Information-Theoretically Secure Distributed Protocols for Two-Party MPC Primitives

TAMIR TASSA*, The Open University of Israel, Israel

AVIAD BEN ARIE, The Open University of Israel, Israel

We study two-party secure computation in a *distributed mediated* model with D servers. We present *information-theoretically secure* protocols for foundational primitives— 1-out-of-N and k-out-of-N oblivious transfer (OT), Priced OT, Generalized OT, and Oblivious Multivariate Polynomial Evaluation (OMPE)—that (i) enforce receiver correctness, (ii) support an offline sender after setup, and (iii) achieve robustness against malicious receivers and malicious servers under honest-majority assumptions.

Our approach reduces each primitive to a distributed scalar-product core combined with a *distributed vector-validation* (DVV) mechanism that prevents unauthorized linear-combination leakage while preserving privacy. We prove unconditional privacy and robustness in the semi-honest setting and extend the guarantees to the malicious-server setting, tolerating up to $c < D/4$ or $c < D/3$ corrupt servers depending on the variant. The constructions are modular, require only lightweight client work (sharing and reconstruction), and allow the sender's one-time upload to serve multiple, independent receivers without further interaction.

We analyze the round and communication complexity of our protocols and provide empirical evidence of practical efficiency. The mediated setting naturally fits privacy-preserving services such as collaborative filtering, distributed constraint optimization, voting, and federated analytics. Collectively, our results show that distributed mediation yields simple, scalable, and unconditionally private realizations of core two-party MPC functionalities.

Additional Key Words and Phrases: Secure Multiparty Computation, Distributed Mediation, Oblivious Transfer, Priced OT, Generalized OT, Oblivious Polynomial Evaluation, Secret Sharing, Information-Theoretic Security

ACM Reference Format:

Tamir Tassa and Aviad Ben Arie. 2025. Information-Theoretically Secure Distributed Protocols for Two-Party MPC Primitives. *ACM Trans. Priv. Sec.* 1, 1 (November 2025), 28 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

*Corresponding author.

Authors' Contact Information: Tamir Tassa, tamir_tassa@yahoo.com, The Open University of Israel, Ra'anana, Israel; Aviad Ben Arie, lionson.aviad@gmail.com, The Open University of Israel, Ra'anana, Israel.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

© 2025 Copyright held by the owner/author(s). Publication rights licensed to ACM.

Manuscript submitted to ACM

1 Introduction

Secure multiparty computation (MPC) [38] allows mutually distrustful parties to compute a function of their private inputs while revealing nothing beyond the output. In the classical setting, n parties $P_1, \ldots, P_n$ with inputs $x_1, \ldots, x_n$ wish to compute $f(x_1, \ldots, x_n)$ with privacy and correctness guarantees. Beyond this standard formulation, an active line of work studies architectures in which computation is partially *outsourced* to a set of mediating servers. Such models include the mediated model [2, 3, 15–17, 30, 32], various client–server frameworks [10, 14, 19, 28], and distributed MPC protocols [8, 12, 13, 21, 24, 26]. In these settings, clients submit secret shares of their private inputs to D servers, which evaluate the desired functionality on shares and return shares of the result. This structure simplifies client involvement, avoids client–client interaction, and supports Cloud-style deployments and computationally constrained clients.

Our focus. We adopt this distributed mediated architecture in the fundamental two-party setting with a *sender* Alice and a *receiver* Bob. Alice holds $\mathbf{a} = (a_1, \ldots, a_N) \in \mathbb{Z}_p^N$ and Bob holds $\mathbf{b} = (b_1, \ldots, b_N) \in \mathbb{Z}_p^N$. Both parties secret-share their inputs among $D > 2$ servers. The servers validate client inputs (when required), compute the desired functionality on shares, and return output shares that clients reconstruct locally. Clients perform only lightweight sharing and reconstruction steps, while servers carry out all core computation.

Challenges in prior work. Existing distributed protocols for oblivious transfer (OT) and oblivious polynomial evaluation (OPE) typically assume semi-honest receivers, permit certain unauthorized linear combinations of the sender's data, or require the sender to remain online for each query. These limitations can impact practical applicability: even a single malicious receiver may extract more than one message or learn non-permitted information about polynomial coefficients, and ongoing sender availability may become a bottleneck in multi-client deployments. Achieving malicious-receiver security together with an *offline sender* and support for *unlimited, independent receiver queries* remains insufficiently addressed in the distributed mediated setting.

Our contributions. We develop a unified framework for distributed, information-theoretically secure realizations of several foundational two-party MPC primitives. Our contributions are organized into three parts.

(I) Methodological contributions. We introduce a simple but powerful combination of:

(1) a lightweight *distributed scalar-product* (DSP) mechanism; and
(2) a *distributed vector-validation* (DVV) procedure that enforces authorized query structure and prevents unauthorized linear-combination leakage.

These components serve as a unifying abstraction enabling a one-time sender upload followed by unlimited, independent queries from multiple receivers.

(II) Protocol contributions. Using DSP and DVV, we obtain:

- **Distributed OT with malicious-receiver security.** We present a simple protocol ensuring that the receiver learns *exactly one* message and no other information. In contrast, prior distributed OT constructions typically allowed the receiver to obtain a single linear combination of all messages, rather than enforcing retrieval of one specific message.
- **k-out-of-N OT, Priced OT, and Generalized OT.** We give the *first distributed protocols* for these variants with malicious-receiver security in the mediated setting, enforcing authorized retrieval via DVV.

- **Oblivious Multivariate Polynomial Evaluation (OMPE).** We provide the *first distributed protocol* for OMPE supporting arbitrary multivariate polynomials, ensuring correctness of the queried evaluation point and preventing leakage of polynomial coefficients.

(III) Security and robustness. All protocols offer unconditional privacy against semi-honest servers with an honest majority. We extend security to malicious servers, proving information-theoretic guarantees for corruption thresholds $c < D/4$ or $c < D/3$, depending on the primitive. Crucially, all variants require only a *single* sender upload, after which the sender may remain offline while multiple receivers issue uncoordinated queries.

Practical relevance. The offline-sender property is particularly useful in applications where a data custodian (e.g., a hospital, financial institution, or service provider) cannot interact with every client request. After the initial upload, clients can privately retrieve data or evaluate functions on it without revealing their queries, while the custodian remains offline. This structure naturally supports asynchronous access, federated analytics, distributed constraint optimization, secure voting, and collaborative filtering [9, 15, 20, 22, 25, 33, 34, 36]. By strengthening distributed OT and OPE mechanisms and providing concrete performance evaluation, our work supports practical deployments of privacy-preserving systems based on secret sharing and distributed cryptography.

Overall, our results show that distributed mediation yields simple, efficient, and unconditionally secure realizations of core two-party MPC primitives and offers a compelling model for practical privacy-preserving outsourced computation.

2 Preliminaries

System and communication model. We consider two clients (Alice and Bob) and D servers $M_1, \dots, M_D$. Communication proceeds in synchronous lock-step rounds: all messages sent in a round are received before the next round begins. Servers communicate over an authenticated broadcast channel; no pairwise channels between servers are assumed. Clients communicate only with servers during the input and output phases and do not communicate directly with one another.

Secret sharing. Clients secret-share their inputs using (t, D) Shamir secret sharing (i.e., a t-out-of-D scheme) [29], where

$$t = \lfloor (D+1)/2 \rfloor . \tag{1}$$

For each entry a_n, Alice samples random coefficients $\alpha_{n,1}, \dots, \alpha_{n,t-1} \in \mathbb{Z}_p$, defines $f_n^A(x) = a_n + \sum_{i=1}^{t-1} \alpha_{n,i} x^i$, and sends the share $[a_n]_d = f_n^A(d)$ to server M_d. Bob shares his inputs identically.[1] Servers compute on shares throughout the protocol and return output shares for local reconstruction. Since we target information-theoretic security, a 64-bit prime p suffices in practice.

Arithmetic on shared secrets. We use standard Shamir-share operations.

Affine combinations. Given (t, D)-shares of s_1, s_2 and public $a, b, c \in \mathbb{Z}_p$, each server computes $a[s_1]_d + b[s_2]_d + c$, yielding a share of $as_1 + bs_2 + c$.

Multiplication. Given shares of s_1 and s_2, server M_d computes $c_d = [s_1]_d \cdot [s_2]_d$. The values $\{c_d\}_{d \in [D]}$ form a $(2t-1, D)$-sharing of $s_1 s_2$. The servers interpolate the unique degree-$(2t-2)$ polynomial through these points, obtaining $s_1 s_2$ as the constant term. Assuming D is odd gives $2t - 1 = D$. We do not require BGW [7] degree reduction since all computations are quadratic.

[1] We use square brackets for both index sets (e.g., $[N] = \{1, \dots, N\}$) and shares (e.g., $[s]_d$); the context and subscripts disambiguate the two uses.

Scrambling shares. When multiplying related secrets (e.g., $s_2 = s_1 - 1$), intermediate products may leak information. To re-randomize, each server broadcasts a random (D, D)-share of 0; summing all broadcasts yields fresh $[0]_d$ shares. Servers output $c'_d = c_d + [0]_d$, obtaining a uniformly random sharing of $s_1 s_2$ independent of $\{c_d\}$. Zero-shares can be precomputed offline.

Adversarial model and security goals. Security is defined via the simulation-based ideal/real paradigm: a protocol is secure if every real-world adversary has a simulator whose ideal-world view is statistically indistinguishable. We first analyze *semi-honest* servers with fewer than t colluding servers (Eq. (1)). We then consider *malicious* servers that may deviate arbitrarily. In Section 8 we prove information-theoretic security for corruption thresholds $c < D/4$ and $c < D/3$, summarized in Table 3.

3 Distributed Scalar Product

The distributed scalar product (DSP) functionality serves as the core building block for all protocols in this paper. It allows two clients to outsource the computation of an inner product to a set of servers while keeping both input vectors private.

Definition 1 (Distributed Scalar Product (DSP)). *Alice holds a vector* $\mathbf{a} = (a_1, \ldots, a_N) \in \mathbb{Z}_p^N$ *and Bob holds a vector* $\mathbf{b} = (b_1, \ldots, b_N) \in \mathbb{Z}_p^N$. *The goal is to compute their scalar product* $\mathbf{a} \cdot \mathbf{b} = \sum_{n=1}^N a_n b_n$ *while revealing nothing about* $\mathbf{a}$ *or* $\mathbf{b}$ *beyond the value of this product.*

Protocol 1 gives a simple and efficient realization of this functionality in the mediated setting. In the first loop (Lines 1–3), Alice and Bob distribute (t, D) Shamir shares of each input coordinate to the D servers. Each server M_d then locally forms

$$s_d \leftarrow \sum_{n \in [N]} \left([a_n]_d \cdot [b_n]_d \right)$$

and sends s_d to Alice and Bob (Lines 4–6). Let f_n^A and f_n^B denote the degree-$(t-1)$ Shamir polynomials used to share a_n and b_n, respectively. Because $[a_n]_d = f_n^A(d)$ and $[b_n]_d = f_n^B(d)$, we have

$$s_d = \sum_n f_n^A(d)\, f_n^B(d) = F(d),$$

where

$$F(x) := \sum_{n=1}^N f_n^A(x)\, f_n^B(x)$$

is a polynomial of degree at most $2t-2$, whose constant term satisfies $F(0) = \sum_n a_n b_n = \mathbf{a} \cdot \mathbf{b}$. Since the D evaluations $\{s_d\}_{d \in [D]}$ uniquely determine any polynomial of degree at most $2t-2$, Alice and Bob interpolate F and recover the correct value $F(0)$ (Line 7).

Theorem 1. *Assume semi-honest servers with an honest majority (i.e., at most* $t-1$ *colluding servers). Then Protocol 1 is correct and statistically (information-theoretically) private for both Alice and Bob. Moreover, even if Alice (resp., Bob) colludes with any subset of* $t-1$ *servers, the coalition learns nothing beyond that party's own input and the output* $\mathbf{a} \cdot \mathbf{b}$.

Proof. *Correctness.* For each n, let f_n^A and f_n^B be the degree-$(t-1)$ sharing polynomials with $f_n^A(0) = a_n$ and $f_n^B(0) = b_n$. Define $F(x) = \sum_{n=1}^N f_n^A(x) f_n^B(x)$. As $\deg F \le 2t-2$ and $D \ge 2t-1$, the set $s_1, \ldots, s_D$ uniquely recovers $F(0) = \sum_n a_n b_n = \mathbf{a} \cdot \mathbf{b}$.

Protocol 1: Distributed Scalar Product

Parameters: p—field size; N—dimension; D—number of servers; $t = \lfloor (D+1)/2 \rfloor$.
Inputs: Alice has $\mathbf{a} = (a_1, \dots, a_N) \in \mathbb{Z}_p^N$, Bob has $\mathbf{b} = (b_1, \dots, b_N) \in \mathbb{Z}_p^N$.

forall $n \in [N]$ **do**
  Alice sends to each M_d, $d \in [D]$, a (t, D)-share $[a_n]_d$ of a_n.
  Bob sends to each M_d, $d \in [D]$, a (t, D)-share $[b_n]_d$ of b_n.
forall $d \in [D]$ **do**
  M_d computes $s_d \leftarrow \sum_{n \in [N]} \big([a_n]_d \cdot [b_n]_d\big)$.
  M_d sends s_d to Alice and to Bob.
Alice and Bob interpolate the (unique) polynomial F of degree $\leq 2t-2$ from $\{(d, s_d)\}_{d \in [D]}$ and output $F(0)$.

Output: Alice and Bob obtain $\mathbf{a} \cdot \mathbf{b}$.

Privacy against Bob and up to $t-1$ servers. Fix any coalition consisting of Bob and a set $S \subseteq [D]$ of servers with $|S| = t-1$. We construct a simulator Sim that, given only Bob's input $\mathbf{b}$ and the output $\alpha := \mathbf{a} \cdot \mathbf{b}$, produces a view identically distributed to the coalition's real view.

Setup and handling of indices. If $\mathbf{b} = \mathbf{0}$, then $\alpha = 0$ and Alice's input remains completely hidden by Shamir secrecy against $t-1$ servers, so simulation is trivial. Assume henceforth $\mathbf{b} \neq \mathbf{0}$. Choose any index $j \in [N]$ with $b_j \neq 0$. If $j \neq N$, relabel the coordinates by swapping positions j and N; this preserves solvability and distributions. After relabeling we have $b_N \neq 0$.

Sampling a consistent fake input for Alice. Sim samples $a'_1, \dots, a'_{N-1} \xleftarrow{\$} \mathbb{Z}_p$ uniformly at random and sets $a'_N := b_N^{-1}\big(\alpha - \sum_{n=1}^{N-1} a'_n b_n\big)$. Thus $\sum_{n=1}^{N} a'_n b_n = \alpha$.

Simulating Alice's shares to the colluding servers. For each $n \in [N]$, the simulator fixes $f'^A_n(0) = a'_n$ and samples a random polynomial f'^A_n of degree at most $t-1$ conditioned on its evaluations at the colluding servers S being uniformly random in $\mathbb{Z}_p$. By Shamir secrecy, conditioned on the constant term a'_n, the evaluations $\{f^A_n(d)\}_{d \in S}$ are uniformly distributed and independent of a'_n; hence $\{f'^A_n(d)\}_{d \in S}$ is identically distributed to the real $\{f^A_n(d)\}_{d \in S}$.

Simulating the servers' outputs. The simulator uses Bob's actual shares and defines $F'(x) := \sum_{n=1}^{N} f'^A_n(x)\, f^B_n(x)$, where f^B_n are Bob's actual sharing polynomials, and outputs $s'_d := F'(d)$ for all $d \in [D]$ as the servers' messages to Bob.

Indistinguishability. By construction, $F'(0) = \sum_n a'_n b_n = \alpha$, so the vector $(s'_d)_{d \in [D]} = (F'(d))_{d \in [D]}$ has the same distribution as in the real execution. Moreover, the joint distribution of the colluding servers' input shares from Alice is identical to the real one by Shamir secrecy. Therefore, the simulator's output is identically distributed to the coalition's real view given $(\mathbf{b}, \alpha)$, establishing statistical privacy for Bob.

Privacy against Alice with up to $t-1$ servers. (The argument is completely symmetric and follows by exchanging the roles of Alice and Bob.) □

4 Distributed Oblivious Transfer

In this section we study several variants of the Oblivious Transfer (OT) protocol. We begin with the standard 1-out-of-N and k-out-of-N variants (Section 4.1), then discuss Priced OT (Section 4.2), and finally address Generalized OT (Section 4.3).

We consider here input-malicious receivers who may submit illegal selection vectors while otherwise performing proper secret sharing; servers are semi-honest with honest majority. Malicious servers are treated later in Section 8.

4.1 k-out-of-N Oblivious Transfer

We first formalize the k-out-of-N OT functionality.

Definition 2. *(OT_k^N) Assume that Alice holds N messages, $m_1, \ldots, m_N \in \mathbb{Z}_p$. Bob wishes to learn k of these messages, say $m_{j_1}, \ldots, m_{j_k}$, for some distinct $j_1, \ldots, j_k \in [N]$. A k-out-of-N Oblivious Transfer (OT_k^N) protocol enables Bob to obtain exactly $m_{j_1}, \ldots, m_{j_k}$ while learning nothing about the remaining messages, and ensures that Alice learns nothing about Bob's chosen indices.*

We begin with the case $k = 1$ and then generalize to arbitrary $k > 1$. The OT_1^N problem can be reduced to the Distributed Scalar Product (DSP) functionality (Section 3) if Alice sets $\mathbf{a} = (m_1, \ldots, m_N)$ and Bob sets $\mathbf{b} = \mathbf{e}_j$, the unit vector whose jth entry is 1 and all other entries are 0. However, executing the DSP protocol directly would allow a dishonest Bob to send shares of a non-unit vector and thereby obtain a linear combination of the messages, potentially leaking additional information. For example, if Bob knows that m_1 and m_2 lie in distinct one-dimensional subspaces of $\mathbb{Z}_p^N$, he could recover both messages by requesting $m_1 + m_2$. To prevent this, the DSP must be preceded by a validation phase.

Definition 3. *(DVV) Let $M_1, \ldots, M_D$ hold (t, D)-shares of a vector $\mathbf{v} \in \mathbb{Z}_p^N$. For a subset $W \subseteq \mathbb{Z}_p^N$, a* Distributed Vector Validation (DVV) *protocol allows the servers to jointly verify that $\mathbf{v} \in W$ (output 1) or otherwise output 0, without revealing any additional information about $\mathbf{v}$ when $\mathbf{v} \in W$.*

In our setting $W = \{\mathbf{e}_j : j \in [N]\}$. The servers validate that $\mathbf{b} \in W$ by checking: (1) $b_n(b_n - 1) = 0$ for all $n \in [N]$, and (2) $\sum_{n=1}^N b_n = 1$. Condition (1) ensures each b_n is in $\{0, 1\}$, and condition (2) ensures that exactly one entry equals 1. If both checks pass, the servers learn only that $\mathbf{b}$ is valid, without revealing which entry is 1. If Bob is dishonest and provides $\mathbf{b} \notin W$, the DVV may leak limited information about $\mathbf{b}$, which is acceptable since Bob forfeits privacy when cheating.

Protocol 2 implements these ideas. After Alice and Bob distribute shares of their vectors (Lines 1–5), the servers validate Bob's input (Lines 6–17), and if validation succeeds, compute (D, D)-shares of the scalar product $\mathbf{a} \cdot \mathbf{b}$ and send them to Bob (Lines 18–21), who recovers his chosen message.

For general $k > 1$, one can run Protocol 2 k times, sharing Alice's vector only once (Lines 1–4 in Protocol 2). However, we describe a more communication-efficient alternative, Protocol 3. It parallels Protocol 2 with three main changes: (1) Bob's vector $\mathbf{b}$ is the sum of k unit vectors; (2) the DVV check is adapted by replacing the condition $\omega > 1$ with $\omega \neq k$; and (3) instead of computing a single scalar product, the servers compute and send shares of each product $a_n b_n$ for all $n \in [N]$. Bob reconstructs only the entries corresponding to his chosen indices.

Communication costs for Protocols 2 and 3, and for subsequent extensions (Priced OT, Generalized OT, and OMPE), are analyzed in Section 6.

Theorem 2. *Assume semi-honest servers with an honest majority (i.e., at most $t - 1$ colluding servers). Then Protocols 2 and 3 satisfy the following:*

Correctness. *If Alice and Bob follow the protocol, Bob obtains exactly the designated messages.*

Privacy. *The view of any coalition of Alice together with up to $t - 1$ servers (resp., Bob together with up to $t - 1$ servers) can be efficiently simulated from that party's input and output alone. In particular, Bob learns nothing about the remaining messages beyond those he is entitled to receive (resp., Alice learns nothing about Bob's choice indices).*

Protocol 2: 1-out-of-N Oblivious Transfer

Parameters: p—field size; N—number of messages; D—number of servers; $t = \lfloor (D+1)/2 \rfloor$.
Inputs: Alice has $U = \{m_1, \dots, m_N\}$; Bob has $j \in [N]$.

Alice sets $\mathbf{a} = (m_1, \dots, m_N)$.
Bob sets $\mathbf{b} = \mathbf{e}_j$.
forall $n \in [N]$ **do**
  Alice sends to M_d, $d \in [D]$, a (t, D)-share of a_n, denoted $[a_n]_d$.
  Bob sends to M_d, $d \in [D]$, a (t, D)-share of b_n, denoted $[b_n]_d$.
forall $1 \le n \le N$ **do**
  Each M_d, $d \in [D]$, sets $c_d = [b_n]_d \cdot ([b_n]_d - 1)$.
  The servers scramble $(c_1, \dots, c_D)$ and compute (D, D)-shares of $b_n(b_n - 1)$, denoted $(c'_1, \dots, c'_D)$.
  Each M_d broadcasts c'_d.
  The servers reconstruct $\eta := b_n(b_n - 1)$.
  if $\eta \neq 0$ **then**
    Abort
forall $d \in [D]$ **do**
  M_d computes $c_d \leftarrow \sum_{n \in [N]} [b_n]_d$.
The servers reconstruct $\omega := \sum_{n=1}^{N} b_n$ from any t shares.
if $\omega \neq 1$ **then**
  Abort
forall $d \in [D]$ **do**
  M_d computes $s_d \leftarrow \sum_{n \in [N]} ([a_n]_d \cdot [b_n]_d)$.
  M_d sends s_d to Bob.
Bob reconstructs $\mathbf{a} \cdot \mathbf{b} = m_j$.

Output: Bob obtains m_j.

Protocol 3: k-out-of-N Oblivious Transfer

Parameters: p—field size; N—number of messages; D—number of servers; $t = \lfloor (D+1)/2 \rfloor$.
Inputs: Alice has $U = \{m_1, \dots, m_N\}$; Bob has distinct $j_1, \dots, j_k \in [N]$.

Alice sets $\mathbf{a} = (m_1, \dots, m_N)$.
Bob sets $\mathbf{b} = (b_1, \dots, b_N)$, where $b_n = 1$ for $n \in \{j_1, \dots, j_k\}$ and $b_n = 0$ otherwise.
Alice and Bob distribute (t, D)-shares of $\mathbf{a}$ and $\mathbf{b}$.
The servers perform a DVV on $\mathbf{b}$'s shares (as in Protocol 2, with the condition $\omega \neq k$).
forall $d \in [D]$ **do**
  forall $n \in [N]$ **do**
    M_d computes $[c_n]_d \leftarrow [a_n]_d \cdot [b_n]_d$.
    M_d sends $[c_n]_d$ to Bob.
forall $n \in \{j_1, \dots, j_k\}$ **do**
  Bob reconstructs $c_n = a_n b_n = m_n$ from $\{[c_n]_1, \dots, [c_n]_D\}$.

Output: Bob obtains $m_{j_1}, \dots, m_{j_k}$.

Soundness against input-malicious Bob. *If Bob submits any vector* $\mathbf{b}$ *that does not satisfy the required form (either a unit vector for* OT_1^N *or a k-sparse* $\{0,1\}$*-vector for* OT_k^N*), then the DVV mechanism causes the protocol to abort. Hence, Bob cannot obtain any unauthorized linear combination or any message outside the designated set.*

Proof. We establish *correctness* and *statistical privacy* in the semi-honest, honest-majority setting by giving straight-line simulators for every admissible coalition.

Correctness. Protocol 2. Bob's vector is $\mathbf{b} = \mathbf{e}_j$, hence for each $n \neq j$ the shares $[b_n]_d$ form a (t, D)-sharing of 0, while for $n = j$ they form a (t, D)-sharing of 1. Each server computes

$$s_d \;=\; \sum_{n\in[N]} [a_n]_d \cdot [b_n]_d \;=\; [a_j]_d \cdot [b_j]_d \;+\; \sum_{n\neq j} [a_n]_d \cdot [b_n]_d.$$

For $n \neq j$, $[a_n]_d \cdot [b_n]_d$ is part of a (D, D)-sharing of 0 and cancels in reconstruction. The term $[a_j]_d \cdot [b_j]_d = f_j^A(d)\, f_j^B(d)$ is the dth evaluation of a degree-$\leq 2t-2$ polynomial with constant term a_j. Thus $\{s_d\}_{d\in[D]}$ are evaluations of a degree-$\leq 2t-2$ polynomial F with $F(0) = a_j = m_j$, and Bob interpolates F to obtain m_j.

Protocol 3. Bob's vector $\mathbf{b} \in \{0,1\}^N$ has Hamming weight k. For each $n \in [N]$ and server M_d, the value $[c_n]_d \leftarrow [a_n]_d[b_n]_d$ is a share of $a_n b_n$. If $b_n = 0$ then $\{[c_n]_d\}_d$ is a (D, D)-sharing of 0; if $b_n = 1$ then it is a (D, D)-sharing of a_n. Hence for $n \in \{j_1, \ldots, j_k\}$ Bob reconstructs $c_n = a_n = m_n$, and for $b_n = 0$ he reconstructs 0. The DVV checks ($b_n(b_n - 1) = 0$ for all n and $\sum_n b_n \in \{1, k\}$) succeed on honest inputs, so no abort occurs.

Privacy. We argue statistical privacy via ideal/real simulation. In all cases, simulated views depend only on the coalition's inputs and designated outputs, and match the real distribution due to the linearity and secrecy of Shamir sharing and the independence of fresh randomness (including scrambling).

(A) Privacy of Alice against Bob.

- Protocol 2. Bob receives $\{s_d\}_{d\in[D]}$, which are evaluations of a degree-$\leq 2t-2$ polynomial with constant term m_j and uniformly random higher coefficients independent of $\mathbf{a}$. A simulator $\mathsf{Sim}_B(j, m_j)$ samples a random degree-$(2t-2)$ polynomial F with $F(0) = m_j$ and outputs $s_d \leftarrow F(d)$. This perfectly matches the real view and hides $\{m_n\}_{n\neq j}$.
- Protocol 3. Bob receives, for each n, a (D, D)-sharing of $a_n b_n$. A simulator $\mathsf{Sim}_B(\{j_1, \ldots, j_k\}, \{m_{j_\ell}\})$ outputs, for $n \notin \{j_\ell\}$, a random (D, D)-sharing of 0, and for $n \in \{j_\ell\}$, a random (D, D)-sharing of m_n. This equals the real distribution and hides all $\{m_n\}$ with $b_n = 0$.

(B) Privacy of Bob against Alice. In both protocols Alice only sends her input shares and receives no server messages. A simulator $\mathsf{Sim}_A(\mathbf{a})$ outputs the shares Alice would generate; thus Alice learns nothing about Bob's indices beyond the fact that DVV succeeded.[2]

(C) Client–server collusions of size t.

- Alice with $(t-1)$ servers. The coalition knows $\mathbf{a}$ and at most $t-1$ shares of each b_n, which are statistically independent of $\mathbf{b}$ by t-privacy. During DVV, the coalition sees fewer than D shares of (D, D)-sharings of $b_n(b_n - 1)$ (after scrambling), hence cannot reconstruct them; the reconstructed sum $\omega = \sum_n b_n \in \{1, k\}$ is at most the Hamming weight, not the support. Therefore nothing about Bob's choice indices is learned.
- Bob with $(t-1)$ servers. In Protocol 2, the coalition knows $\mathbf{b}$ and at most $t-1$ shares of each a_n, plus $\{s_d\}$ distributed as in (A); it learns only m_j. In Protocol 3, the coalition additionally sees (D, D)-sharings of $a_n b_n$, which reveal exactly $\{m_{j_1}, \ldots, m_{j_k}\}$ and nothing about other a_n's; fewer than t shares of any a_n reveal nothing by t-privacy.

[2]DVV broadcasts occur among servers over the authenticated broadcast channel; clients do not receive these messages.

(D) Server privacy (for completeness). Any set of at most $t-1$ semi-honest servers alone sees at most $t-1$ shares of each value; by Shamir secrecy these are independent of the secrets. Scrambling re-randomizes (D, D)-sharings so local products do not leak information about the original shares.

Conclusion. For every admissible coalition there exists a straight-line simulator that reproduces its view from its inputs and designated outputs. Hence the real and ideal executions are statistically indistinguishable. Together with correctness, Protocols 2 and 3 are correct and statistically private for both parties under semi-honest servers with an honest majority. Furthermore, even if Alice (resp. Bob) colludes with any subset of $t-1$ servers, the coalition learns nothing beyond that party's input and designated output. □

4.1.1 An efficient DVV subprotocol. The most communication-intensive part of Protocols 2 and 3 is the verification that Bob's input vector is binary (Lines 6–12 in both protocols). This step incurs communication costs that scale linearly with N. To mitigate this, we propose an alternative, lightweight verification method described in Subprotocol 4.

Timing and randomness (clarification). The servers first agree on a pseudorandom number generator (PRNG) and jointly derive a shared seed σ *before Bob's shares are processed and without revealing the multipliers to Bob*. Concretely, each server M_d locally generates a random seed σ_d, and the shared seed is computed as $\sigma = \sum_{d\in[D]} \sigma_d \bmod p$. Using this common seed and the agreed-upon PRNG, all servers deterministically generate the same sequence of pseudorandom nonzero multipliers $(c_1, \ldots, c_N) \in (\mathbb{Z}_p^*)^N$ (Subprotocol 4, Lines 1-3). The multipliers are *not disclosed to clients*; Bob therefore cannot adapt his (secret-shared) selection vector to the multipliers.

Each server then computes a local share ω_d as defined in Line 4 in Subprotocol 4. These shares collectively form a (D, D)-sharing of the value $\omega := \sum_{n\in[N]} c_n \cdot b_n \cdot (b_n - 1)$ which the servers reconstruct in Line 5. If $\omega \neq 0$, then there exists at least one index $n \in [N]$ such that $b_n \notin \{0, 1\}$, implying that Bob's vector is not binary. In this case, the servers abort the protocol (Lines 6–7). Otherwise, they accept the input as binary.

This test introduces a negligible soundness error: a non-binary vector passes the test with probability at most $1/p$, where p is the field size. Since the multipliers (c_n) are fixed by the servers and *hidden from Bob at the time he submits his shares*, he cannot bias the outcome. Thus, the servers accept valid binary vectors with certainty and reject invalid ones with high probability.

While the DVV subprotocol in Protocols 2 and 3 is fully secure, Subprotocol 4 offers a dramatic reduction in communication cost and runtime, at the expense of a negligible error probability. We refer to the modified versions of Protocols 2 and 3, where the validation that Bob's vector is binary (Lines 6–12 in Protocol 2) is replaced with Subprotocol 4, as Protocols 2^* and 3^*, respectively.

Subprotocol 4: Efficient verification that Bob's input vector is binary

1 Each M_d locally generates a random seed σ_d, for $d \in [D]$.
2 The servers compute the shared seed $\sigma := \sum_{d\in[D]} \sigma_d \bmod p$.
3 Each M_d uses σ to compute $c_n \in \mathbb{Z}_p$ for all $n \in [N]$ (multipliers are internal to servers and not revealed to clients).
4 Each M_d broadcasts $\omega_d := \sum_{n\in[N]} c_n \cdot [b_n]_d \cdot ([b_n]_d - 1)$.
5 The servers reconstruct $\omega := \sum_{n\in[N]} c_n \cdot b_n \cdot (b_n - 1)$ from $\{\omega_d : d \in [D]\}$.
6 **if** $\omega \neq 0$ **then**
7 | **Abort**

Lemma 1. *Let p be prime and let $c_1, \ldots, c_N \xleftarrow{\$} \mathbb{Z}_p$ be jointly pseudorandom and independent of Bob's selection vector $\mathbf{b} \in \mathbb{Z}_p^N$ at the time $\mathbf{b}$ is secret-shared, as in Protocols 2* and 3*. Define $g_n := b_n(b_n - 1)$ for $n \in [N]$ and $\omega := \sum_{n=1}^{N} c_n g_n \in \mathbb{Z}_p$. Then Subprotocol 4 accepts exactly the binary vectors (i.e., $\mathbf{b} \in \{0,1\}^N$), and for any non-binary $\mathbf{b}$ we have $\Pr[\ \omega = 0\ ] = \frac{1}{p}$, where the probability is over the internal randomness that generates $(c_1, \ldots, c_N)$.*

For brevity, all remaining proofs are deferred to the Appendix. The proof of Lemma 1 is given in Appendix A.1.

Remark (Selective-failure security). All DVV checks, including the efficient test above, are completed *before* the servers send any message-dependent value to Bob. Consequently, a malicious receiver cannot submit different malformed inputs and infer information from whether the protocol aborts or proceeds. In particular, the continue/abort behavior of the DVV step cannot be used to extract any information about Alice's messages.

4.2 Priced Oblivious Transfer

Consider a setting of OT in which each of Alice's messages has a weight and the retrieval policy allows Bob to learn any subset of messages in which the sum of weights does not exceed some given threshold. For example, if Alice holds a database of movies and each movie has a price tag, then if Bob had prepaid some amount, Alice wishes to guarantee that he retrieves movies of aggregated cost that does not exceed what he had paid, while Bob wishes to prevent Alice from knowing what movies he chose to watch. Such a setting is referred to as Priced OT [1].

Definition 4. *Let $U = \{m_1, \ldots, m_N\}$ be the set of messages that Alice has. Assume that each message m_n has a weight $w_n \geq 0$, $n \in [N]$, and let $T > 0$ be some given threshold. Then a Priced OT protocol allows Bob to retrieve any subset $B \subseteq U$ for which $\sum_{m_n \in B} w_n \leq T$. Bob cannot learn any information on the messages in $U \setminus B$, while Alice has to remain oblivious of Bob's choice.*

We assume that the weights $w_1, \ldots, w_N$ are publicly known, since they represent information that is supposed to be known to all. The threshold T, on the other hand, represents the amount that Bob had paid and, therefore, it is private and should remain so.

k-out-of-N OT is a special case of Priced OT, where $w_n = 1$ for all $n \in [N]$ and $T = k$. Hence, Priced OT can be implemented by Protocol 3 with the following modifications: (a) Alice and Bob send to the servers (t, D)-secret shares of T, and the servers verify that they received secret shares of the same value, without revealing it; and (b) in validating Bob's vector, instead of checking that $\sum_n b_n = k$, the servers will verify that $\sum w_n b_n \leq T$, where the weights w_n, $n \in [N]$, are publicly known.

Checking inequalities between shared secrets can be carried out by secure comparison protocols, e.g. [27]. However, as secure comparison is significantly more costly than testing equality to a public value, we proceed to suggest a simple manner in which the servers can verify the inequality $\sum w_n b_n \leq T$ without invoking a costly secure comparison protocol.

Let $\ell := \lfloor \log_2 T \rfloor + 1$. Hence, the difference $e := T - \sum_n w_n b_n$ is smaller than 2^ℓ. Therefore, if Alice adds to her list of messages ℓ phantom messages with weights 2^i, $0 \leq i \leq \ell - 1$, Bob can add to his list of requested (genuine) messages a subset of the phantom messages for which the sum of weights is e. Bob will ignore the phantom messages later on, but by adopting such a practice, the sum of weights of messages that Bob selects becomes exactly T. Hence, each server M_d, $d \in [D]$, may compute locally and then broadcast

$$[e]_d \leftarrow [T]_d - \sum_{n \in [N]} w_n [b_n]_d \,.$$

Subsequently, the servers will use any t shares out of $\{[e]_1, \ldots, [e]_D\}$ to compute $e = T - \sum_{n\in[N]} w_n b_n$. They will proceed to serve Bob's request only if $e = 0$.

The security of this extension follows directly from Theorem 2, as the only modification is replacing the cardinality check with an affine check on secret-shared values. A proof sketch appears in Appendix A.2.

A concluding comment. In practical deployments of Priced OT, Bob will typically not exhaust his entire budget T in a single retrieval. Thus, it is desirable to support *dynamic budgets*, whereby Bob may retrieve a subset of messages of total weight less than T and subsequently have his remaining budget updated, or alternatively be able to purchase additional budget to increase T. Such dynamic budget management can be realized by combining Priced OT with a secure comparison protocol, ensuring that budget updates preserve privacy. The resulting extension follows standard techniques and is omitted for brevity.

4.3 Generalized Oblivious Transfer

Ishai and Kushilevitz [18] introduced an extension of oblivious transfer (OT) called *Generalized Oblivious Transfer* (GOT). As in the basic OT setting (Definition 2), we consider two parties, Alice and Bob. Alice holds a collection of N messages $m_1, \ldots, m_N$, each an element of $\mathbb{Z}_p$. Bob wishes to obtain a subset of these messages according to a prescribed retrieval policy. In OT_k^N, the policy restricts Bob to retrieve any subset of at most k messages. GOT generalizes this notion by allowing an arbitrary policy specified by an *access structure*.

Intuition. In a generalized OT, Bob is permitted to retrieve any subset of messages belonging to a family of authorized sets. To enforce this restriction, Alice selects a random secret s and applies a secret-sharing scheme whose access structure is *induced by* the retrieval policy, ensuring that Bob can reconstruct s exactly when his chosen subset is authorized. Bob can reconstruct s, and hence proceed, if and only if his requested subset is authorized.

Definition 5. *Let $U = \{m_1, \ldots, m_N\}$ denote Alice's set of messages. An* access structure *is a collection $\mathcal{A} \subseteq 2^U$ that is* monotone decreasing, *i.e., if $B \in \mathcal{A}$ and $B' \subset B$, then $B' \in \mathcal{A}$. The* basis *of $\mathcal{A}$, denoted $\mathcal{A}^0$, is the collection of all maximal subsets in $\mathcal{A}$: $B \in \mathcal{A}^0$ iff $B \in \mathcal{A}$ and, for every $B \subsetneq B' \subseteq U$, it holds that $B' \notin \mathcal{A}$.*

Bob may retrieve any subset $B \subseteq U$ such that $B \in \mathcal{A}$. As in standard OT, Bob learns nothing about the remaining messages $U \setminus B$, while Alice remains oblivious to Bob's choice.

Our distributed GOT protocol, denoted Protocol 5, follows the construction of Tassa [31] and builds upon the OT_k^N protocol (Protocol 3). We first describe the case of *uniform bases*, where all subsets in $\mathcal{A}^0$ have the same cardinality k. The general (non-uniform) case can be reduced to the uniform one, as shown in [31].

Define the *monotone increasing closure* of $\mathcal{A}^0$ as

$$\Gamma = \Gamma(\mathcal{A}^0) = \{\, C \subseteq U \;:\; \exists B \in \mathcal{A}^0,\ B \subseteq C \,\}. \tag{2}$$

The collection Γ is monotone increasing: if $B \in \Gamma$ and $B \subset B' \subseteq U$, then $B' \in \Gamma$. Let Σ be a secret-sharing scheme that realizes Γ in the following sense: the participants are the messages U, and the access structure is Γ. Given a secret s, the scheme Σ assigns to each message $m_n \in U$ a share s_n such that the shares of any subset in Γ reconstruct s, whereas any other subset reveals no information about s. It is known that any monotone increasing access structure admits a realizing secret-sharing scheme, e.g., [4].

Protocol overview. Protocol 5 proceeds as follows. Alice samples a secret $s \in \mathbb{Z}_p$ (Line 1) and computes its shares according to the access structure $\Gamma(\mathcal{A}^0)$ (Line 2). In particular, if $B = \{m_{j_1}, \dots, m_{j_k}\} \in \mathcal{A}^0$, then the shares $\{s_{j_1}, \dots, s_{j_k}\}$ reconstruct s; otherwise, they reveal nothing about it. Alice then distributes (t, D)-shares of s to the servers (Line 3).

Next, Alice distributes to the servers (t, D)-shares of two private vectors: (i) the vector of secret-sharing shares (Line 4) and (ii) the vector of private messages (Line 5). At this point, Alice's role is complete; the remainder of the execution involves Bob and the servers.

Bob and the servers engage in a k-out-of-N OT to retrieve the k shares corresponding to Bob's selected messages (Lines 6-8). Thus, Bob obtains the shares $s_{j_1}, \dots, s_{j_k}$ but not the messages themselves. He reconstructs $s^B := s$ via the reconstruction procedure of Σ (Line 9) and distributes (t, D)-shares of s^B to the servers (Line 10). The servers then verify that $s^B = s^A$ without learning the value of s (Lines 11–12). If the difference $e = s^A - s^B$ is nonzero, the protocol aborts (Line 13); otherwise, the servers are convinced that Bob's choice corresponds to an authorized subset and complete the OT_k^N using the shares of the actual messages (Line 14), allowing Bob to obtain his selected messages.

Security and correctness follow from Theorem 2 and from the proof sketch in Appendix A.3.

Non-ideal access structures. We have so far assumed that the access structure $\Gamma(\mathcal{A}^0)$, defined in (2), is *ideal*, meaning that there exists a secret-sharing scheme Σ realizing it with all shares $s_1, \dots, s_N$ taken from the same field $\mathbb{Z}_p$ as the secret s. If $\Gamma(\mathcal{A}^0)$ is non-ideal, or if one employs a non-ideal scheme Σ that is simpler to implement,[3] then the shares $s_1, \dots, s_N$ may reside in a larger field $\mathbb{Z}_q$ for some prime $q \geq p$. In that case, Protocol 5 proceeds unchanged, except that the invocation of Protocol 3 with the vector $\mathbf{a} = (s_1, \dots, s_N)$ is executed over $\mathbb{Z}_q$.

Protocol 5: Generalized Oblivious Transfer

Parameters: p—field size; N—number of messages; D—number of servers; $t = \lfloor (D+1)/2 \rfloor$.
Inputs: Alice has $U = \{m_1, \dots, m_N\} \subset \mathbb{Z}_p$ and an access structure $\mathcal{A}$ on U, with a k-uniform basis $\mathcal{A}^0$; Bob has distinct $j_1, \dots, j_k \in [N]$, where $B := \{m_{j_1}, \dots, m_{j_k}\} \in \mathcal{A}^0$.

1 Alice selects uniformly at random a secret $s \in \mathbb{Z}_p$.
2 Alice computes shares $\{s_1, \dots, s_N\} \subset \mathbb{Z}_p$ of s using a secret sharing scheme Σ that realizes the access structure $\Gamma(\mathcal{A}^0)$ on U.
3 Alice distributes to the servers (t, D)-shares of $s^A := s$; M_d's share is denoted $[s^A]_d$, $d \in [D]$.
4 Alice sets $\hat{\mathbf{a}} = (s_1, \dots, s_N)$ and then distributes to the servers (t, D)-shares of $\hat{\mathbf{a}}$.
5 Alice sets $\mathbf{a} = (m_1, \dots, m_N)$ and then distributes to the servers (t, D)-shares of $\mathbf{a}$.
6 Bob sets $\mathbf{b} = \sum_{i=1}^{k} \mathbf{e}_{j_i}$ and then distributes to the servers (t, D)-shares of $\mathbf{b}$.
7 The servers execute the DVV mechanism on $\mathbf{b}$'s shares, as in Protocol 3, and abort if the verification fails.
8 Bob and the servers perform Lines 5-10 in Protocol 3, using the shares of $\hat{\mathbf{a}}$ that were distributed in Line 4 above.
9 Bob recovers $s^B := s$ from $\{s_{j_1}, \dots, s_{j_k}\}$, using the reconstruction function of the secret sharing scheme Σ.
10 Bob distributes to the servers (t, D)-shares of the secret s^B that he had computed above; M_d's share is denoted $[s^B]_d$, $d \in [D]$.
11 M_d, for all $d \in [D]$, computes $[e]_d = [s^A]_d - [s^B]_d$.
12 The servers recover $e := s^A - s^B$ from any t shares out of $\{[e]_d : d \in [D]\}$.
13 **if** $e \neq 0$ **then** Abort.
14 Bob and the servers perform Lines 5-10 in Protocol 3, using the shares of $\mathbf{a}$ that were distributed in Line 5 above.
Output: Bob gets $m_{j_1}, \dots, m_{j_k}$.

[3] A non-ideal scheme may sometimes offer a more efficient or practical realization of the same access structure.

5 Oblivious Polynomial Evaluation

The oblivious polynomial evaluation (OPE) problem was presented in [23], and extended to multivariate polynomials in [35]. We devise herein a distributed protocol for the multivariate case.

We begin by defining multivariate polynomials (Definitions 6 and 7) and then define the corresponding MPC problem (Definition 8).

Definition 6. *(Monomial) Let $\mathbb{Z}_p$ be a finite field, $\mathbf{x} = (x_1, \ldots, x_k)$ a k-dimensional vector over $\mathbb{Z}_p$, and $\mathbf{j} = (j_1, \ldots, j_k)$ a k-dimensional vector of nonnegative integers. Then the monomial $\mathbf{x}^{\mathbf{j}}$ is defined as $\mathbf{x}^{\mathbf{j}} := \prod_{i=1}^{k} x_i^{j_i}$.*

Definition 7. *(Multivariate Polynomial) Let $\mathbb{Z}_+^k := \{\mathbf{j} = (j_1, \ldots, j_k) : j_i \in \mathbb{Z}_+ = \{0, 1, 2, \ldots\}\}$, and let $\mathbb{Z}_+^{k,N} := \{\mathbf{j} \in \mathbb{Z}_+^k : |\mathbf{j}| := \sum_{i=1}^{k} j_i \leq N\}$. An N-degree k-variate polynomial $f(\mathbf{x})$ over $\mathbb{Z}_p$, where $\mathbf{x} = (x_1, \ldots, x_k) \in \mathbb{Z}_p^k$, is defined as*

$$f(\mathbf{x}) = \sum_{\mathbf{j} \in \mathbb{Z}_+^{k,N}} a_{\mathbf{j}} \cdot \mathbf{x}^{\mathbf{j}}, \qquad a_{\mathbf{j}} \in \mathbb{Z}_p. \tag{3}$$

Definition 8. *(OMPE) Assume that Alice has an N-degree multivariate polynomial $f(\mathbf{x}) = f(x_1, \ldots, x_k)$, and Bob has a point $\alpha = (\alpha_1, \ldots, \alpha_k) \in \mathbb{Z}_p^k$. They wish to enable Bob to learn $f(\alpha)$ and nothing else about f, while keeping Alice oblivious to α.*

OMPE can be solved by reducing it to the distributed scalar product (DSP) functionality, along with the required validation checks. Alice submits the vector of coefficients

$$\mathbf{a} = (a_{\mathbf{j}} : \mathbf{j} \in \mathbb{Z}_+^{k,N}),$$

while Bob submits

$$\mathbf{b} = (b_{\mathbf{j}} : \mathbf{j} \in \mathbb{Z}_+^{k,N}), \qquad b_{\mathbf{j}} := \alpha^{\mathbf{j}}. \tag{4}$$

The dimension of these vectors is $\binom{N+k}{k}$.

To ensure correct pairing of coefficients and powers, the parties agree in advance on an ordering of $\mathbb{Z}_+^{k,N}$. We group monomials into $N+1$ tiers: $T_0 := \mathbb{Z}_+^{k,0}$, and for $n = 1, \ldots, N$, let $T_n := \mathbb{Z}_+^{k,n} \setminus \mathbb{Z}_+^{k,n-1}$. Monomials within each tier are ordered lexicographically.

Protocol 6 begins with Alice and Bob forming their vectors and sharing them (Lines 1–5). The first entry in $\mathbf{b}$ equals 1 for $\mathbf{j} = (0, \ldots, 0)$; hence the servers may set that share locally. The servers then validate that $\mathbf{b}$ corresponds to a consistent evaluation of some α (Lines 6–11). The following lemma captures the correctness of this check. (The proof is given in Appendix A.4.)

Lemma 2. *The vector $\mathbf{b} = (b_{\mathbf{j}})_{\mathbf{j} \in \mathbb{Z}_+^{k,N}}$, with $b_{(0,\ldots,0)} = 1$, is of the form (4) if and only if $\omega = 0$ in every iteration of the validation loop in Lines 6–11 of Protocol 6.*

Finally, the servers compute shares of the scalar product and send them to Bob (Lines 12–14), who reconstructs $f(\alpha)$ (Line 15).

Example. For $k = 2$ and $N = 2$ we have $\mathbf{b} = (1, \alpha_1, \alpha_2, \alpha_1^2, \alpha_1\alpha_2, \alpha_2^2)$. Validation checks that $b_{(2,0)} = \alpha_1^2$, $b_{(0,2)} = \alpha_2^2$, and $b_{(1,1)} = \alpha_1\alpha_2$ by verifying $b_{(2,0)} = b_{(1,0)}^2$, $b_{(0,2)} = b_{(0,1)}^2$, and $b_{(1,1)} = b_{(1,0)}b_{(0,1)}$.

The security guarantees of Protocol 6 follow from Theorem 2, see a proof sketch in Appendix A.5.

Protocol 6: Oblivious Multivariate Polynomial Evaluation

Parameters: p—field size; k—number of variables; N—polynomial degree; D—number of servers; $t = \lfloor (D+1)/2 \rfloor$.

Inputs: Alice has a secret N-degree k-variate polynomial $f(\mathbf{x})$ (Eq. (3)); Bob has a secret point $\alpha \in \mathbb{Z}_p^k$.

Alice sets $\mathbf{a} = (a_{\mathbf{j}})_{\mathbf{j} \in \mathbb{Z}_+^{k,N}}$.

Bob sets $\mathbf{b} = (b_{\mathbf{j}} = \alpha^{\mathbf{j}})_{\mathbf{j} \in \mathbb{Z}_+^{k,N}}$.

forall $\mathbf{j} \in \mathbb{Z}_+^{k,N}$ **do**

  Alice sends (t, D)-shares of $a_{\mathbf{j}}$ to the servers.

  Bob sends (t, D)-shares of $b_{\mathbf{j}}$ to the servers.

forall $2 \le n \le N$ **do**

  forall $\mathbf{j} \in T_n$ **do**

    Select $\mathbf{h} \in T_{n-1}$ and $1 \le i \le k$ such that $\mathbf{j} = \mathbf{h} + \mathbf{e}_i$.

    Servers compute $\omega := b_{\mathbf{h}} \cdot b_{\mathbf{e}_i} - b_{\mathbf{j}}$.

    if $\omega \neq 0$ **then**

      Abort.

forall $d \in [D]$ **do**

  M_d computes $s_d \leftarrow \sum_{\mathbf{j}} [a_{\mathbf{j}}]_d \cdot [b_{\mathbf{j}}]_d$.

  M_d sends s_d to Bob.

Bob reconstructs $f(\alpha) = \mathbf{a} \cdot \mathbf{b}$.

Output: Bob learns $f(\alpha)$.

6 Communication Complexity

Communication is the primary performance bottleneck in distributed cryptographic protocols, especially in multi-server settings. We therefore quantify the communication cost of all our protocols. Communication is measured in field elements over $\mathbb{Z}_p$, each represented using $\lceil \log p \rceil$ bits. We distinguish three communication sources:

- $\mathrm{Com}_{\mathrm{AM}}$: messages between Alice and the servers;
- $\mathrm{Com}_{\mathrm{BM}}$: messages between Bob and the servers;
- $\mathrm{Com}_{\mathrm{MM}}$: messages among servers.

Servers communicate only via broadcast; a broadcast of L field elements counts as $L(D-1)$ units in $\mathrm{Com}_{\mathrm{MM}}$, where D is the number of servers.

Complexity highlight. Our primary asymptotic savings occur in the server-side communication for distributed verification of Bob's selection vector (DVV). The optimized DVV replaces an $O(ND^2)$ server communication term in OT_1^N and OT_k^N with an $O(\theta D^2)$ term, where θ is the seed length in field elements (e.g., $\theta \approx 3$–4 for 128–160 bits at typical field sizes). Thus, for large N, the improvement is asymptotically and practically substantial.

Table 1 summarizes the communication for all protocols. Detailed derivations are provided in Appendix B.

7 Experiments

7.1 Implementation details

We implemented our protocols in Java (OpenJDK 17) using the Eclipse environment on a Lenovo Ideapad Gaming 3 laptop equipped with an AMD Ryzen 7 5800H CPU and 16 GB RAM, running Windows 11 64-bit. The machine was

Problem	Protocol	Com_{AM}	Com_{BM}	Com_{MM}
SP	1	$(N+1)D$	$(N+1)D$	0
OT_1^N	2	ND	$(N+1)D$	$(2N+1)D(D-1)$
	2^*	ND	$(N+1)D$	$(\theta+2)D(D-1)$
OT_k^N	3	ND	$2ND$	$(2N+1)D(D-1)$
	3^*	ND	$2ND$	$(\theta+2)D(D-1)$
Priced OT	3 modified	$(N+1)D$	$(2N+1)D$	$2(N+1)D(D-1)$
	3^* modified	$(N+1)D$	$(2N+1)D$	$(\theta+3)D(D-1)$
Generalized OT	5	$(2N+1)D$	$(3N+1)D$	$2(N+1)D(D-1)$
	5^*	$(2N+1)D$	$(3N+1)D$	$(\theta+3)D(D-1)$
OMPE	6	N_kD	$(N_k+1)D$	$(N_k-k-1)D(D-1)$

Table 1. Communication costs of all distributed protocols with D servers. N is the vector length (and polynomial degree in OMPE). k is the selection parameter in OT_k^N and the number of variables in OMPE, where $N_k = \binom{N+k}{k}$. θ is the seed length in field elements in Subprotocol 4.

plugged into AC power during all experiments to avoid energy-saving performance scaling. All experiments were executed in a single thread, running one protocol instance at a time, in order to isolate the computational cost of the cryptographic operations and avoid confounding effects due to thread scheduling or parallel execution overhead.

A 64-bit prime p was chosen at random as the modulus of the field $\mathbb{Z}_p$, and modular arithmetic was implemented using Java's `BigInteger` class. Random values were generated using the default secure pseudorandom generator provided by the Java standard library. The source code is publicly available at https://github.com/b1086960/Distributed_OT_OPE.

We emphasize that these measurements focus on computation only; communication costs are derived analytically in Section 6. Evaluating full distributed system performance and network behavior is beyond the scope of this study and is left for future work.

The system configuration is summarized in Table 2.

Component	**Specification**
CPU	AMD Ryzen 7 5800H
RAM	16 GB
Operating System	Windows 11 (64-bit)
Java Version	`OpenJDK 17`
IDE / Build Environment	Eclipse Workspace
Execution Mode	Single-threaded, AC power enabled
Arithmetic Library	Java `BigInteger`
Randomness	`SecureRandom` (default provider)

Table 2. Experimental environment.

7.2 Results

We evaluated the protocols' runtimes as a function of the vector dimension N and the number of servers D. The tested values were $N \in \{10^2, 10^3, 10^4, 10^5, 10^6\}$ and $D \in \{3, 5, 7, 9, 11\}$. To test dependence on N we fixed $D = 7$, and to test dependence on D we fixed $N = 10^6$. Due to the wide range of N values, the plots for varying N use a logarithmic scale.

All experiments were conducted on randomly generated vectors (or message sets / polynomials). Each experiment was repeated 50 times, and we report the average runtime for Alice, Bob, and the servers (with server times averaged over the D servers and ten trials). Standard deviations are omitted as they were negligible and visually indistinguishable in all experiments.

7.2.1 Distributed scalar product. In the first experiment, we tested our basic protocol that solves DSP (Protocol 1). Figure 1 shows the runtimes for Alice, Bob, and the average runtime for the servers as a function of N. The results confirm the expected linear dependence of the runtime on N.

Figure 2 displays the runtimes as a function of D. The growth for Alice and Bob is consistent with the theoretical $O(D^2)$ computational complexity, as each party performs D evaluations of a polynomial of degree $(D-1)/2$. Minor deviations in this small-parameter regime are attributable to fixed overheads and constant-factor effects. In contrast, the servers' computation cost remains essentially constant, fluctuating between 125 and 150 milliseconds for all tested values of D.

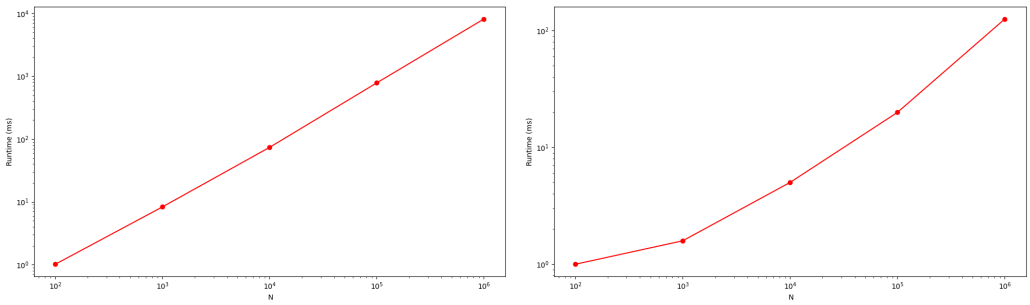

Fig. 1. Runtimes (milliseconds) for Protocol 1 (DSP) as a function of N (log scale), for $D = 7$. The left plot shows the runtimes for Alice and Bob; the right plot shows the average runtimes for the servers.

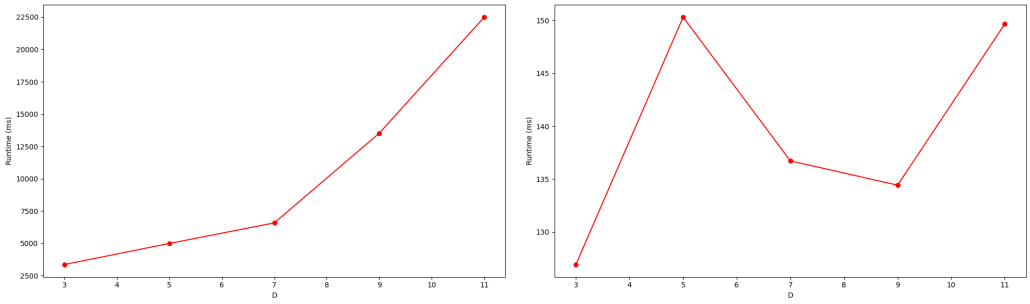

Fig. 2. Runtimes (milliseconds) for Protocol 1 (DSP) as a function of the number of servers D, for $N = 10^6$. The left plot shows the runtimes for Alice and Bob; the right plot shows the average runtime for the servers.

7.2.2 Oblivious transfer. We next evaluate Protocol 3 for the OT_k^N functionality. We focus on the servers' computation, since Bob's work in this protocol matches that of Protocol 1, and Alice performs the same operations as in the initialization phase of Protocol 1. The resulting server runtimes are shown in Figure 3.

The left panel shows the runtime as a function of N (with $D = 7$ and $k = 10$), demonstrating a clear linear scaling in N. In contrast to Protocol 1, where the server cost is independent of D, the right panel illustrates that here the runtime grows with D; this effect stems from the DVV component of the protocol.

As predicted by the analysis, the cost is essentially independent of k, hence we omit runtime plots for different values of k.

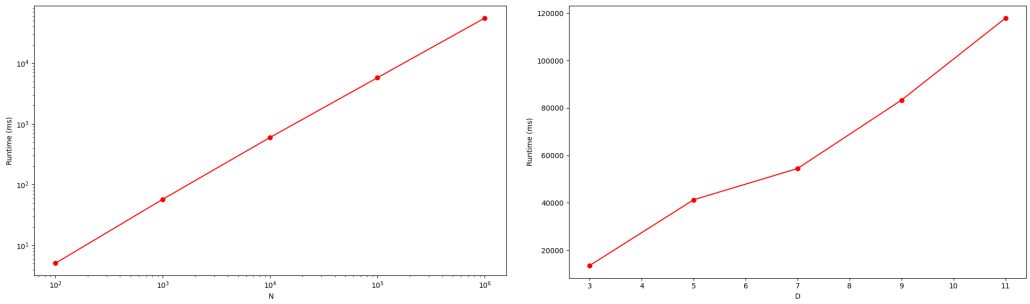

Fig. 3. Average server runtimes (milliseconds) for Protocol 3 (OT_k^N). Left: runtime (log scale) as a function of N, for $D = 7$ and $k = 10$. Right: runtime as a function of D, for $N = 10^6$ and $k = 10$.

We further evaluated Protocol 3*, where the DVV subroutine is replaced by Subprotocol 4. In this variant, each server begins with a 160-bit seed σ (see Lines 1–2 in Subprotocol 4), and each fresh field element is derived by updating $\sigma \leftarrow \text{SHA-3}(\sigma)$ and reducing $\sigma \bmod p$ (Line 3).

The resulting runtimes appear in Figure 4. Consistent with the analysis, the optimized variant significantly reduces server computation time, with improvements reaching up to two orders of magnitude in our experiments.

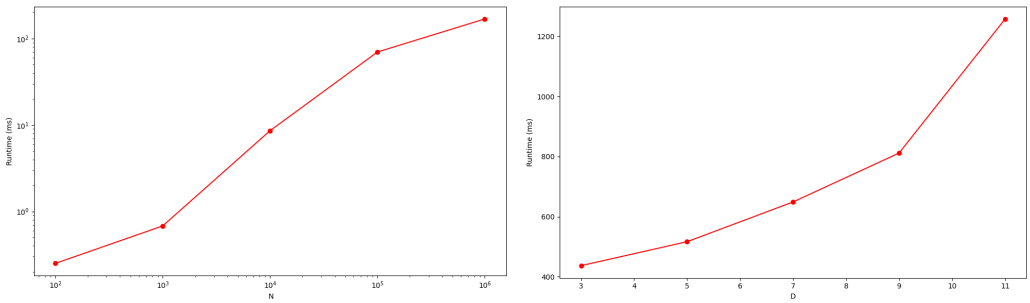

Fig. 4. Average server runtimes (milliseconds) for Protocol 3* (OT_k^N), where DVV is implemented via Subprotocol 4. Left: runtime (log scale) as a function of N, for $D = 7$ and $k = 10$. Right: runtime as a function of D, for $N = 10^6$ and $k = 10$.

7.2.3 Priced OT. We simulated the Priced OT protocol, where message weights were selected uniformly at random from $1, 2, \ldots, 10$ and the threshold was set to $T = 100$. Similar to the baseline experiments, we focus on the servers' average runtime as a function of N and D, ignoring client-side cost.

The resulting performance trends are nearly identical to those in Figures 3 and 4, with only a negligible constant-factor overhead ($< 2\%$ in all measurements). For brevity, we omit the plots. This similarity confirms that pricing introduces no significant computational overhead.

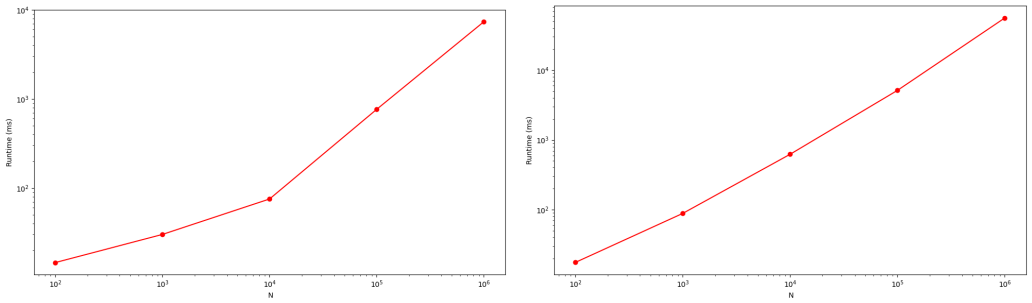

Fig. 5. Runtimes (milliseconds) for Protocol 5 (GOT)), in the case of compartmented access structures as a function of N, for $D = 7$. The left plot shows the runtimes for Bob; the right plot shows the average runtimes for the servers. The runtimes are presented on a logarithm scale.

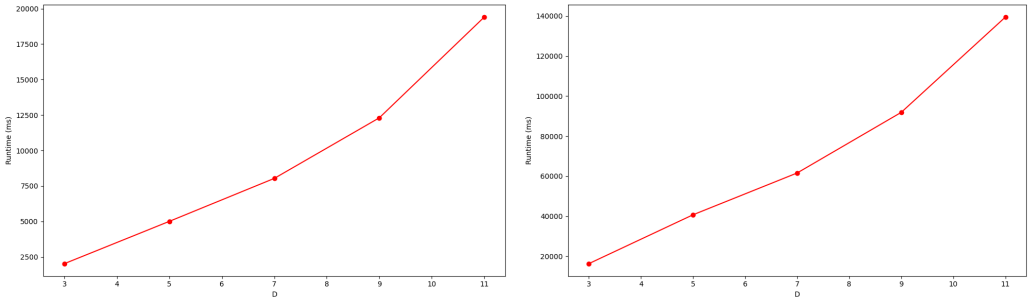

Fig. 6. Runtimes (milliseconds) for Protocol 5 (GOT)), in the case of compartmented access structures as a function of D, for $N = 10^6$. The left plot shows the runtimes for Bob; the right plot shows the average runtimes for the servers.

7.2.4 Generalized OT. We evaluate Protocol 5 (Generalized OT) under a compartmented access structure. Specifically, the message universe is partitioned into $r > 1$ disjoint compartments of equal size, $U = \biguplus_{i=1}^{r} U_i$, where $|U_i| = k = 10$, for all $1 \le i \le r$, and Bob is allowed to retrieve messages from only one compartment.

The corresponding monotone access structure Γ (Eq. (2)) is realized using a simple additive secret sharing scheme Σ applied independently within each compartment: each share $s_1, \ldots, s_N$ is chosen uniformly at random, and the sum of the shares associated with messages in a given compartment U_i equals the secret s (see Lines 1–2 of Protocol 5).

Figures 5 and 6 report the runtime of Bob and the servers as a function of N and D. Additionally, Figure 7 shows the average server runtime for the optimized variant Protocol 5*, where the DVV step is implemented using the efficient Subprotocol 4.

7.2.5 Oblivious multivariate polynomial evaluation. Here we evaluate Protocol 6 (OMPE). We ran that protocol with random polynomials of degrees $N \in \{5, 10, 20, 30, 40, 50\}$, where the number of variables was set to $k = 3$ — see Figure 8. The shown runtimes grow linearly with $\binom{N+k}{k}$, since that is the size of the two vectors in the scalar product.

8 The Case of Malicious Servers

The protocols evaluated experimentally in Section 7 operate in the semi-honest model. In this section we show how they can be upgraded to withstand *malicious* servers. These extensions do not change the communication pattern and

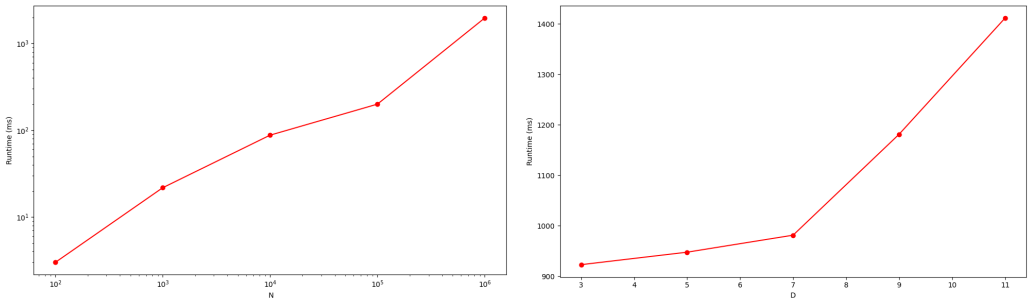

Fig. 7. Average server runtimes (milliseconds) for Protocol 5* (GOT), in the case of compartmented access structures, where DVV is implemented via Subprotocol 4. Left: runtime (log scale) as a function of N, for $D = 7$. Right: runtime as a function of D, for $N = 10^6$.

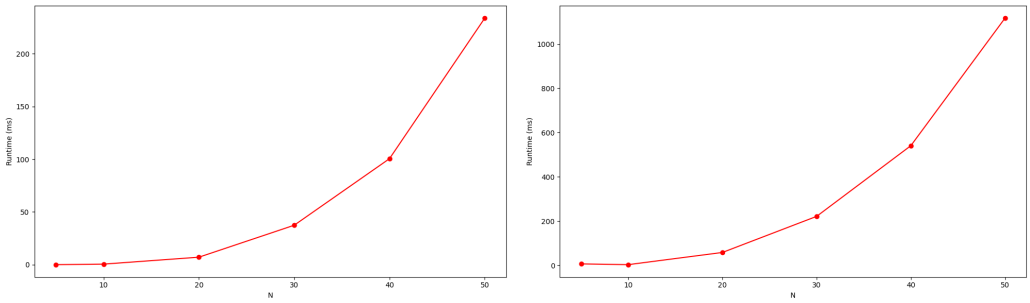

Fig. 8. Runtimes (milliseconds) for Protocol 6 (OMPE), as a function of N, the polynomial degree, for $k = 3$. Left: runtimes for Bob. Right: average runtimes for the servers.

rely only on standard VSS and BGW components; therefore, we present them as a theoretical enhancement without separate performance benchmarking.

Throughout this section, we write m for the number of malicious servers (deviating arbitrarily from the protocol). Note that in the semi-honest setting discussed earlier, security relied only on the sharing threshold t, which ensured that any coalition of up to $t - 1$ (semi-honest) servers learns nothing. Here, by contrast, we consider fully malicious deviations.

Adversarial model. Alice and Bob are assumed honest with respect to *dealing* their shares, and hence we do not require verifiability of their share generation.[4] However, Bob may still behave *maliciously in his choice of input*: he may attempt to submit an *illegal* selection vector, even if perfectly secret-shared. This "input malice" is precisely what our distributed vector validation (DVV) mechanism prevents. Our primary concern here is malicious servers: up to m servers may deviate arbitrarily (e.g., send malformed shares or inconsistent openings). Our security goals are: (i) **correctness** (honest parties reconstruct the correct output), (ii) **privacy** (no leakage beyond the ideal functionality), and (iii) **robustness** (detect and abort upon inconsistency).

[4] Alice has no incentive to cheat in the share–generation stage: if she wished to sabotage the protocol, she could simply supply an incorrect input vector or refuse to participate. Bob likewise has no incentive to distribute malformed shares of his input, since he is the only party who expects an output and improper sharing would only harm his own ability to reconstruct it.

Building on the guarantees proven in the semi-honest setting, we now extend our protocols to handle malicious servers. The malicious hardening touches only the secret-sharing layer (dealing and reconstruction) and does not alter the high-level information flow of DSP/OT/OPE. For concreteness we present the changes for Protocol 2*, where Lines 6–12 are replaced by Subprotocol 4 (DVV). The same techniques apply verbatim to the other OT and OMPE protocols.

Applicability. The same secret-sharing hardening applies unchanged to Priced OT, Generalized OT, and OMPE: only the sharing/reconstruction layers change, not the high-level message pattern (see Sections 6 and 7).

Before describing the cryptographic enhancements for malicious servers, we summarize the achievable security guarantees under different trust models. Table 3 highlights the maximum tolerated adversarial participation under semi-honest and malicious settings.

Adversary Type	Curruption Threshold	Protection Mechanism
Semi-honest collusion	$c_{\text{sh}} < D/2$	Shamir SS
Malicious servers (light)	$m < D/4$	Shamir SS + RS decoding
Malicious servers (full)	$m < D/3$	BGW + VSS + RS decoding

Table 3. Security regimes under different adversarial models. Note the different symbols: c_{sh} bounds *semi-honest* collusions in the main protocols, whereas m bounds *malicious* servers in Section 8.

8.1 Verifiable Secret Sharing in Brief

We recall the two classical VSS concerns [11]: (a) a malicious dealer may distribute shares *not* lying on a degree-$(t-1)$ polynomial; (b) malicious share-holders may submit *wrong* shares at reconstruction so that distinct t-subsets interpolate to different secrets. For (b), let $f(x) = \sum_{i=0}^{t-1} a_i x^i$ and consider the Reed–Solomon code $\mathcal{C} = \{(f(1), \dots, f(D)) : (a_0, \dots, a_{t-1}) \in \mathbb{Z}_p^t\}$. Minimum distance $D - t + 1$ implies that, if all D shares are broadcast at reconstruction, up to $(D-t)/2$ errors can be corrected to recover $f(0)$ via nearest-codeword decoding [37]. For (a), a standard fix uses *bivariate* VSS: the dealer samples $S(x, y)$ of bidegree $(t-1, t-1)$ with $S(0,0) = s$, sends $f_d(x) = S(x, d)$ and $g_d(y) = S(d, y)$ to party M_d, and parties cross-check $f_d(d') = g_{d'}(d)$ for all pairs [5, Section 5]. We will rely on these textbook tools in our constructions below. Full algebraic details are moved to Appendix C.

We conclude by noting that VSS techniques rely on the availability of private point-to-point channels between every pair of parties. Therefore, to support the malicious-secure enhancements described below, we assume that the servers are equipped with private pairwise communication channels, in addition to the authenticated broadcast channel assumed in Section 2.

8.2 The Case $m < \frac{D}{4}$: Lightweight Robustness

In Protocol 2*, only Alice and Bob *deal* secrets (Lines 4–5); servers only reconstruct intermediate values (Line 5 of Subprotocol 4 and Line 15 of the main protocol), and Bob reconstructs the final output (Line 21). We *do not* VSS-check Alice or Bob as dealers (they are honest dealers here), and we continue to treat Bob as potentially *malicious in his input*; DVV already enforces that his selection vector is binary and well-formed.

Modifications.

(1) Use (t, D)-sharing with $t = \lceil D/4 \rceil$ in Lines 4–5 (instead of $t = \lfloor (D+1)/2 \rfloor$, cf. Eq. (1)).

(2) For ω at Line 15, interpolate from *all* D shares; if the interpolated degree exceeds $t-1$, apply RS decoding to correct up to $(D-t)/2$ adversarial shares.
(3) For ω at Line 5 of Subprotocol 4, where the target degree is $2t-2$, correct up to $\big(D-(2t-1)\big)/2$ errors via RS decoding.
(4) For Bob's final reconstruction at Line 21 (degree $2t-2$), apply the same RS decoding bound as in item 3.

Theorem 3. *If $m < \frac{D}{4}$, the modified Protocol 2* is correct and perfectly private. Upon detection of inconsistent openings, the protocol aborts.*

The proof of Theorem 3 is given in Appendix A.6.

Cost. When no cheating occurs, the only change is that Line 15 uses all D shares (vs. $\lfloor (D+1)/2 \rfloor$ originally). If cheating occurs, RS decoding may run up to three times; its cost is independent of N and linear (or near-linear) in D, thus negligible relative to the base protocol.

8.3 The Case $m < \frac{D}{3}$: BGW + VSS

To tolerate higher malicious fractions we must support secure *multiplication with degree reduction*. Our semi-honest protocols avoid BGW because all computed polynomials in shared secrets have degree at most 2; with $t \le \lfloor (D+1)/2 \rfloor - 1$, no degree reduction is needed. For malicious servers, however, we invoke BGW [7] with VSS so that every server sometimes acts as a dealer and all dealings/reconstructions are verifiable. This yields perfect security against $m < D/3$ [5, Section 6].

Modifications.

(1) Use (t, D)-sharing with $t = \lceil D/3 \rceil$ for Lines 4–5.
(2) For ω at Line 15, interpolate from all D shares; if degree exceeds $t-1$, apply RS decoding.
(3) Implement the multiplication in Line 4 of Subprotocol 4 by BGW's malicious-secure multiplication (with degree reduction).
(4) For ω at Line 5 of Subprotocol 4, if the reconstructed degree exceeds $t-1$, apply RS decoding.
(5) Implement the multiplication in Line 19 of the main protocol by BGW's malicious-secure multiplication.
(6) For Bob's output at Line 21, if the reconstructed degree exceeds $t-1$, apply RS decoding.

Theorem 4. *If $m < \frac{D}{3}$, the BGW+VSS-modified Protocol 2* is perfectly correct and private; adversarial deviations are either corrected or detected, in which case the protocol aborts.*

A proof sketch of Theorem 4 is given in Appendix A.7.

Cost. The dominant overhead is BGW multiplication (one additional round and $O(D^2)$ field elements per multiplication). As usual, this can be substantially amortized by precomputing Beaver triples offline [6]; the online path then remains light.

9 Related Work

Historical context. Oblivious transfer (OT) and oblivious polynomial evaluation (OPE) have been studied in distributed settings where a sender uploads data once and receivers interact only with subsets of servers. Early works [8, 12, 13, 21, 24, 26] explored this mediated or outsourced OT/OPE paradigm, seeking to offload computation while protecting

private queries and sensitive data. Our work continues this line but departs from prior polynomial encodings and semi-honest assumptions by enforcing correct receiver behavior in an information-theoretic setting.

Distributed OT. Naor and Pinkas [24] introduced distributed OT_1^2, in which the sender uploads her data once and the receiver later contacts ℓ out of D servers to obtain his message of choice. They proposed two polynomial-based protocols. The first is conceptually simple but suffers from two weaknesses: each server learns the difference between the sender's two messages, and a collusion between the receiver and a single server fully compromises the sender's privacy. The second protocol uses a more sophisticated polynomial encoding that mitigates these issues, yet it still allows the receiver to reconstruct arbitrary linear combinations of the two messages instead of exactly one.

Blundo et al. [8] extended the work of [24] to distributed OT_1^N. Their construction exhibits similar privacy leakages: differences between all pairs of the sender's messages become exposed, so that learning a single message reveals them all, and a collusion between the receiver and one server compromises the entire message set. They further showed that achieving stronger one-round security requires additional assumptions.

Nikov et al. [26] analyzed the ℓ-out-of-D distributed OT_1^N framework involving a sender, a receiver, and D servers, where the receiver retrieves one of N messages by contacting ℓ servers. They defined (t, k)-security: (a) any t–1 colluding servers learn nothing about the receiver's choice, and (b) a coalition of the receiver with k servers gains no additional information. They proved that such a scheme exists iff $\ell \geq t + k$ and presented a construction achieving the minimal threshold $\ell = t + k$. Their protocol still allows the receiver to learn any linear combination of the sender's messages, thus realizing a weaker form of OT.

Corniaux and Ghodosi [13] proposed a multi-round approach where servers communicate. Their protocol computes a distributed scalar product between secret shares but lacks enforcement of a valid selection vector, again yielding only a weak form of OT where arbitrary linear combinations can be learned.

Distributed OPE. Naor and Pinkas [23] introduced OPE, where a receiver learns $f(\alpha)$ without revealing α. Li et al. [21] presented distributed OPE protocols based on secret sharing, but with semi-honest receivers who can query arbitrary vectors, effectively learning arbitrary linear combinations of coefficients. Cianciullo and Ghodosi [12] improved collusion resistance using masked coefficients and structured interpolation, but still allow unrestricted linear combinations and require the sender to participate per query.

Our OMPE protocol supports multivariate polynomials, enforces correct query structure, and enables a one-time sender upload supporting unlimited future queries.

Positioning and adversarial model. All prior distributed OT and OPE protocols assume semi-honest servers and are vulnerable to message-difference leakage, linear-combination recovery, or require sender participation in every query. Our protocols overcome all these limitations: they enforce correct receiver behavior via distributed vector validation, achieve true OT and OMPE under information-theoretic guarantees, support multivariate polynomial inputs, allow a one-time sender upload serving multiple future receivers, and remain secure against up to $c < D/4$ or $c < D/3$ malicious servers.

10 Conclusion

We presented distributed, information-theoretically secure protocols for core MPC primitives, including k-out-of-N OT, priced OT, generalized OT, and oblivious multivariate polynomial evaluation (OMPE). Our constructions are the first to support both malicious receivers and malicious servers in this distributed mediated model, while enabling an offline

sender and multiple independent receivers. The protocols achieve unconditional privacy and exhibit strong practical efficiency.

These results demonstrate the power of the distributed model for secure computation: delegating computation to external servers yields simple and scalable protocols, and eliminates direct interaction between sender and receiver after setup. We believe this paradigm provides a compelling foundation for further advances in distributed cryptography. Future work includes extending our approach to richer two-party functionalities such as oblivious function evaluation and oblivious Turing-machine evaluation.

References

[1] William Aiello, Yuval Ishai, and Omer Reingold. Priced oblivious transfer: How to sell digital goods. In *EUROCRYPT*, pages 119–135, 2001.

[2] Joël Alwen, Jonathan Katz, Yehuda Lindell, Giuseppe Persiano, Abhi Shelat, and Ivan Visconti. Collusion-free multiparty computation in the mediated model. In *CRYPTO*, pages 524–540, 2009.

[3] Joël Alwen, Abhi Shelat, and Ivan Visconti. Collusion-free protocols in the mediated model. In *CRYPTO*, pages 497–514, 2008.

[4] Benny Applebaum, Amos Beimel, Oded Nir, and Naty Peter. Better secret sharing via robust conditional disclosure of secrets. In *STOC*, pages 280–293, 2020.

[5] Gilad Asharov and Yehuda Lindell. A full proof of the BGW protocol for perfectly secure multiparty computation. *J. Cryptol.*, 30:58–151, 2017.

[6] Donald Beaver. Efficient multiparty protocols using circuit randomization. In *CRYPTO*, pages 420–432, 1991.

[7] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In *STOC*, pages 1–10, 1988.

[8] Carlo Blundo, Paolo D'Arco, Alfredo De Santis, and Douglas R. Stinson. On unconditionally secure distributed oblivious transfer. *J. Cryptol.*, 20(3):323–373, 2007.

[9] Kallista A. Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H. Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. Practical secure aggregation for privacy-preserving machine learning. In *CCS*, pages 1175–1191, 2017.

[10] Octavian Catrina and Florian Kerschbaum. Fostering the uptake of secure multiparty computation in e-commerce. In *ARES*, pages 693–700, 2008.

[11] Benny Chor, Shafi Goldwasser, Silvio Micali, and Baruch Awerbuch. Verifiable secret sharing and achieving simultaneity in the presence of faults (extended abstract). In *FOCS*, pages 383–395, 1985.

[12] Louis Cianciullo and Hossein Ghodosi. Unconditionally secure oblivious polynomial evaluation: A survey and new results. *J. Comput. Sci. Technol.*, 37(2):443–458, 2022.

[13] Christian L. F. Corniaux and Hossein Ghodosi. Scalar product-based distributed oblivious transfer. In *ICISC*, pages 338–354, 2010.

[14] Ivan Damgård and Yuval Ishai. Constant-round multiparty computation using a black-box pseudorandom generator. In *CRYPTO*, pages 378–394, 2005.

[15] Lihi Dery, Tamir Tassa, and Avishay Yanai. Fear not, vote truthfully: Secure multiparty computation of score based rules. *Expert Systems with Applications*, 168:114434, 2021.

[16] Lihi Dery, Tamir Tassa, Avishay. Yanai, and Arthur Zamarin. Demo: A secure voting system for score based elections. In *CCS*, pages 2399–2401, 2021.

[17] Alon Ben Horin and Tamir Tassa. Privacy preserving collaborative filtering by distributed mediation. In *RecSys*, pages 332–341, 2021.

[18] Yuval Ishai and Eyal Kushilevitz. Private simultaneous messages protocols with applications. In *ISTCS*, pages 174–184, 1997.

[19] Seny Kamara, Payman Mohassel, and Mariana Raykova. Outsourcing multi-party computation. *IACR Cryptology ePrint Archive*, 2011. 272.

[20] Pablo Kogan, Tamir Tassa, and Tal Grinshpoun. Privacy preserving solution of dcops by mediation. *Artif. Intell.*, 319:103916, 2023.

[21] Hong-Da Li, Xiong Yang, Dengguo Feng, and Bao Li. Distributed oblivious function evaluation and its applications. *J. Comput. Sci. Technol.*, 19(6):942–947, 2004.

[22] Zhaohao Lin, Weike Pan, and Zhong Ming. FR-FMSS: federated recommendation via fake marks and secret sharing. In *RecSys*, pages 668–673, 2021.

[23] Moni Naor and Benny Pinkas. Oblivious transfer and polynomial evaluation. In *STOC*, pages 245–254, 1999.

[24] Moni Naor and Benny Pinkas. Distributed oblivious transfer. In *ASIACRYPT*, pages 205–219, 2000.

[25] Valeria Nikolaenko, Udi Weinsberg, Stratis Ioannidis, Marc Joye, Nina Taft, and Dan Boneh. Privacy-preserving ridge regression on hundreds of millions of records. In *IEEE Symposium on Security and Privacy (S&P)*, pages 334–348, 2013.

[26] Ventzislav Nikov, Svetla Nikova, Bart Preneel, and Joos Vandewalle. On unconditionally secure distributed oblivious transfer. In *INDOCRYPT*, pages 395–408, 2002.

[27] Takashi Nishide and Kazuo Ohta. Multiparty computation for interval, equality, and comparison without bit-decomposition protocol. In *PKC*, pages 343–360, 2007.

[28] Johannes Schneider. Lean and fast secure multi-party computation: Minimizing communication and local computation using a helper. In *SECRYPT*, pages 223–230, 2016.

[29] Adi Shamir. How to share a secret. *Commun. ACM*, 22(11):612–613, 1979.

[30] Erez Shmueli and Tamir Tassa. Mediated secure multi-party protocols for collaborative filtering. *ACM Transactions on Intelligent Systems and Technology*, 11:1–25, 2020.
[31] Tamir Tassa. Generalized oblivious transfer by secret sharing. *Des. Codes Cryptogr.*, 58(1):11–21, 2011.
[32] Tamir Tassa, Tal Grinshpoun, and Avishay Yanai. PC-SyncBB: A privacy preserving collusion secure DCOP algorithm. *Artificial Intelligence*, 297:103501, 2021.
[33] Tamir Tassa, Tal Grinshpoun, and Avishay Yanai. PC-SyncBB: A privacy preserving collusion secure DCOP algorithm. *Artif. Intell.*, 297:103501, 2021.
[34] Tamir Tassa and Alon Ben Horin. Privacy-preserving collaborative filtering by distributed mediation. *ACM Trans. Intell. Syst. Technol.*, 13:102:1–102:26, 2022.
[35] Tamir Tassa, Ayman Jarrous, and Yonatan Ben-Ya'akov. Oblivious evaluation of multivariate polynomials. *J. Math. Cryptol.*, 7(1):1–29, 2013.
[36] Marino Tejedor-Romero, David Orden, Ivan Marsa-Maestre, Javier Junquera-Sanchez, and Jose Manuel Gimenez-Guzman. Distributed Remote E-Voting System Based on Shamir's Secret Sharing Scheme. *Electronics*, 10:3075, 2021.
[37] Stephen B. Wicker and Vijay K. Bhargava. *Reed-Solomon Codes and their Applications*. John Wiley Sons, 1999.
[38] Andrew C. Yao. Protocols for secure computation. In *FOCS*, pages 160–164, 1982.

A Additional Proofs

A.1 Proof of Lemma 1

If $\mathbf{b} \in \{0,1\}^N$ then $g_n = b_n(b_n - 1) = 0$ for all n, hence $\omega = \sum_n c_n g_n = 0$ deterministically and the test accepts with certainty. Next, assume that $\mathbf{b}$ is non-binary, so there exists an index i with $g_i \neq 0$. Condition on arbitrary fixed values of $\{c_n\}_{n\neq i}$ and let $R := -\sum_{n\neq i} c_n g_n$. The acceptance condition $\omega = 0$ is then equivalent to $c_i g_i = R$. Since $g_i \neq 0$, we have a unique solution $c_i = R g_i^{-1}$. Since c_i is sampled uniformly from $\mathbb{Z}_p$ and is independent of $\mathbf{b}$ and $\{c_n\}_{n\neq i}$, it follows that $\Pr[\omega = 0] = 1/p$. Averaging over the randomness of the remaining multipliers preserves this probability. □

A.2 Proof Sketch of the Correctness and Security of the Priced OT Construction

The correctness and security of the Priced OT construction follow directly from the security of Protocol 3 (Theorem 2). The modified protocol invokes Protocol 3 as a black box and only alters the validity check performed on Bob's selection vector. Instead of verifying that $\sum_n b_n = k$ for a public value k, the servers verify that $\sum_n w_n b_n = T$ for a secret value T by evaluating the affine function $T - \sum_n w_n b_n$ on secret shares. This computation uses the same secret-sharing operations as in Protocol 3. No additional information about T or about Bob's choice leaks to Alice or to the servers beyond what is implied by correctness, since the protocol reveals only whether the affine relation holds (i.e., whether $e = 0$). Similarly, Bob's view is unchanged from Protocol 3, except for receiving the same set of messages conditioned on the validity of his request. Therefore, correctness and statistical privacy for both parties, including against collusions of up to $t-1$ servers, are preserved. Hence, Theorem 2 extends to the Priced OT variant. □

A.3 Proof Sketch of the Correctness and Security of Protocol 5

The correctness and security of Protocol 5 follow from Theorem 2 and from the secrecy of the secret-sharing scheme Σ. The protocol invokes Protocol 3 twice: first, to allow Bob to retrieve the shares of s corresponding to his chosen messages, and later, to obtain the actual messages themselves. The only additional mechanism is the consistency test on the secret s, in which Alice and Bob supply (t, D)-shares of s, and the servers verify equality without learning s.

In Protocol 5, Bob is required to request a *maximal authorized subset*, that is, a set $B \in \mathcal{A}^0$. If Bob's selection belongs to $\mathcal{A}^0$, then by the definition of Γ the retrieved shares reconstruct s; consequently, the consistency check succeeds and correctness follows. If Bob's selection is not in $\mathcal{A}^0$, the retrieved shares reveal no information about s by the secrecy of Σ, and Bob's only option is to guess $s \in \mathbb{Z}_p$, which succeeds with probability exactly $1/p$. Except for this negligible probability, the servers detect the inconsistency and abort the protocol. No additional information about the messages or Bob's choice is revealed beyond what is inherent in Protocol 3. Hence, Protocol 5 preserves correctness and statistical privacy against any coalition of up to $t-1$ servers. □

A.4 Proof of Lemma 2

Assume that $\mathbf{b}$ is as in Eq. (4). Then for every multi-index $\mathbf{j} \in \mathbb{Z}_+^{k,N}$, the corresponding entry in $\mathbf{b}$ is $b_{\mathbf{j}} := \alpha^{\mathbf{j}}$. Hence, for any $2 \le n \le N$ and for any $\mathbf{j} \in T_n$, there exists at least one monomial $\mathbf{h} \in T_{n-1}$ such that $\mathbf{j} = \mathbf{h} + \mathbf{e}_i$, for some $1 \le i \le k$. Let us compare the monomial $b_{\mathbf{j}} := \alpha^{\mathbf{j}}$ with the monomial $b_{\mathbf{h}} := \alpha^{\mathbf{h}}$. The two multi-indices $\mathbf{j}$ and $\mathbf{h}$ equal in all entries except for the ith entry, where $j_i = h_i + 1$. Therefore,

$$b_{\mathbf{j}} := \alpha^{\mathbf{j}} = \alpha^{\mathbf{h}} \cdot \alpha_i = b_{\mathbf{h}} \cdot b_{\mathbf{e}_i}\,.$$

Hence, such a vector will pass all stages of the DVV in Lines 6-11.

Assume next that $\mathbf{b}$ does not comply with the form as in Eq. (4). That means that

$$\mathbf{b} = (1, \alpha_1, \ldots, \alpha_k, b_{\mathbf{j}} \,:\, 2 \le |\mathbf{j}| \le N),$$

where there exists at least one entry $b_{\mathbf{j}}$, where $2 \le |\mathbf{j}| \le N$, that is not of the form as in Eq. (4). Let us focus on the first multi-index $\mathbf{j}$ that is not of that form. Namely, $\mathbf{j}$ is the first multi-index for which

$$b_{\mathbf{j}} \ne \alpha^{\mathbf{j}}, \tag{5}$$

where $\alpha = (\alpha_1, \ldots, \alpha_k)$. Assume that $|\mathbf{j}| = n \in [2, N]$ and let i be any index between 1 and k such that $\mathbf{j} = \mathbf{h} + \mathbf{e}_i$ for some $\mathbf{h} \in T_{n-1}$. By the minimality of $\mathbf{j}$ it means that

$$b_{\mathbf{h}} = \alpha^{\mathbf{h}}. \tag{6}$$

From Eqs. (5) and (6) it follows that the validation check in Lines 9+10 would fail for those multi-indices. That completes the proof. □

A.5 Proof Sketch of the Correctness and Security of Protocol 6

Correctness follows since, conditioned on successful validation of $\mathbf{b}$, the protocol computes $\sum_{\mathbf{j}\in\mathbb{Z}_+^{k,N}} a_{\mathbf{j}} \cdot \alpha^{\mathbf{j}} = f(\alpha)$ via the distributed scalar product functionality. Each server outputs its Shamir share of the result, and Bob recovers $f(\alpha)$ from the D replies.

Privacy for Alice follows from the DVV validation step together with the security of Shamir secret sharing. The DVV procedure guarantees that Bob's submitted vector $\mathbf{b}$ must be of the form $(\alpha^{\mathbf{j}})_{\mathbf{j}\in\mathbb{Z}_+^{k,N}}$. Thus, Bob cannot deviate from the prescribed input format and cannot induce the servers to compute any linear combination of the coefficients $\{a_{\mathbf{j}}\}$ beyond the value $f(\alpha)$. Since the servers hold Shamir shares of Alice's input, any coalition of at most $t-1$ servers learns no information beyond what Bob obtains from the output.

Privacy for Bob follows from the fact that Alice receives no information from the servers during the computation, and the DVV check reveals only whether Bob's shares satisfy the multiplicative consistency relations among monomials. These checks do not leak information about the underlying point α beyond what is implied by the final value $f(\alpha)$.

Therefore, the protocol achieves information-theoretic security for both parties under an honest-majority assumption, inheriting the guarantees of Theorem 2. □

A.6 Proof of Theorem 3

Write $D = 4j - i$ with $j \ge 1$ and $i \in \{0, 1, 2, 3\}$. Then $t = \lceil D/4 \rceil = j$ and, since $m < D/4$, we have $m \le t - 1$. Alice and Bob use sharing polynomials of degree $t-1$, so at least t shares are needed to reconstruct any secret; consequently any coalition of at most $t-1$ (in particular, the m malicious) servers learns no information about the shared secrets.

When the servers compute ω in Line 15, the sharing polynomial has degree $t-1$. The corresponding Reed–Solomon code has parameters $[D, t, D-t+1]$ and corrects up to $\lfloor (D-t)/2 \rfloor$ errors upon reconstruction from all D broadcast shares. Here

$$\frac{D-t}{2} \;=\; \frac{4j-i-j}{2} \;=\; \frac{3j-i}{2} \;\ge\; j-1 \;\ge\; m,$$

so decoding succeeds against all m adversarial shares.

In the other two reconstructions (Line 5 of Subprotocol 4 and Line 21 of the main protocol) the target degree is $2t-2$. The code parameters are $[D, 2t-1, D-(2t-2)]$, which corrects up to $\lfloor (D-(2t-1))/2 \rfloor$ errors. Since

$$\frac{D-(2t-1)}{2} \;=\; \frac{4j-i-(2j-1)}{2} \;=\; \frac{2j-i+1}{2} \;\geq\; j-1 \;\geq\; m,$$

decoding again succeeds against all m adversarial shares.

Therefore all reconstructions yield the correct values, and privacy holds because any set of at most $t-1$ shares reveals nothing by Shamir secrecy. □

A.7 Proof sketch of Theorem 4

The theorem follows directly from the security of BGW [7] with verifiable secret sharing (VSS). In the $m < D/3$ regime every server occasionally acts as a dealer, and all shares are verified via bivariate VSS. As in BGW, all intermediate values remain perfectly hidden, and any deviation during dealing or reconstruction is either detected or correctable via Reed–Solomon decoding. Degree-reduction is performed using standard BGW multiplication, ensuring that shared polynomials remain of degree at most $t-1 = \lceil D/3 \rceil - 1$. Thus, correctness, perfect privacy, and robustness follow exactly as in the classical BGW model. □

B Communication Cost Derivations

We derive the communication costs presented in Table 1. Communication is measured in field elements over $\mathbb{Z}_p$; a broadcast of L elements contributes $L(D-1)$ to $\mathrm{Com}_{\mathrm{MM}}$ since servers communicate only via broadcast. As in the main text, $\mathrm{Com}_{\mathrm{AM}}$ and $\mathrm{Com}_{\mathrm{BM}}$ denote messages from Alice and Bob to servers, and $\mathrm{Com}_{\mathrm{MM}}$ messages among servers.

Secret Sharing (DSP).. Alice and Bob secret-share N values to D servers and each server returns one share. Thus $\mathrm{Com}_{\mathrm{AM}} = \mathrm{Com}_{\mathrm{BM}} = (N+1)D$ and $\mathrm{Com}_{\mathrm{MM}} = 0$.

OT_1^N (Protocol 2). $\mathrm{Com}_{\mathrm{AM}} = ND$ (Line 4), $\mathrm{Com}_{\mathrm{BM}} = (N+1)D$ (Line 5 and Line 20). As for the communication between the servers, it is executed in the DVV subprotocol. We have here $ND(D-1)$ due to the first part in the validation (Line 9) and $t(D-1) < D(D-1)$ due to the second part (Line 15). In addition, the servers have to communicate also in performing the scrambling (Line 8) in order to generate random shares of 0. That communication can take place offline and its cost is $ND(D-1)$. Hence, the overall communication cost among the servers is bounded by $\mathrm{Com}_{\mathrm{MM}} = (2N+1)D(D-1)$.

OT_1^N with optimized DVV (Protocol 2).* Recall that Subprotocol 4 verifies that Bob's vector is binary. The cost of that validation in Protocol 2 is $2ND(D-1)$, as argued above. However, the communication cost of Subprotocol 4 is due to Lines 2 and 4 there and it is only $(\theta+1)D(D-1)$, where θ is the size of the random seeds σ_d, $d \in [D]$, in terms of field elements. (For example, we used a seed of 160 bits, so that $\theta = 160/\lceil \log p \rceil$.) Hence, the overall communication cost among the servers in Protocol 2* is bounded by $\mathrm{Com}_{\mathrm{MM}} = (\theta+2)D(D-1)$.

OT_k^N (Protocol 3). Identical to OT_1^N except Bob receives N shares per server at the end, giving $\mathrm{Com}_{\mathrm{AM}} = ND$, $\mathrm{Com}_{\mathrm{BM}} = 2ND$, and the same $\mathrm{Com}_{\mathrm{MM}} = (2N+1)D(D-1)$.

OT_k^N with optimized DVV (Protocol 3).* As above but with optimized DVV, yielding $\mathrm{Com}_{\mathrm{AM}} = ND$, $\mathrm{Com}_{\mathrm{BM}} = 2ND$, and $\mathrm{Com}_{\mathrm{MM}} = (\theta+2)D(D-1)$.

Computing OT_k^N via k calls to OT_1^N. Alice shares once, Bob repeats k times. Thus $\mathrm{Com}_{\mathrm{AM}} = ND$, $\mathrm{Com}_{\mathrm{BM}} = k(N+1)D$, and $\mathrm{Com}_{\mathrm{MM}} = k(2N+1)D(D-1)$, which exceeds the communication cost in Protocol 3 for all $k \geq 2$.

Priced OT.. Alice and Bob additionally share T, adding D field elements to each of $\mathrm{Com}_{\mathrm{AM}}$ and $\mathrm{Com}_{\mathrm{BM}}$, while one equality check among servers adds $D(D-1)$ to $\mathrm{Com}_{\mathrm{MM}}$. Thus $\mathrm{Com}_{\mathrm{AM}} = (N+1)D$, $\mathrm{Com}_{\mathrm{BM}} = (2N+1)D$, and relative to OT_k^N we obtain $\mathrm{Com}_{\mathrm{MM}} = 2(N+1)D(D-1)$ (or $(\theta+3)D(D-1)$ in the optimized version).

Generalized OT (Protocol 5). Two OT_k^N calls are run on Bob's input, but DVV applies once; additionally Alice and Bob share s^A and s^B. This gives $\mathrm{Com}_{\mathrm{AM}} = (2N+1)D$, $\mathrm{Com}_{\mathrm{BM}} = (3N+1)D$, and $\mathrm{Com}_{\mathrm{MM}} = (2N+2)D(D-1)$ (or $(\theta+3)D(D-1)$ when using the optimized DVV).

OMPE (Protocol 6). Alice and Bob share vectors of size $N_k = \binom{N+k}{k}$, the DVV mechanism checks $N_k - k - 1$ entries, and finally Bob receives shares in the desired output. Thus $\mathrm{Com}_{\mathrm{AM}} = N_k D$, $\mathrm{Com}_{\mathrm{BM}} = (N_k+1)D$, and $\mathrm{Com}_{\mathrm{MM}} = (N_k - k - 1)D(D-1)$.

All communication complexities in Table 1 now follow.

C Additional Details on Verifiable Secret Sharing

For completeness we summarize the algebraic components underlying the malicious-server extensions in Section 8. Our protocols rely only on standard VSS tools; the presentation here follows [5, 11].

Reed–Solomon correction bounds. Let $f(x) = \sum_{i=0}^{t-1} a_i x^i$ be a degree-$(t-1)$ polynomial over $\mathbb{Z}_p$, and let $(f(1), \ldots, f(D)) \in \mathbb{Z}_p^D$ be the corresponding Shamir shares. The set of all such vectors forms a $[D, t, D-t+1]$ Reed–Solomon code. Thus, if all D shares are broadcast, nearest-codeword decoding corrects up to $\left\lfloor \frac{D-t}{2} \right\rfloor$ adversarially corrupted shares and recovers $f(0)$. When the target polynomial has degree at most $2t-2$ (as in DVV and final output reconstruction), the code parameters become $[D, 2t-1, D-(2t-2)]$, and we can correct $\left\lfloor \frac{D-(2t-1)}{2} \right\rfloor$ errors. These bounds yield the thresholds used in Section 8.

Bivariate VSS.. To guarantee that a dealer distributes consistent shares from a degree-$(t-1)$ polynomial, we employ the classical bivariate VSS technique [5, Section 5]. The dealer samples

$$S(x,y) = \sum_{i=0}^{t-1} \sum_{j=0}^{t-1} a_{i,j} x^i y^j, \quad S(0,0) = s,$$

and sends to party M_d the univariate polynomials

$$f_d(x) = S(x,d), \qquad g_d(y) = S(d,y).$$

For every pair of indices $d, d' \in [D]$, the corresponding parties check that

$$f_d(d') = g_{d'}(d) \qquad \text{and} \qquad f_{d'}(d) = g_d(d')\,.$$

If all equalities hold, the shares are consistent with a single bivariate polynomial, and the secret can be reconstructed from $f_d(0) = S(0,d)$, $d \in [D]$, by interpolating a degree-$(t-1)$ univariate polynomial $S(0,y)$ and outputting $S(0,0)$.

Security. Any $t-1$ parties learn no information about s (perfect secrecy). If up to $\lfloor (D-t)/2 \rfloor$ corrupted parties broadcast inconsistent shares, the honest parties still recover s by decoding; otherwise the inconsistency is detected and the protocol aborts.