$ makes. Recall that all parties are honest-but-curious, and hence, they follow the instructions of π. That leads us to

Definition 1. ***Perfect correctness and perfect passive security of*** π**.** *Let f a function, $t \in \mathbb{N}$, and π be an N-party protocol for computing f. We say that π realizes f with perfect correctness and perfect passive security with threshold t if (a) by executing π, all parties learn $y = f(s_1, \ldots, s_N)$, and (b) for every adversarial coalition of honest-but-curious parties $\mathcal{T} \subseteq \mathcal{P}$ with $|\mathcal{T}| \leq t$ there exists a simulator — a probabilistic algorithm* `Sim` *— which on inputs y and $\{s_j\}_{\mathcal{P}_j \subseteq \mathcal{T}}$, its output is identically distributed to $view_T = \{view_j\}_{\mathcal{P}_j \in \mathcal{T}}$.*

Correlated Randomness (CR). In the preprocessing model, we assume that π may include an offline preprocessing phase in which the parties obtain correlated randomness. That is, each party $\mathcal{P}_j$ obtains a secret random binary string string $R_j \in \{0,1\}^r$ such that $R = R_1 || \ldots || R_N$ is a $\mathcal{D}$-distributed element of $\{0,1\}^{rN}$, where $\mathcal{D}$ is some predefined public distribution over $\{0,1\}^{rN}$ independent of the inputs. If $\mathcal{D}$ is also independent of f, we say that π uses f-independent CR.

We consider two ways of obtaining CR. The first involves a trusted initializer which provides the parties with the random strings R_j. We stress that there is a fundamental difference between the trusted entity Ted mentioned earlier in the ideal world solution to the trusted initializer that we consider now. While Ted receives the actual secret inputs from the parties and performs the computation in their behalf, the trusted initializer remains utterly oblivious to the secret inputs as the CR phase takes place before the secret inputs are known. Considering the presence of a trusted initializer seems quite natural because whenever we engage in digital communication, we use the services of a trusted server, which provides authentication for the communicating parties. That server might as well provide the parties with CR. The second way of obtaining CR requires the parties running some offline protocol to generate and store correlated random strings.

Minimal multivariate polynomial representation. Any function $f : \mathbb{F}_p^N \to \mathbb{F}_p$ can be represented as a multivariate polynomial. This representation may be obtained, for example, by solving a system of linear equations as in the example in Appendix F. The fact that $x^p \equiv x \pmod{p}$ implies that for a given $f : \mathbb{F}_p^N \to \mathbb{F}_p$ there are infinitely many polynomial representations of it. 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 for the sake of readability. Whenever a function $f : \mathbb{F}_p^N \to \mathbb{F}_p$ is discussed, we assume that f is given with its minimal multivariate polynomial representation. We write

$$f(x_1, \ldots, x_N) = \sum_{l=(l_1,\ldots,l_N)\in\mathcal{L}} a_l \cdot x_1^{l_1} \ldots x_N^{l_N}, \tag{1}$$

where $\mathcal{L} = \{0, \ldots, p-1\}^N$ and $a_l \in \mathbb{F}_p$. For $l = (l_1, \ldots, l_N) \in \mathcal{L}$, we denote $x_1^{l_1} \ldots x_N^{l_N}$ by A_l. We refer to $a_l A_l$ as *the l'th monomial of f*. If a_l is zero, then $a_l A_l$ is a *zero monomial*. Otherwise, $a_l A_l$ is a *non-zero monomial*.

q-Bounded functions. Let $s = (s_1, \ldots, s_N) \in \mathbb{F}_p^N$. One can compute $f(s)$ by performing operations in $\mathbb{F}_p$ 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 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 $f(s)_{\mathbb{N}} \equiv f(s) \pmod{p}$. If q is a prime number such that for every $s \in \mathbb{F}_p^N$, computation of $f(s)$ over the integers yields an integer result, $f(s)_{\mathbb{N}}$, which is smaller than q, then f is *q-bounded*. In practice, we look for the minimal prime q for which f is q-bounded, which is the first prime larger than $f(p, \ldots, p)_{\mathbb{N}}$. It always holds that f is q-bounded for $q \geq p^{Np}$.

The Split-Input version of f. Let f as in (1), and $\varphi : \mathbb{F}_p^{2N} \to \mathbb{F}_p$ be the function obtained from f by replacing each variable x_i of f with the sum of two variables, z_i and w_i, as follows:

$$\begin{aligned} \varphi(z_1 \ldots, z_N, w_1, \ldots, w_N) &= \sum_{l\in\mathcal{L}} a_l \cdot (z_1 + w_1)^{l_1} \cdots (z_N + w_N)^{l_N} \\ &= \sum_{\lambda\in\Lambda} b_\lambda \cdot z_1^{\lambda_1} \cdots z_N^{\lambda_N} \cdot w_1^{\lambda_{N+1}} \cdots w_N^{\lambda_{2N}}, \end{aligned} \tag{2}$$

where $\lambda = (\lambda_1, \ldots, \lambda_{2N})$, $\Lambda = \{0, 1, \ldots, p-1\}^{2N}$, and $b_\lambda \in \mathbb{F}_p$. φ is the *split-input version* of f.

Let $\alpha_i, \beta_i \in \mathbb{F}_p$ $(1 \leq i \leq N)$. If for every $i \in [N]$ it holds that $s_i = \alpha_i + \beta_i$, then $f(s_1, \ldots, s_N) = \varphi(\alpha_1, \ldots, \alpha_N, \beta_1, \ldots, \beta_N)$.

B Proof of Theorem 2

Security. Let $h \in [N]$ and denote by $\mathcal{T}_h = \mathcal{P} - \{\mathcal{P}_h\}$ the adversarial coalition of size $N-1$ which contains all parties except for $\mathcal{P}_h$. W.l.o.g, we assume that $h = N$. We construct `Sim` as follows. Given $\{s_j\}_{j\in[N-1]}$ and $y = f(s_1, \ldots, s_N)$, the simulator chooses a uniformly random element from the set of possible values of s_N. Formally, let

$$E = \big\{a \big| f(s_1, \ldots, s_{N-1}, a) = y\big\} \subseteq \mathbb{F}_p,$$

and pick a uniformly randomly element s'_N of E. Then, `Sim` simulates the actions of all N parties according to the instructions of DBO-NV for N parties with secrets $s_1, \ldots, s_{N-1}, s'_N$, and outputs the simulated view of the first $N-1$ parties. By Lemma 1, for every secret, any subset of $N-1$ shares is uniformly distributed over $\mathbb{F}_p^{N-1}$, and hence, so is the part of the view of $\mathcal{T}_h$ that is viewed at the secret sharing stage. In *Com. 1*, each message received by $\mathcal{T}_h$ is some non-zero element $\prod_{m=1}^{N} s_{m,j}^{l_m}$, multiplied by some $c_{i,j}$. Since, those $c_{i,j}$'s are uniformly random non-zero elements, and multiplication by non-zero elements in $\mathbb{F}_p$ is a bijection, the part of the view of $\mathcal{T}_h$ that is viewed at *Com. 1* is uniformly distributed. The last message obtained by $\mathcal{T}_h$ from $\mathcal{P}_N$ at *Com. 2* is a function of the input y and the additive shares held by $\mathcal{T}_h$. We conclude that, given $s_1, \ldots, s_{N-1}$ and y, the view of $\mathcal{T}_h$ is uniformly distributed over the domain of possible views. Now, since the parties are honest-but-curious, they follow the instructions of DBO-NV, and hence, the simulated view output by `Sim` is identically distributed to $view_{\mathcal{T}_h}$.

Correctness. The correctness of the scheme follows from the fact that

$$\sum_{i=1}^{N} y_i = \sum_{l}\sum_{i=1}^{N} U_i^{(l)} = \sum_{l}\sum_{i=1}^{N} a_l \prod_{j=1}^{N} (\alpha_j)_i = \sum_{l} a_l \sum_{i=1}^{N}\prod_{j=1}^{N} \big(s_{1j}^{l_1} \ldots s_{Nj}^{l_N} \cdot [C^{(l)}]_j\big)_i$$

$$= \sum_{l} a_l \sum_{i=1}^{N}\left(s_1^{l_1} \ldots s_N^{l_N} \cdot \prod_{j=1}^{N} c_{ij}^{(l)}\right) = \sum_{l} a_l \cdot s_1^{l_1} \ldots s_N^{l_N} \cdot \sum_{i=1}^{N} \gamma_i^{(l)},$$

where $\gamma_i^{(l)}$ denotes the product $c_{i1}^{(l)} \ldots c_{in}^{(l)}$. Since each $C^{(l)}$ is a matrix-random-split of 1, we have $\sum_{i=1}^{N} \gamma_i^{(l)} = 1$ and hence, $\sum_{i=1}^{N} y_i = f(s_1, \ldots, s_N)$.

Communication complexity. We inspect the online phase of the scheme. By construction, the scheme requires two rounds. The total number of bits communicated, is $\mathcal{O}(N^2 kn)$. Indeed, in *Com. 1*, each of the N parties sends N messages, where each message is a k-tuple, and each entry of

that k-tuple is an $\mathbb{F}_p$ element. That is a total of N^2kn bits. In *Com. 2*, each party sends a single $\mathbb{F}_p$ element to all other parties, i.e., a total of Nn bits. All in all, $N^2kn + Nn$ bits are communicated.

Space complexity. How many bits of CR are required? At the preprocessing phase, for each monomial of f, each party obtains a single column of an order-N $\mathbb{F}_p$-valued square matrix. That is a total of kNn bits obtained by each party. Hence, the space complexity of the scheme is $\mathcal{O}(kNn)$. □

C Proof of Theorem 3

Since all q-world inputs are non-zero elements of $\mathbb{F}_q$, security of DBO-qB follows from that of DBO-NV. Correctness follows from that of DBO-NV and the fact that f is q-bounded. By construction, the round complexity is two. Now, in DBO-qB, all the messages and the CR are $\mathbb{F}_q$ elements. At the worst case, $q \approx p^{pN}$, and hence, the number of bits required for each element is $\lceil \log q \rceil \approx \log\left(p^{pN}\right) = pN \log p = 2^{\log p} N \log p = nN2^n$. Since the number of messages and amount of CR remains unchanged, the exact space and communication complexities of DBO-qB are obtained from those of DBO-NV by replacing n with $nN2^n$. □

D From the full version of the paper — OTS schemes

Theorem 6. *OTS-NV is a two-round N-party P-MPC scheme for arithmetic functions over non-zero inputs which has perfect correctness, perfect passive security, threshold $N-1$, communication complexity $\mathcal{O}(N^2nk)$, space complexity $\mathcal{O}(Nnk)$, and f-independent CR.*

Theorem 7. *OTS-qB is a two-round N-party P-MPC scheme for arithmetic functions which has perfect correctness, perfect passive security, threshold $N-1$, communication complexity $\mathcal{O}(kN^3n2^n)$, space complexity $\mathcal{O}(kN^2n2^n)$, and f-independent CR.*

Theorem 8. *OTS-IS is a two-round N-party P-MPC scheme for $\mathbb{F}_p$ functions ($p \neq 2$) which has perfect correctness, perfect passive security, threshold $N-1$, f-independent CR, communication complexity $\mathcal{O}(N^2nK)$, and space complexity $\mathcal{O}(NnK)$, where K is the number of monomials of the split-inputs version of f.*

Theorem 9. *OTS-IS$_2$ is a two-round N-party P-MPC scheme for arithmetic functions over $\mathbb{F}_2$ which has perfect correctness, perfect passive security, threshold $N-1$, f-independent CR, communication complexity $\mathcal{O}(N^2nK)$, and space complexity $\mathcal{O}(NnK)$, where K is the number of monomials of the split-input version of the $\mathbb{F}_3$ correspondent of f.*

E Bootstrapping DRM

The schemes described above may be bootstrapped to enable the users generate the CR themselves. To jointly generate a matrix random split of $1 \in \mathbb{F}_p$, let each party $\mathcal{P}_i$, $1 \leq i \leq N-1$ 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 $c'_{jN} = \frac{1}{c_{jN}}$. The function f may be written as an N-monomials polynomial by replacing the $c'_{i,j}s$ with their corresponding $\mathbb{F}_p$ inverses $c'_{i,j}$'s. Then, invoke the client-server scheme (qB version) 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 $\mathcal{P}_N$ obtains the N'th entry of an appropriate `DRM` share of $1 \in \mathbb{F}_p$. This way, the parties may generate the CR themselves. We note that, using such self-made CR induces leakage of some information in face of coalitions that contain $\mathcal{P}_N$.

F Polynomial representation of comparison function

To find the polynomial representation of the millionaire-comparison function f defined in Section 4, we solve the $\mathbb{F}_{11}$-system of 11^3 linear equations with 11^3 variables a_l, where $l = (l_1, l_2, l_3) \in \{0, 1, \ldots, 10\}^3$:

$$\begin{aligned} \sum_l a_l 0^{l_1} 0^{l_2} 0^{l_3} &= f(0,0,0), \\ &\vdots \\ \sum_l a_l 10^{l_1} 10^{l_2} 10^{l_3} &= f(10,10,10). \end{aligned} \tag{3}$$

We obtain: $f(x,y,z) = x^{10} + 7x^2y + 6x^4y + 7x^6y + 9x^8y + 3x^9y + 10x^{10}y + 4xy^2 + 10x^3y^2 + x^5y^2 + 8x^7y^2 + 3x^8y^2 + 5x^9y^2 + x^2y^3 + 2x^4y^3 + 7x^6y^3 + 3x^7y^3 + 7x^8y^3 + 5xy^4 + 9x^3y^4 + 6x^5y^4 + 3x^6y^4 + 8x^7y^4 + 10x^2y^5 + 5x^4y^5 + 3x^5y^5 + 2x^6y^5 + 4xy^6 + 4x^3y^6 + 3x^4y^6 + 9x^5y^6 + 3x^2y^7 + 3x^3y^7 + 3x^4y^7 + 2xy^8 + 3x^2y^8 + 4x^3y^8 + 3xy^9 + 6x^2y^9 + 2y^{10} + xy^{10} + 7x^{10}y^{10} + 3x^2z + x^4z + 3x^6z + 7x^8z + 8x^9z + 9x^{10}z + 8xyz + 9x^3yz + 3x^4yz + 2x^5yz + 10x^6yz + 5x^7yz + 5x^8yz + 10x^9yz + 8x^{10}yz + 7y^2z + 3x^2y^2z + 7x^3y^2z + 8x^4y^2z + 2x^5y^2z + 5x^7y^2z + 10x^8y^2z + 10x^9y^2z + 4x^{10}y^2z + 6xy^3z + x^2y^3z + 3x^3y^3z + 4x^4y^3z + 6x^5y^3z + 3x^6y^3z + 7x^7y^3z + x^9y^3z + 7x^{10}y^3z + 6y^4z + 4x^2y^4z + 7x^3y^4z + x^4y^4z + 2x^5y^4z + 9x^6y^4z + x^8y^4z + 9x^9y^4z + 5x^{10}y^4z + 6xy^5z + 10x^2y^5z + 7x^3y^5z + x^4y^5z + 5x^5y^5z + 2x^6y^5z + x^7y^5z + 7x^8y^5z + 4x^9y^5z + 6x^{10}y^5z + 7y^6z + 2x^2y^6z + 7x^3y^6z + 3x^4y^6z + 8x^5y^6z + x^6y^6z + 6x^7y^6z +$

$$4x^8y^6z+3x^9y^6z+4x^{10}y^6z+8xy^7z+3x^2y^7z+9x^3y^7z+6x^4y^7z+x^5y^7z+5x^6y^7z+4x^7y^7z+$$
$$5x^8y^7z+x^9y^7z+7x^{10}y^7z+9y^8z+10xy^8z+5x^2y^8z+7x^3y^8z+x^4y^8z+4x^5y^8z+4x^6y^8z+$$
$$6x^7y^8z+x^8y^8z+9x^9y^8z+2x^{10}y^8z+2y^9z+xy^9z+3x^2y^9z+x^3y^9z+2x^4y^9z+4x^5y^9z+$$
$$8x^6y^9z+x^7y^9z+2x^8y^9z+6x^9y^9z+10x^{10}y^9z+10y^{10}z+8x^2y^{10}z+4x^3y^{10}z+10x^4y^{10}z+$$
$$5x^5y^{10}z+8x^6y^{10}z+4x^7y^{10}z+4x^8y^{10}z+8x^9y^{10}z+8xz^2+9x^3z^2+2x^5z^2+5x^7z^2+$$
$$8x^8z^2+10x^9z^2+4yz^2+x^2yz^2+6x^3yz^2+3x^5yz^2+6x^6yz^2+x^7yz^2+7x^8yz^2+7x^9yz^2+$$
$$7x^{10}yz^2+10xy^2z^2+7x^3y^2z^2+x^5y^2z^2+7x^6y^2z^2+5x^7y^2z^2+7x^8y^2z^2+x^{10}y^2z^2+10y^3z^2+$$
$$10xy^3z^2+x^2y^3z^2+7x^4y^3z^2+5x^5y^3z^2+10x^6y^3z^2+3x^7y^3z^2+10x^8y^3z^2+2x^9y^3z^2+$$
$$x^{10}y^3z^2+10xy^4z^2+2x^3y^4z^2+6x^4y^4z^2+5x^5y^4z^2+10x^6y^4z^2+10x^7y^4z^2+x^8y^4z^2+$$
$$7x^9y^4z^2+9x^{10}y^4z^2+y^5z^2+xy^5z^2+8x^2y^5z^2+8x^4y^5z^2+6x^5y^5z^2+10x^6y^5z^2+9x^7y^5z^2+$$
$$3x^8y^5z^2+5x^9y^5z^2+10x^{10}y^5z^2+3xy^6z^2+10x^2y^6z^2+9x^3y^6z^2+2x^4y^6z^2+10x^5y^6z^2+$$
$$2x^6y^6z^2+3x^7y^6z^2+3x^8y^6z^2+9x^9y^6z^2+9x^{10}y^6z^2+8y^7z^2+6xy^7z^2+6x^2y^7z^2+5x^3y^7z^2+$$
$$10x^4y^7z^2+2x^5y^7z^2+3x^6y^7z^2+8x^8y^7z^2+6x^9y^7z^2+3x^{10}y^7z^2+2y^8z^2+10xy^8z^2+$$
$$10x^3y^8z^2+7x^4y^8z^2+3x^5y^8z^2+10x^6y^8z^2+8x^7y^8z^2+6x^8y^8z^2+4x^9y^8z^2+2x^{10}y^8z^2+$$
$$5y^9z^2+8xy^9z^2+2x^2y^9z^2+9x^3y^9z^2+4x^4y^9z^2+6x^5y^9z^2+5x^7y^9z^2+5x^8y^9z^2+6x^9y^9z^2+$$
$$4x^{10}y^9z^2+3xy^{10}z^2+2x^3y^{10}z^2+6x^4y^{10}z^2+9x^5y^{10}z^2+3x^6y^{10}z^2+6x^7y^{10}z^2+4x^9y^{10}z^2+$$
$$7x^{10}y^{10}z^2+2x^2z^3+4x^4z^3+3x^6z^3+8x^7z^3+3x^8z^3+3xyz^3+5x^2yz^3+8x^4yz^3+7x^5yz^3+$$
$$6x^6yz^3+3x^7yz^3+4x^8yz^3+10x^9yz^3+3x^{10}yz^3+y^2z^3+4xy^2z^3+6x^2y^2z^3+2x^4y^2z^3+$$
$$4x^5y^2z^3+9x^6y^2z^3+9x^7y^2z^3+x^8y^2z^3+4x^9y^2z^3+10x^{10}y^2z^3+8xy^3z^3+8x^3y^3z^3+$$
$$3x^4y^3z^3+7x^5y^3z^3+3x^6y^3z^3+6x^8y^3z^3+3x^{10}y^3z^3+2y^4z^3+4xy^4z^3+3x^2y^4z^3+10x^3y^4z^3+$$
$$9x^4y^4z^3+2x^5y^4z^3+x^8y^4z^3+9x^{10}y^4z^3+9xy^5z^3+9x^2y^5z^3+4x^3y^5z^3+9x^4y^5z^3+4x^6y^5z^3+$$
$$8x^7y^5z^3+4x^8y^5z^3+2x^9y^5z^3+7y^6z^3+2xy^6z^3+7x^2y^6z^3+6x^5y^6z^3+8x^6y^6z^3+6x^8y^6z^3+$$
$$5x^9y^6z^3+4x^{10}y^6z^3+2y^7z^3+xy^7z^3+6x^2y^7z^3+x^4y^7z^3+8x^5y^7z^3+3x^6y^7z^3+5x^7y^7z^3+$$
$$x^8y^7z^3+10x^9y^7z^3+3x^{10}y^7z^3+7y^8z^3+4xy^8z^3+3x^2y^8z^3+9x^4y^8z^3+x^5y^8z^3+8x^6y^8z^3+$$
$$6x^7y^8z^3+3x^8y^8z^3+2x^9y^8z^3+2x^{10}y^8z^3+8xy^9z^3+2x^2y^9z^3+6x^3y^9z^3+2x^4y^9z^3+$$
$$7x^5y^9z^3+x^6y^9z^3+6x^7y^9z^3+2x^8y^9z^3+7x^9y^9z^3+8x^{10}y^9z^3+7xy^{10}z^3+9x^2y^{10}z^3+$$
$$7x^4y^{10}z^3+8x^5y^{10}z^3+8x^6y^{10}z^3+2x^7y^{10}z^3+10x^9y^{10}z^3+10xz^4+7x^3z^4+x^5z^4+8x^6z^4+$$
$$5x^7z^4+5yz^4+8xyz^4+3x^2yz^4+3x^3yz^4+8x^4yz^4+10x^6yz^4+5x^7yz^4+10x^8yz^4+$$
$$7x^9yz^4+6x^{10}yz^4+6xy^2z^4+3x^3y^2z^4+10x^4y^2z^4+9x^5y^2z^4+8x^6y^2z^4+x^7y^2z^4+5x^8y^2z^4+$$
$$4x^9y^2z^4+3x^{10}y^2z^4+9y^3z^4+7xy^3z^4+9x^2y^3z^4+6x^3y^3z^4+3x^4y^3z^4+4x^5y^3z^4+10x^7y^3z^4+$$
$$10x^8y^3z^4+9x^9y^3z^4+2x^{10}y^3z^4+4xy^4z^4+3x^2y^4z^4+8x^3y^4z^4+3x^6y^4z^4+4x^8y^4z^4+$$
$$5x^{10}y^4z^4+6y^5z^4+8xy^5z^4+5x^2y^5z^4+2x^3y^5z^4+7x^5y^5z^4+7x^6y^5z^4+3x^7y^5z^4+3x^8y^5z^4+$$
$$9x^9y^5z^4+5x^{10}y^5z^4+2y^6z^4+5xy^6z^4+9x^2y^6z^4+7x^5y^6z^4+4x^6y^6z^4+7x^7y^6z^4+8x^8y^6z^4+$$
$$5x^9y^6z^4+4x^{10}y^6z^4+8y^7z^4+5xy^7z^4+3x^2y^7z^4+10x^3y^7z^4+8x^4y^7z^4+10x^5y^7z^4+$$
$$9x^6y^7z^4+8x^7y^7z^4+9x^8y^7z^4+8x^9y^7z^4+x^{10}y^7z^4+8xy^8z^4+4x^2y^8z^4+5x^3y^8z^4+$$
$$8x^5y^8z^4+9x^6y^8z^4+4x^7y^8z^4+8x^8y^8z^4+9x^9y^8z^4+5x^{10}y^8z^4+9xy^9z^4+9x^2y^9z^4+$$
$$9x^3y^9z^4+10x^4y^9z^4+6x^5y^9z^4+7x^6y^9z^4+3x^7y^9z^4+8x^8y^9z^4+10x^9y^9z^4+xy^{10}z^4+$$
$$5x^2y^{10}z^4+4x^3y^{10}z^4+10x^5y^{10}z^4+4x^6y^{10}z^4+9x^7y^{10}z^4+7x^8y^{10}z^4+9x^2z^5+10x^4z^5+$$
$$8x^5z^5+4x^6z^5+10xyz^5+8x^2yz^5+9x^3yz^5+9x^4yz^5+9x^5yz^5+5x^6yz^5+10x^7yz^5+3x^8yz^5+$$
$$7x^9yz^5+x^{10}yz^5+10y^2z^5+9xy^2z^5+4x^2y^2z^5+5x^3y^2z^5+4x^4y^2z^5+x^6y^2z^5+7x^7y^2z^5+$$
$$8x^8y^2z^5+10x^9y^2z^5+x^{10}y^2z^5+10xy^3z^5+4x^2y^3z^5+x^3y^3z^5+7x^4y^3z^5+9x^6y^3z^5+$$
$$3x^7y^3z^5+3x^8y^3z^5+9x^9y^3z^5+3x^{10}y^3z^5+5y^4z^5+7xy^4z^5+8x^2y^4z^5+9x^3y^4z^5+4x^6y^4z^5+$$
$$4x^7y^4z^5+8x^8y^4z^5+3x^9y^4z^5+6x^{10}y^4z^5+2y^5z^5+2xy^5z^5+5x^2y^5z^5+4x^4y^5z^5+4x^6y^5z^5+$$
$$7x^8y^5z^5+5x^{10}y^5z^5+2y^6z^5+3xy^6z^5+3x^2y^6z^5+5x^3y^6z^5+x^4y^6z^5+7x^6y^6z^5+5x^7y^6z^5+$$
$$4x^8y^6z^5+3x^9y^6z^5+7x^{10}y^6z^5+8xy^7z^5+9x^2y^7z^5+9x^3y^7z^5+x^4y^7z^5+5x^5y^7z^5+x^6y^7z^5+$$
$$10x^7y^7z^5+7x^8y^7z^5+2x^9y^7z^5+8x^{10}y^7z^5+7xy^8z^5+2x^2y^8z^5+10x^3y^8z^5+7x^4y^8z^5+$$
$$5x^6y^8z^5+4x^7y^8z^5+9x^8y^8z^5+5x^9y^8z^5+10xy^9z^5+5x^2y^9z^5+10x^3y^9z^5+5x^4y^9z^5+$$
$$8x^5y^9z^5+4x^6y^9z^5+6x^7y^9z^5+x^8y^9z^5+6xy^{10}z^5+2x^2y^{10}z^5+3x^3y^{10}z^5+x^4y^{10}z^5+$$
$$6x^5y^{10}z^5+10x^6y^{10}z^5+x^7y^{10}z^5+8xz^6+8x^3z^6+8x^4z^6+7x^5z^6+4yz^6+xyz^6+10x^2yz^6+$$

$$3x^3yz^6+9x^4yz^6+6x^5yz^6+10x^6yz^6+x^7yz^6+7x^8yz^6+6x^9yz^6+7x^{10}yz^6+5xy^2z^6+2x^2y^2z^6+7x^3y^2z^6+3x^4y^2z^6+x^5y^2z^6+8x^7y^2z^6+4x^8y^2z^6+2x^9y^2z^6+6x^{10}y^2z^6+4y^3z^6+6xy^3z^6+6x^2y^3z^6+8x^3y^3z^6+2x^5y^3z^6+3x^6y^3z^6+8x^7y^3z^6+5x^8y^3z^6+4x^9y^3z^6+7x^{10}y^3z^6+2y^4z^6+10xy^4z^6+x^2y^4z^6+8x^4y^4z^6+4x^5y^4z^6+4x^7y^4z^6+10x^8y^4z^6+6x^9y^4z^6+6x^{10}y^4z^6+9y^5z^6+9xy^5z^6+3x^2y^5z^6+7x^3y^5z^6+x^4y^5z^6+7x^5y^5z^6+10x^6y^5z^6+4x^7y^5z^6+10x^8y^5z^6+9x^9y^5z^6+8xy^6z^6+9x^2y^6z^6+9x^3y^6z^6+7x^4y^6z^6+2x^5y^6z^6+5x^7y^6z^6+5x^8y^6z^6+10x^9y^6z^6+5x^{10}y^6z^6+6xy^7z^6+2x^2y^7z^6+8x^3y^7z^6+3x^4y^7z^6+10x^5y^7z^6+7x^8y^7z^6+2x^9y^7z^6+10xy^8z^6+x^2y^8z^6+6x^3y^8z^6+2x^4y^8z^6+6x^5y^8z^6+10x^6y^8z^6+x^7y^8z^6+2x^8y^8z^6+3xy^9z^6+8x^2y^9z^6+10x^3y^9z^6+9x^4y^9z^6+5x^5y^9z^6+6x^6y^9z^6+8x^7y^9z^6+3xy^{10}z^6+8x^2y^{10}z^6+3x^3y^{10}z^6+8x^4y^{10}z^6+7x^5y^{10}z^6+6x^2z^7+8x^3z^7+6x^4z^7+8x^2yz^7+7x^3yz^7+6x^4yz^7+10x^5yz^7+10x^6yz^7+7x^7yz^7+10x^8yz^7+10x^9yz^7+3x^{10}yz^7+3y^2z^7+4xy^2z^7+7x^2y^2z^7+2x^3y^2z^7+x^4y^2z^7+4x^5y^2z^7+8x^6y^2z^7+3x^8y^2z^7+x^9y^2z^7+8x^{10}y^2z^7+2y^3z^7+3xy^3z^7+8x^2y^3z^7+x^4y^3z^7+3x^5y^3z^7+3x^6y^3z^7+6x^7y^3z^7+6x^8y^3z^7+x^9y^3z^7+7x^{10}y^3z^7+3y^4z^7+3x^2y^4z^7+5x^4y^4z^7+7x^5y^4z^7+8x^6y^4z^7+5x^8y^4z^7+4x^9y^4z^7+6x^{10}y^4z^7+8xy^5z^7+2x^2y^5z^7+9x^3y^5z^7+8x^4y^5z^7+10x^5y^5z^7+7x^6y^5z^7+2x^8y^5z^7+7x^9y^5z^7+7x^{10}y^5z^7+5xy^6z^7+2x^2y^6z^7+5x^4y^6z^7+6x^5y^6z^7+6x^6y^6z^7+10x^7y^6z^7+10x^8y^6z^7+5x^9y^6z^7+10xy^7z^7+7x^3y^7z^7+3x^4y^7z^7+x^5y^7z^7+9x^6y^7z^7+5x^7y^7z^7+3x^8y^7z^7+5xy^8z^7+9x^2y^8z^7+5x^3y^8z^7+7x^4y^8z^7+5x^5y^8z^7+7x^6y^8z^7+8xy^9z^7+6x^2y^9z^7+10x^3y^9z^7+6x^4y^9z^7+5x^5y^9z^7+3x^6y^9z^7+7xy^{10}z^7+5x^2y^{10}z^7+10x^3y^{10}z^7+8x^4y^{10}z^7+10x^5y^{10}z^7+4xz^8+8x^2z^8+8x^3z^8+2yz^8+4xyz^8+8x^2yz^8+7x^3yz^8+10x^4yz^8+8x^5yz^8+7x^6yz^8+x^7yz^8+10x^8yz^8+7x^9yz^8+9x^{10}yz^8+2y^2z^8+5xy^2z^8+4x^2y^2z^8+x^3y^2z^8+6x^4y^2z^8+8x^5y^2z^8+7x^6y^2z^8+3x^7y^2z^8+7x^9y^2z^8+8x^{10}y^2z^8+4y^3z^8+3x^2y^3z^8+5x^3y^3z^8+4x^4y^3z^8+8x^5y^3z^8+9x^6y^3z^8+5x^7y^3z^8+10x^9y^3z^8+5x^{10}y^3z^8+8xy^4z^8+10x^2y^4z^8+2x^3y^4z^8+7x^4y^4z^8+7x^5y^4z^8+x^6y^4z^8+6x^7y^4z^8+10x^8y^4z^8+9x^{10}y^4z^8+4xy^5z^8+2x^2y^5z^8+7x^3y^5z^8+x^4y^5z^8+4x^5y^5z^8+x^6y^5z^8+7x^7y^5z^8+8x^8y^5z^8+4x^9y^5z^8+10xy^6z^8+8x^2y^6z^8+8x^3y^6z^8+3x^4y^6z^8+7x^5y^6z^8+4x^6y^6z^8+9x^7y^6z^8+6xy^7z^8+9x^2y^7z^8+10x^3y^7z^8+2x^4y^7z^8+2x^5y^7z^8+x^6y^7z^8+8x^7y^7z^8+8xy^8z^8+5x^2y^8z^8+8x^3y^8z^8+x^4y^8z^8+7x^5y^8z^8+9x^6y^8z^8+9xy^9z^8+7x^3y^9z^8+x^4y^9z^8+10x^5y^9z^8+7xy^{10}z^8+x^2y^{10}z^8+6x^3y^{10}z^8+4x^4y^{10}z^8+8xz^9+x^2z^9+2yz^9+3xyz^9+4x^2yz^9+10x^3yz^9+4x^4yz^9+7x^5yz^9+5x^6yz^9+10x^7yz^9+4x^8yz^9+5x^9yz^9+8x^{10}yz^9+6y^2z^9+xy^2z^9+4x^2y^2z^9+7x^3y^2z^9+9x^4y^2z^9+x^5y^2z^9+6x^6y^2z^9+10x^7y^2z^9+9x^8y^2z^9+10x^9y^2z^9+3x^{10}y^2z^9+8xy^3z^9+9x^2y^3z^9+x^3y^3z^9+2x^4y^3z^9+8x^5y^3z^9+7x^6y^3z^9+4x^7y^3z^9+10x^8y^3z^9+2x^9y^3z^9+9x^{10}y^3z^9+2xy^4z^9+6x^2y^4z^9+9x^4y^4z^9+8x^5y^4z^9+10x^6y^4z^9+5x^7y^4z^9+9x^8y^4z^9+10xy^5z^9+6x^2y^5z^9+4x^3y^5z^9+2x^4y^5z^9+5x^5y^5z^9+4x^7y^5z^9+7x^8y^5z^9+8xy^6z^9+10x^2y^6z^9+6x^3y^6z^9+6x^5y^6z^9+2x^6y^6z^9+6x^7y^6z^9+8xy^7z^9+5x^2y^7z^9+6x^3y^7z^9+x^4y^7z^9+9x^5y^7z^9+9x^6y^7z^9+2xy^8z^9+x^2y^8z^9+7x^3y^8z^9+6x^5y^8z^9+4xy^9z^9+3x^2y^9z^9+8x^3y^9z^9+x^4y^9z^9+4xy^{10}z^9+2x^2y^{10}z^9+x^3y^{10}z^9+3z^{10}+2xz^{10}+6x^{10}z^{10}+yz^{10}+3xyz^{10}+4x^2yz^{10}+8x^3yz^{10}+5x^4yz^{10}+10x^5yz^{10}+4x^6yz^{10}+8x^7yz^{10}+2x^8yz^{10}+4x^9yz^{10}+3x^{10}yz^{10}+7xy^2z^{10}+10x^2y^2z^{10}+x^3y^2z^{10}+8x^4y^2z^{10}+10x^5y^2z^{10}+5x^6y^2z^{10}+3x^7y^2z^{10}+4x^8y^2z^{10}+3x^9y^2z^{10}+4xy^3z^{10}+10x^2y^3z^{10}+8x^3y^3z^{10}+9x^4y^3z^{10}+8x^5y^3z^{10}+4x^6y^3z^{10}+5x^7y^3z^{10}+x^8y^3z^{10}+2x^9y^3z^{10}+6xy^4z^{10}+2x^2y^4z^{10}+2x^3y^4z^{10}+6x^4y^4z^{10}+5x^5y^4z^{10}+6x^6y^4z^{10}+2x^8y^4z^{10}+5xy^5z^{10}+x^2y^5z^{10}+6x^4y^5z^{10}+7x^5y^5z^{10}+6x^6y^5z^{10}+4x^7y^5z^{10}+7xy^6z^{10}+2x^2y^6z^{10}+7x^3y^6z^{10}+8x^4y^6z^{10}+10x^5y^6z^{10}+6x^6y^6z^{10}+4xy^7z^{10}+8x^2y^7z^{10}+9x^3y^7z^{10}+5x^4y^7z^{10}+3x^5y^7z^{10}+9xy^8z^{10}+10x^2y^8z^{10}+4x^3y^8z^{10}+6x^4y^8z^{10}+2xy^9z^{10}+2x^2y^9z^{10}+3x^3y^9z^{10}+5y^{10}z^{10}+6xy^{10}z^{10}+4x^2y^{10}z^{10}.$$