MOMPE: Multiparty Oblivious Multivariate Polynomial Evaluation

Duobin Lyu, Hongwei Zhang, Zening Zhao, Jinsong Wang✉ .

***Abstract*—This paper proposes a novel Multiparty Oblivious Multivariate Polynomial Evaluation (MOMPE) protocol, which enables secure multivariate polynomial computation among multiple parties for the first time. Unlike traditional Oblivious Polynomial Evaluation (OPE) protocols limited to two-party secure computation, MOMPE divides participating parties into a sender holding the multivariate polynomial P and n receivers, each holding a secret value α_i corresponding to a variable in P. Upon completion, the receivers learn $P(\alpha_1, \alpha_2, \ldots, \alpha_n)$, while the sender learns nothing. Assuming no collusion between the sender and receivers, we demonstrate MOMPE's security under the stringent Universal Composability (UC) framework, ensuring it remains secure in complex multiparty interactive environments. Our implementation and experiments validate MOMPE's performance, showing significantly less time consumption than the state-of-the-art Overdrive protocol for multiparty multivariate polynomial computation, while maintaining similar efficiency to OPE for participating parties. MOMPE has extensive application potential in privacy-preserving machine learning and collaborative data analysis, where multiple parties need to compute and analyze data together while protecting privacy. MOMPE offers a secure and efficient method for parties to utilize data without revealing sensitive information, opening new possibilities for privacy-preserving data collaboration and analysis.**

***Index Terms*—oblivious polynomial evaluation, multivariate polynomial, multiparty computation**

I. Introduction

OPE is a fundamental building block in cryptographic protocols for evaluating private polynomials over private inputs. In OPE, there are two types of participants: the sender and the receiver. The sender possesses a secret univariate polynomial $P(x)$, while the receiver holds a private input α. Upon completion of OPE, the receiver learns $P(\alpha)$, whereas the sender learns nothing. OPE can be considered as a generalization of Oblivious Linear Evaluation (OLE). Over the past three decades, the general-purpose Secure Multiparty Computation(SMPC) field has seen significant development and progress. Nevertheless, MPC protocols tailored for specific computations are still widely adopted worldwide due to their outstanding performance and efficiency, playing a crucial role in various real-world applications. The OPE protocol is a prime example, designed specifically for handling polynomial computation problems and also serving as a foundational component in several areas, including data mining, secure keyword search, and private set intersection. [1]–[9]

OPE was first introduced by Naor and Pinkas in 1999 [10], and over the past two decades, various OPE protocols have been developed. The OPE scheme proposed in [11], [12] can withstand malicious adversaries. However, [11] requires at least 17 rounds of interaction to complete the computation, with each party sending (λn) Paillier encryptions to the other, where λ is the security parameter, and n is the degree of the polynomial. The authors claim that their scheme is only suitable for low-degree polynomials. [12] presents an OPE scheme constructed using algebraic pseudorandom functions (PRF) to evaluate a $\log_2 d$ group's exponent. This scheme improves the computational efficiency of [11] by reducing the number of modular exponentiations and removing the need for a trusted setup while maintaining the same number of interaction rounds and communication complexity as [11]. It applies to private set membership 2PC. Malika et al. [13] proposed a "myope" non-interactive OPE scheme, myope, which can also withstand malicious adversaries but is complex and challenging to implement. The OPE scheme under the semi-honest model proposed by Gajera et al. [14] requires a third-party server, which cannot avoid the issue of the server and sender colluding. [15] shows that the functionality of OPE can be used as a subprotocol for secure data entanglement. [16] introduced the concept of blind polynomial evaluation, a primitive closely related to OPE, and explored its application in secure data trading. [14] proposed a verifiable and private OPE scheme and presented a verifiable IND-CFA OPE scheme based on Paillier. [17] distributed the OPE scheme for multivariate polynomials proposed in [18] to improve computational efficiency.

In summary, OPE protocols constructed based on noise encoding and OT (Oblivious Transfer) techniques are considered the most efficient method for securely computing polynomials in a multiparty computation setting. They rely solely on simple finite field operations, thus requiring minimal computational resources and being easy to implement and operate. However, despite various OPE schemes that effectively handle the computation of univariate polynomials, extending these schemes to multivariate polynomials remains a problem that needs to be fully addressed. A novel OPE scheme proposed in [18] has achieved some capability for computing multivariate polynomials. However, a significant limitation is that all secret data is still concentrated in the hands of a single receiver. This design inherently belongs to the traditional two-party

This work was supported by the National Natural Science Foundation of China [grant number 62072336]; the Tianjin Technical Innovation Guidance Special Project [grant number 22YDPYGX00040]; the Key Research and Development Program of Tianjin [grant number 23YFZCSN00240]; and the Tianjin Graduate Research Innovation Project [grant number 2022BKYZ045].

Jinsong Wang is corresponding author. The authors are with the School of Computer Science and Engineering, Tianjin University of Technology, Tianjin 300384, China (e-mail: jswang@tjut.edu.cn)

979-8-3503-5171-2/24/$31.00 ©2024 IEEE

secure computation framework and is not suitable for scenarios involving multiparty data sharing or computation. To overcome this limitation, this paper introduces a multivariate polynomial-oriented OPE scheme, MOMPE, which accommodates multiple receivers. In MOMPE, there are multiple receivers, each holding a secret data corresponding to a variable in the multivariate polynomial. Upon completion of the protocol, the sender remains unaware of any receiver's secret value, and the receivers do not know the sender's polynomial but the computation result. MOMPE, compared to OPE which is confined to two-party secure computations, has a broader application scope, particularly in scenarios involving multi-party data collaborative computations where data privacy must be maintained. Specifically, in the field of multi-party joint privacy-preserving machine learning, as well as in industries such as finance, healthcare, and federated data analysis, MOMPE facilitates effective data sharing and collaboration while ensuring information security.

A. Our Contributions

In this paper, we innovatively propose the Multiparty Oblivious Multivariate Polynomial Evaluation protocol (MOMPE), aiming to address the limitation of traditional OPE protocols regarding the number of participants. Our work primarily contributes in the following three aspects:

- We have extended the number of participants. The MOMPE protocol breaks through the limitation of traditional OPE protocols confined to a single receiver, allowing multiple receivers to participate, each holding a secret value corresponding to a variable in the multivariate polynomial. This design supports the participation of an unlimited number of receivers, greatly enhancing the applicability and flexibility of the protocol.
- We have proved the UC security of the MOMPE. In the paper, we elaborate on the security proof of the MOMPE protocol. As long as the attacker cannot corrupt both the sender and the receivers simultaneously, our protocol provides strong security guarantees under the UC model, which is crucial for multiparty computation environments.
- We have implemented the MOMPE protocol and conducted a comprehensive experimental comparison with the most efficient SMPC protocol currently available for computing multivariate polynomials, Overdrive, and the most efficient OPE protocol. The experimental results demonstrate that our protocol's time consumption is comparable to that of the OPE protocol and significantly lower than that of the Overdrive protocol, thus proving the efficiency and competitiveness of MOMPE in real-world applications.

Overview of the paper. In Section II, we provide an overview of prior protocols and explain the intuition behind the construction of our protocol. After some preliminaries are given in Section III, we give a detailed description of our protocol in Section IV. In Section V, we discuss the security of our protocol. Section VI evaluates our scheme experimentally and compares it with the previous scheme.

II. Technical Overview

The core of our protocol is still the Noise Polynomial Reconstruction (NPR) problem, which was first introduced by Naor and Pinkas when they introduced their OPE protocol [10]. NPR provides an effective method for implicitly computing multiplication by encoding messages through linear codes and mixing these codes with random values. This mixture results in a vector that obscures which elements are part of the codeword and which are random, thus hiding the original message, and uses OT to filter out the codeword. The core of our protocol is still the Noise Polynomial Reconstruction (NPR) problem, which was first introduced by Naor and Pinkas when they introduced their OPE protocol [10]. NPR provides an effective method for implicitly computing multiplication by encoding messages through linear codes and mixing these codes with random values. This mixture results in a vector that obscures which elements are part of the codeword and which are random, thus hiding the original message, and uses OT to filter out the codeword. In a little more detail, the OPE protocol is divided into three stages: encoding, OT, and interpolation. In the encoding stage, the receiver inputs $x \in \mathbb{F}$ as $d > t$ sampling points at locations α_i for a random polynomial P of degree $d > t$. The polynomial is then evaluated at, for example, $4d$ positions β_i, with most of these positions being replaced by uniformly random values to create the encoding. This encoding is indistinguishable from a uniformly random vector. Currently, OPE protocols involve only two parties: the receiver and the sender. Although [18] proposed an OPE protocol based on multivariate polynomials, a single receiver still holds all secret values. The key challenge when extending the OPE protocol to multiple parties is how the receivers can agree upon the same set of random values. Using the same set of random values among all receivers is crucial to ensure the correctness of subsequent interpolation computations.

A. Our Solution

This paper presents MOMPE, a Multiparty Oblivious Multivariate Polynomial Evaluation protocol designed to accomplish the multiparty computation task of the multivariate polynomial P. As shown in Fig. 1. MOMPE involves two types of participants: the sender and the receivers. Only one sender provides the computation method: multivariate polynomial P_s. The number of receivers, denoted as n, corresponds to the number of variables in the multivariate polynomial. Each receiver holds their private data $\alpha_1, \alpha_2, ..., \alpha_n$ as inputs to provide for P_s. MOMPE consists of two stages: the negotiation phase and the extension phase. During the negotiation phase, the sender does not participate in the protocol. Each receiver generates a random number ϵ and transmits it to the other receiver. After receiving $n-1$ random numbers, each receiver computes the function G with n random numbers $\epsilon_1, \epsilon_2, \ldots, \epsilon_n$, as input, and the result of the computation is $seed$. Each receiver computes the seed's hash using the hash

function $\mathsf{H}: \{0,1\}^* \to \mathsf{h}$ and discloses it publicly. After verifying that the publicly disclosed hashes from all receivers are identical, the seed is used as the random seed. Subsequently, a pseudo-random function $\mathsf{PRF}: \{0,1\}^* \to \mathbb{F}$ is employed to generate a set of random numbers, $\mathcal{T}$, which serves as the agreed-upon set of random numbers for all receivers. In the extension phase, each receiver hides its secret value within a random polynomial and sends the points on the random polynomial, along with noise, to the sender. The sender then employs OT to filter out the noise. Upon completion of the protocol, the sender obtains only the result of the target function without gaining knowledge of the receivers' private data. Meanwhile, the receivers learn nothing about each other's private data. The protocol is defined as follows:

Input:

- Sender: a multivariate polynomials of degree d_{ps} over a finite field $\mathbb{F}$,

$$P_s(x_1, x_2, ..., x_n) = \sum_{i_1=0}^{m_1} \sum_{i_2=0}^{m_2} \cdots \sum_{i_n=0}^{m_n} a_{i_1,i_2,...,i_n} x_1^{i_1} x_2^{i_2} ... x_n^{i_n}$$

  where $a_{i_1,i_2,...,i_n}$ is the coefficient of the polynomial P_s and $\{m_1, m_2, \ldots, m_n\}$ is the degree of the variable $\{x_1, x_2, \ldots, x_n\}$. The degree of polynomial P_s is d_{ps}.
- Receiver: Let $\mathsf{R}_1, \mathsf{R}_2, \ldots, \mathsf{R}_\mathsf{n}$ be n receivers, each with its own private data $\alpha_1, \alpha_2, \ldots, \alpha_n \in \mathbb{F}$, where $\alpha_1, \alpha_2, \ldots, \alpha_n$ is the input of each receiver.

Output:

- Sender: Nothing.
- Receiver: $P_s(\alpha_1, \alpha_2, ..., \alpha_n)$.

III. Preliminaries

We use the standard notions of probabilistic polynomial time (PPT) algorithms, negligible and overwhelming functions. Further, we let α_i denote the private value of the ith receiver. Unless noted otherwise, P denotes a polynomial in $\mathbb{F}$. We will typically denote a value $\widehat{x}$ chosen or extracted by the simulator, while x^* is chosen by the adversary $\mathcal{A}$.

A. Oblivious Transfer

Oblivious Transfer (OT) is one of the fundamental tools in cryptographic protocols and finds widespread application in secure multiparty computation (MPC). Typically, an OT protocol involves two types of participants: the sender and the receiver. The sender possesses multiple messages, while the receiver wishes to obtain one or more of these messages. Upon completion of the protocol, the receiver acquires the desired message(s) without gaining any knowledge of the other messages; meanwhile, the sender needs to be made aware of which message(s) the receiver has obtained. OT is instrumental in constructing Yao's two-party computation (2PC) protocols and building many other MPC protocols that offer security against semi-honest and malicious adversaries. Additionally, OT is a versatile tool that can design many other cryptographic protocols. OT was first introduced in 1981 with a 1-out-of-2 OT [19] protocol(OT_2^1). Over the past five decades, various OT protocols, such as OT_n^1 and OT_n^k, have been developed. Our work uses an OT_n^k, where Sender holds n messages $m_1, m_2, \ldots, m_n$ and Receiver has k selections $s_1, s_2, \ldots, s_k$, $s_i \in \{1, \cdots, n\}$. Upon completion of the protocol, receiver learns k messages $m_{s_1}, m_{s_2}, \ldots, m_{s_k}$ from sender corresponding to its selections while remaining oblivious to the content of the other messages. There are two primary methods for constructing an OT_n^k: one involves parallelizing the execution of k instances of OT_n^1 [20], and the other constructs the protocol directly using cryptographic techniques such as the Decisional Diffie-Hellman (DDH) assumption, bilinear mappings, and elliptic curve cryptography [21]–[24]. The most communication-efficient scheme to date was constructed in 2018 by Lai et al. [25] based on two new assumptions, where the communication from Sender to Receiver consists of $n+1$ group elements and the communication from receiver to sender is a fixed 3 group element, independent of n and k. The MOMPE structure described in this paper can be based on any OT_n^k from these two categories (of course, the security depends on the security of the oblivious transfer protocol in use). The ideal functionality for OT_n^k is specified as follows Fig. 2:

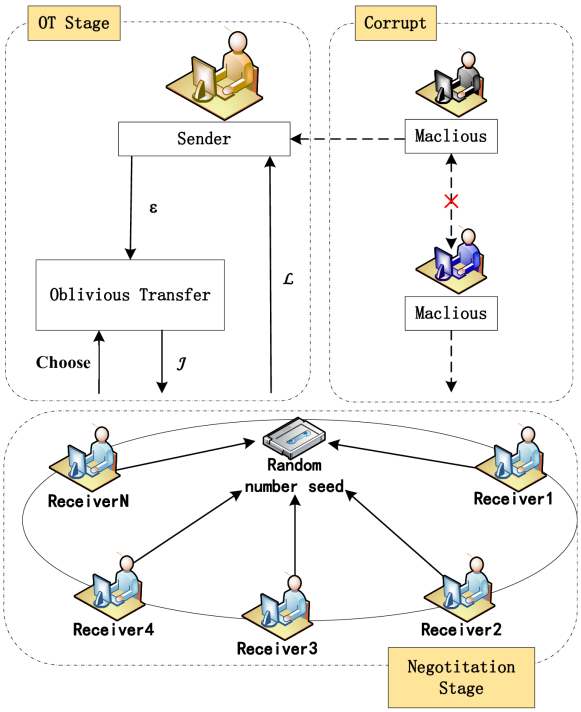

Fig. 1: System model.

B. NPR-based Pseudo-randomness Assumption

NPR-based Pseudo-randomness Assumption(NPA) is a significant issue studied in cryptography and computational complexity theory. It pertains to how to recover the original polynomial from a set of point-value pairs. This assumption is significant in theoretical computation because it involves

Functionality $\mathcal{F}_{\mathsf{OT}_n^k}$

1. Upon receiving a message (inputS, $m_1, m_2, \ldots, m_n$) from sender S, where each $m_i \in \{0,1\}^*$, verify that there is no stored tuple, else ignore that message. Store (inputS, $m_1, m_2, \ldots, m_n$) and send a message (input) to $\mathcal{A}$.
2. Upon receiving a message (inputR, $s_1, s_2, \ldots, s_k$) from sender S, where each $m_i \in \{0,1\}^*$, verify that there is no stored tuple, else ignore that message. Store (inputR, $m_1, m_2, \ldots, m_n$) and send a message (input) to $\mathcal{A}$.
3. Upon receiving a message (deliver, S) from $\mathcal{A}$, check if both (inputS, $m_1, m_2, \ldots, m_n$) and (inputR, $s_1, s_2, \ldots, s_k$) are stored, else ignore that message. Send (delivered) to S.
4. Upon receiving a message (deliver, R) from $\mathcal{A}$, check if both (inputS, $m_1, m_2, \ldots, m_n$) and (inputR, $s_1, s_2, \ldots, s_k$) are stored, else ignore that message. Send (output, $m_{s_1}, m_{s_2}, \ldots, m_{s_k}$) to R.

Fig. 2: Ideal functionality for k-out-of-n Oblivious Transfer.

NPA

NPR:

- Input: Integer k, n and N number pair $(x_i.y_i)_{i=1}^N$, where $x_i, y_i \in \mathbb{F}$, $\mathbb{F}$ is a finite field.
- Output: All univariate polynomials $P(x)$ of degree no more than k. And there exist at least n (x_i, y_i) in each polynomial of the output that satisfy $P(x_i) = y_i$.

NPA procedure: Let k, n, m be integer. $\mathbb{F}$be a finite field, and α be a constant value in $\mathbb{F}$. Let $\Pr_{k,\alpha}^{n,m}$ be the probability distribution of the point set generated by the following steps.

1. Draw a random polynomials $P \in \mathbb{F}$ of degree at most k that satisfies $P(0) = \alpha$.
2. Generating nm random numbers $\{x_1, x_2 \ldots x_{nm}\}$ that are not equal to each other and are not equal to 0 over finite field $\mathbb{F}$.
3. Randomly choose a set $\mathcal{T}$ containing n elements,$\mathcal{T} \subset \{1, 2, \ldots, mn\}$.
4. Let $y_i = P(x_i)$ for $i \in \mathcal{T}$ and let y_i be a random value in $\mathbb{F}$for $i \notin \mathcal{T}$.
5. Output set $\{(x_i, y_i)\}_{i=1,j=1}^{nm,i=1}$.

Fig. 3: Encoding procedure for NPA.

recovering information and security. The security of our protocol primarily relies on this assumption. In brief, NPA is a form of information encoding that results from mixing random field elements with the original message. It is closely related to the effective decoding of randomly linear codes or Reed-Solomon codes with high noise [26]. The encoded information is irretrievable. NPA was first introduced by Naor and Pinkas in [10], specifically for implementing OPE. [27] used the techniques of [28] to propose an encoding process that can encode multiple field elements. They also used Reed-Solomon codes and then artificially obscured the codeword using random errors to conceal the location of the codeword elements in the resulting string. In [29], Kiayias and Yung provided additional evidence for the hypothesis, showing that if the adversary cannot determine whether the randomly encoded position i is part of the codeword, then the noisy codeword is indeed pseudo-random. [10] proposed two different encodings and related assumptions for their protocol, and one of the assumptions was later broken by [30], [31], and a fixed version was proposed in [32]. [33] provides proof of security for the robustness of this assumption against leakage. After a long development period, many scholars [34], [35] have taken this complex problem as the underlying assumption of cryptographic schemes. For the assumption to hold, [27] propose that the signal-to-noise ratio should less than 0.5. Fig. 3 shows the NPA procedure.

Assumption 1: Let λ be a security parameter, and let $n(\lambda)$, $m(\lambda)$, $k(\lambda)$, $F(\lambda)$ be polynomially bounded functions that define the parameters n, m, k and the size in bits of the representation of an element in the field $\mathbb{F}$. Let $A_{k,\alpha}^{n,m}$ and $A_{k,\alpha'}^{n,m}$ be random variables that are chosen according to the distributions $\Pr_{k,\alpha}^{n,m}$ and $\Pr_{k,\alpha'}^{n,m}$, respectively. Then it holds that for every $\alpha, \alpha' \in \mathbb{F}$ the probability ensembles $\Pr_{k,\alpha}^{n,m}$and $\Pr_{k,\alpha'}^{n,m}$ are computationally indistinguishable for PPT adversaries.

C. Universal Composability Framework

We state and prove our results in the Universal Composability (UC) framework of [36], [37]. Our protocols work with n receivers and a sender. For our construction, we consider security against malicious, static adversaries, i.e., corruption may only occur before the protocols start. In the malicious model, our MOMPE protocol requires that the sender can be malicious but will not participate in collusion. An adversarial, corrupt sender and a corrupt receiver will not launch a joint attack. We demonstrate the security of accessing OT in a hybrid model.

Security is defined by comparing an *ideal model* and a *real model*. The protocol Π between protocol participants is executed in the real model. In contrast, in the ideal model, participants only communicate with the ideal function $\mathcal{F}$, which should guarantee the ideal security of the protocol. For an adversary $\mathcal{A}$ in the real protocol who coordinates the behavior of all malicious parties, there has to exist a simulator $\mathcal{S}$ for $\mathcal{A}$ in the ideal protocol. Environment $\mathcal{Z}$ exists in real and ideal models, provides input to the participants and can read the outputs. The simulator $\mathcal{S}$ must ensure that $\mathcal{Z}$ cannot distinguish between these models. Therefore, its security can still be ensured even if a concurrent protocol is utilized. Typically, we suppose that $\mathcal{A}$ is a fictitious adversary under

the control of $\mathcal{Z}$. This means that $\mathcal{Z}$ can adaptively choose its inputs based on the protocol messages it receives and send messages on behalf of the (corrupted) protocol party.

More formally, let $\mathrm{Real}_{\Pi}^{\mathcal{A}}(\mathcal{Z})$ represents the random variable output by $\mathcal{Z}$ when interacting with the real model. Let $\mathrm{Ideal}_{\mathcal{F}}^{\mathcal{S}}(\mathcal{Z})$ represent the random variable output by $\mathcal{Z}$ when interacting with the ideal model. The protocol Π is UC secure means that for any PPT adversary $\mathcal{A}$, there exists a PPT simulator $\mathcal{S}$ such that for any PPT environment $\mathcal{Z}$, $\mathrm{Real}_{\Pi}^{\mathcal{A}}(\mathcal{Z}) \approx \mathrm{Ideal}_{\mathcal{F}}^{\mathcal{S}}(\mathcal{Z})$.

IV. Multiparty Oblivious Polynomial Evaluation

Multiparty Oblivious Polynomial Evaluation(MOMPE) is designed to compute multivariate polynomials in the following setting: There are two types of participating parties. One party, which we refer to as the sender S, provides the multivariate polynomial $P_s \in \mathbb{F}$. The other party consists of n receivers $\{\mathsf{R}_1, \mathsf{R}_2, \ldots, \mathsf{R}_n\}$, each possessing a secret value $\alpha_i \in \mathbb{F}$ corresponding to the variables in the multivariate polynomial P_s. Receiver R_i wants to evaluate P_s on its input α_i. This task becomes non-trivial if the parties want to evaluate the function in such a way that the sender learns nothing about α_i, while the receiver learns only $P_s(\alpha_1, \alpha_2, \ldots, \alpha_n)$.

A. Our Protocol

The starting point for our protocol is the Oblivious Polynomial Evaluation protocol proposed by Naor et al. [32] Limited by the OT protocol, their protocol could only involve two parties and was only applicable for the computation of univariate polynomials. In their protocol, the receiver had to hide their private data within a univariate polynomial and decompose it into data points to protect data privacy. The subsequent computation was completed by adding noise to these points and utilizing the OT protocol. We have retained some of the structural elements from [32], but to enable the computation of multivariate polynomials and to enhance transmission efficiency, we have altered the construction of the protocol. We provide a high-level description of these ideas in Fig. 4.

B. Properties of MOMPE

Correctness: In the protocol mentioned above, once the interaction between the receiver and the sender is completed, each receiver obtains K points $\mathcal{J}$. Subsequently, they use the known degree d_R of the polynomial R to interpolate and reconstruct R. They then compute $result$ to obtain the final result.

Complexity: The communication overhead of the protocol is primarily incurred during the random number negotiation and the oblivious transfer (OT). In the random number negotiation phase, each receiver must send a random number to the other receivers to generate a seed. Assuming the length of the random number is ℓ_R, and the communication cost for each party involved in one OT operation is $\ell_O T$, then the communication cost for the receivers is $(2N+n-1)\ell_R+\ell_O T$. For the sender, it is $\ell_O T$. The computational overhead of the MOMPE protocol mainly stems from the encryption, decryption, and signature verification operations during the OT phase. The time required to execute the protocol is mainly due to the communication overhead. Compared to the time required for communication overhead, the time required for computational overhead is negligible.

V. Proof of Security

The remainder of this section is devoted to a proof of the following result:

Theorem 1: We assume that he communication channels of the participants are secure. The protocol Π_{MOMPE} securely implements $\mathcal{F}_{MOMPE}$ in the OT-hybrid model, with computational security λ.

The security analysis and scheme design for the protocols involved in this paper are considered under the Universally Composable (UC) security framework in the scenario of static corruption. This means that the adversary can only determine the mode and target of corruption before the protocol execution and cannot adaptively change the target of corruption during the protocol execution. Considering the roles of the compromised participants, we analyze three scenarios: only the receivers are corrupted, only the sender is corrupted, and both some receivers and the sender are corrupted. (It is meaningless for adversary $\mathcal{A}$to corrupt all participants or none at all.) Now, let us consider what would happen if the parties were controlled by adversary $\mathcal{A}$.

Suppose only the sender is corrupted by $\mathcal{A}$. In that case, the sender can only act maliciously during the OT stage by sending incorrect pairs $\mathcal{E}$ or refusing to continue with the protocol. The goal of $\mathcal{A}$is to obtain the receiver's data α. If onlyρ receivers are corrupted, then these ρ receivers might send false or tampered messages to honest receiver in Negotiation stage. The goal of $\mathcal{A}$is to learn the data of some of the remaining $n-\rho$ receivers or to infer sender's target polynomial P_s. If the sender and ρ receivers are corrupted simultaneously, then the goal of $\mathcal{A}$is to reconstruct the data of some of the remaining $n-\rho$ receivers. For the case where both the sender and some receivers are corrupted simultaneously, since our protocol operates under the assumption that the sender and receivers will not collude, this scenario is not within the scope of discussion for this paper. The ideal function of MOMPE is presented in Fig. 5.

We construct simulator $\mathcal{S}$ for any adversary $\mathcal{A}$ who corrupts one party and any environment Z who chooses both parties' inputs and sees all outputs in order for $\mathcal{Z}$ cannot distinguish between a real execution of the protocol Π_{MOMPE} between $\mathcal{A}$ and an honest party, or a simulated execution of the protocol between $\mathcal{S}$ and the ideal functionality $\mathcal{F}_{MOMPE}$. $\mathcal{S}$ first runs a copy of $\mathcal{A}$. Each input message received by $\mathcal{S}$ from $\mathcal{Z}$ is written to the input tape of $\mathcal{A}$, and each output message that $\mathcal{A}$ writes to its output tape is copied to $\mathcal{S}$'s output tape. $\mathcal{S}$ first receives the corrupted party's from $\mathcal{Z}$ input, copies it to the input tape of $\mathcal{A}$, and then invokes $\mathcal{A}$. All communication with $\mathcal{A}$ is passed back to $\mathcal{Z}$.

Protocol Π_{MOMPE}

The degree of P_s is d_{ps}, and the security parameter is λ.

Negotiation Stage

1. Sender S (Input $P_s \in \mathbb{F}$): 0
   - Hides P_s in a $(n+1)$-variate polynomial: Draw a random univariate masking polynomial M such that $M(0) = 0$; for example, let $M(x) = \sum_{i=1}^{d_M} a_i x^i$. The degree of M is set to d_M, where $d_M = \lambda d_{ps}$, and λ is the security parameter.
   - Define an $(n+1)$-variate polynomial Q.

$$\begin{aligned} Q(x, x_1, x_2, \ldots, x_n) &= M(x) + P_s(x_1, x_2, ..., x_n) \\ &= \sum_{i=1}^{d_M} a_i x^i + \sum_{i_1=0}^{m_1} \sum_{i_2=0}^{m_2} \cdots \sum_{i_n=0}^{m_n} a_{i_1,i_2,\ldots,i_n} x_1^{i_1} x_2^{i_2} ... x_n^{i_n} \end{aligned}$$

   Obviously, we can deduce that $Q(0, x_1, x_2, ..., x_n) = P_s(x_1, x_2, ..., x_n)$. At this point, the sender has successfully hidden P_s within Q.
2. Receiver $\mathsf{R_i}$ (Input α_i):
   - Draw a random univariate polynomial S_i, ensure that $S_i(0) = \alpha_i$, the degree of P_i does not exceed λ.
   - Define

$$R(x) = Q(x, S_1(x), S_2(x), ..., S_n(x))$$

   Obviously, the degree of polynomial R in terms of $d_R = d_M = \lambda d_{ps}$.

$$\begin{aligned} R(0) &= Q\big(0, S_1(0), S_2(0), ..., S_n(0)\big) \\ &= Q(0, \alpha_1, \alpha_2, ..., \alpha_n) \\ &= 0 + P_s(\alpha_1, \alpha_2, ..., \alpha_n) \\ &= P_s(\alpha_1, \alpha_2, \ldots, \alpha_n)^{\leftarrow} \end{aligned}$$

   - Generate a random number ϵ_i, and send it to all other receivers.
   - Upon receiving the random number sent by another receiver, computes $seed = \sum_{i=1}^{n} \epsilon_i$.
   - Compute $\mathsf{H}(seed)$ and make it public. Verify whether the publicly disclosed $\mathsf{H}(seed)$ from other receiver matches your own. If they are equal, continue with the protocol. If they do not match, it indicates the presence of a dishonest party. Terminate the protocol or renegotiate the random number.
   - Using the PRF, input $seed$ to obtain a set of random numbers $\mathcal{N} = \{r_1, r_2, \ldots, r_N\}$. $\mathcal{N}$ contains $N = (d_R+1)*m$ unique and non-zero random numbers. Select the first $K = d_R + 1$ random numbers from $\mathcal{N}$ to form the set $\mathcal{T}$.
   - Defines N numbers $y_1, y_2, \ldots, y_N$, for $1 \leq i \geq N$, if $i \in \mathcal{T}$, set $y_i = S_i(r_i)$, and is a random value in $\mathbb{F}$ otherwise.
   - Sends $\mathcal{L} = (r_i, y_i)_{i=1}^{N}$ to S.

OT Stage

1. Sender S:
   - After receiving all $\mathcal{L}$ sent by the receivers, S calculates the value of $\mathcal{E} = \{r_i, R(r_i)\}_{i=1}^{N}$,

$$R(r_i) = Q\big(r_i, S_1(r_i), S_2(r_i), ..., S_n(r_i)\big) = Q(r_i, y_{i1}, y_{i2}, \ldots, y_{in})$$

2. - The receiver and sender execute an OT_n^k, for the values $\mathcal{E}$. $\mathsf{R_i}$ learns $\mathcal{J} = \{r_j, R(r_j)\}_{j \in T}$.
   - $\mathsf{R_i}$ performs interpolation using the values of $\mathcal{J}$ to obtain the polynomial $R(x)$.
   - $\mathsf{R_i}$ computes $result = R(0) = P_s(\alpha_1, \alpha_2, ..., \alpha_n)$.

Fig. 4: Full description of our MOMPE protocol.

Functionality $\mathcal{F}_{MOMPE}$

Negotiation Stage:

1. Upon S receives a message ($\mathtt{inputR}_i$, r_i) from $\mathsf{R_i}$, verify that there is no stored tuple, else ignore that message. Store r_i, and if the number of r received equals n, calculate $seed = \sum_{i=1}^{n} r_i$. The sender then send $seed$ to all receivers.

OT Stage:

1. Upon receiving a message ($\mathtt{inputS}$, P_s) from S with $P_s \in \mathbb{F}$, verify that there is no stored tuple, else ignore that message. Store P_s and send a message ($\mathtt{input}$) to $\mathcal{A}$.
2. Upon receiving a message ($\mathtt{inputS}$, α_i) from R_i with $\alpha_i \in \mathbb{F}$, verify that there is no stored tuple, else ignore that message. Store α_i and send a message ($\mathtt{input}$) to $\mathcal{A}$.
3. Upon receiving a message ($\mathtt{deliver}$, S) from $\mathcal{A}$, check if P_s and all α are stored, else ignore that message. Send ($\mathtt{delivered}$) to $\mathcal{A}$.
4. Upon receiving a message ($\mathtt{deliver}$, R_i) from $\mathcal{A}$, check if P_s and all α_i are stored, else ignore that message. Check if the number of α is n, else wait. Set $result = P_s(\alpha_1, \alpha_2, \ldots, \alpha_n)$ and Send ($\mathtt{output}$, $result$) to all R.

Fig. 5: Ideal functionality for MOMPE.

A. Corrupted Sender

In the following we present a simulator $\mathcal{S}_s$ which provides a computationally indistinguishable simulation of Π_{MOMPE} to a malicious Sender $\mathcal{A}_s$(cf. Fig. 6). We now discuss the indistinguishability between the real and ideal execution for the environment $\mathcal{Z}$. Recall that $\mathcal{Z}$ can choose the inputs for all participants (P_s and α). The perspective of $\mathcal{Z}$ in the real world includes this information: the negotiation stage where the receiver publicly reveals $\mathsf{H}(seed)$ to verify the $seed$'s validity, the OT stage where the attacker $\mathcal{A}_s$ receives information, and the receiver's output $result$. In the ideal world, the publicly verifiable $\mathsf{H}(seed)$ is computed by the same Hmethod, generated from a random value chosen by the simulator. The output $result$ for each honest receiver is computed by the ideal functionality based on the input P_s^* provided by $\mathcal{S}_s$ and the honest input α.

Proof 1: In both worlds, due to the properties of the hash function, $\mathsf{H}(seed)$ and $\mathsf{H}(\widehat{seed})$ are indistinguishable across the two worlds. Moreover, since the receiver is honest, the rest of the protocol entirely determines the result. Therefore, we only need to prove the indistinguishability between the real world values $\widehat{\mathcal{L}} = \{(r_i, \widehat{y_i})\}_{i=1}^{N}$ and the ideal world $\mathcal{L}$. According to NPA, $\widehat{\mathcal{L}}$ and $\mathcal{L}$ are indistinguishable. Thus, for any adversary $\mathcal{A}$ that corrupts the sender, $\mathcal{Z}$ cannot distinguish between interacting with the participants in the real world

Simulator $\mathcal{S}_s$

1. $\mathcal{S}_s$ selects a random number $\widehat{\epsilon_i}$ to simulate an honest receiver R_i during the negotiation stage, which is used to generate $\widehat{seed}$. $\mathcal{S}_s$ then simulates the sending process by sending $\widehat{\epsilon_i}$ to the other receivers. Afterward, $\mathcal{S}_s$ calculates $seed$ and publicly discloses $\mathsf{H}(\widehat{seed})$ to verify the legality of the seed. If the $\widehat{seed}$ is deemed legal, the protocol continues; otherwise, it is terminated.
2. $\mathcal{S}_s$ utilizes $\widehat{\epsilon_i}$ to generate a set of random numbers $\widehat{N} = \{r_1, r_2, \ldots, r_N\}$.
3. $\mathcal{S}_s$ selects random numbers $\widehat{y_i}$ to simulate y_i and generates $\widehat{\mathcal{L}} = \{(r_i, \widehat{y_i})\}_{i=1}^{N}$, sending it to $\mathcal{A}_s$.
4. $\mathcal{S}_s$ learns the input of $\mathcal{A}_s$ from $\mathcal{Z}$, which is P_s^* and $\mathcal{E}_i^* = \{(r_i, R(r_i))\}_{i=1}^{N}$.
5. $\mathcal{S}_s$ sends P_s^* to $\mathcal{F}_{MOMPE}$ just as an honest sender would.

Fig. 6: Simulator against a corrupted sender in Π_{MOMPE}.

of the Π_{MOMPE} or interacting with $\mathcal{F}_{MOMPE}$ in the ideal world.

B. Corrupted Receiver

In the following we present a simulator $\mathcal{S}_R$ which provides a computationally indistinguishable simulation of Π_{MOMPE} to a malicious Sender $\mathcal{A}_R$(cf. Fig. 7). We now discuss the indistinguishability between the real and ideal execution for the environment $\mathcal{Z}$. The perspective of $\mathcal{Z}$ in the real world includes the following information: in the negotiation stage, the random numbers ϵ sent between the receivers, as well as the publicly revealed $\mathsf{H}(seed)$ to verify the $seed$'s validity; in the OT stage, the information received by the attacker $\hat{\mathcal{J}} = \{r_j, R(r_j)\}_{j \in \mathcal{T}}$, and the output $result$ of the honest receivers. In the ideal world, all receivers' $\widehat{\epsilon_i}$ are randomly generated by the simulator, the publicly verifiable $\mathsf{H}(seed)$ is computed by the same hash method, and $R(r_i)$ in $\hat{\mathcal{E}}$ is generated from random values chosen by the simulator. The output $result$ for each honest receiver is computed by the ideal functionality based on the input P_s^* provided by $\mathcal{S}_R$ and the honest input α.

Proof 2: In both the real and ideal worlds, ϵ_i and $\widehat{\epsilon_i}$ are randomly generated uniform random numbers, so these values have the same distribution and are indistinguishable across the two worlds. Due to the properties of the hash function, $\mathsf{H}(seed)$ and $\mathsf{H}(\widehat{seed})$ are indistinguishable in both worlds. Therefore, we only need to prove the indistinguishability between the real world values $\mathcal{J} = \{r_j, R(r_j)\}_{j \in \mathcal{T}}$ and the ideal-world $\hat{J}$.

Because the OT protocol is UC-secure, and the receiver's input depends on the $seed$, to prove the indistinguishability of $\mathcal{J}$, it is sufficient to demonstrate that the sender's input is indistinguishable. If the sender's input is indistinguishable,

Simulator $\mathcal{S}_R$

1. During Negotiation Stage: $\mathcal{S}_R$ receives the input ϵ_j from $\mathcal{A}_R$. For the other honest R_i, S_R selects a random number $\widehat{\epsilon_i}$ to simulate the random number used by the honest participant R_i for generating the random seed in the random number negotiation stage. $\mathcal{S}_R$ then simulates the negotiation process, sending ϵ_i to the other receivers. Subsequently, upon receiving $seed$, $\mathcal{S}_R$ computes $\mathsf{H}\{\widehat{seed}\}$ and reveals it publicly to verify the legitimacy of $seed$. If the $seed$ is deemed legitimate, the protocol continues; otherwise, it is terminated.
2. $\mathcal{S}_R$ In the OT stage, S_R learns the input α_j of $\mathcal{A}_R$ from $\mathcal{Z}$ and sends it to $\mathcal{F}_{MOMPE}$. $\mathcal{S}_R$ then receives $result$ from $\mathcal{F}_{MOMPE}$.
3. Upon $\mathcal{S}_R$ receives the input $\mathcal{L}_j^* = \{(r_i, \widehat{y}_l)\}_{i=1}^N$ from $\mathcal{A}_R$, it generates a random polynomial $\widehat{R(x)}$ satisfying $\widehat{R(0)} = result$, Using the $seed$, generates a set of random numbers $\widehat{\mathcal{T}}$. In $\widehat{\mathcal{E}} = \{r_i, R(r_i)\}_{i=1}^N$, for $r_i \in \widehat{\mathcal{T}}$, set $R(r_i) = \hat{R}(r_i)$. For $r_i \notin \hat{\mathcal{T}}$, set $R(r_i)$ to a random value in $\mathbb{F}$. After performing the OT with $\mathcal{A}_R$, $\mathcal{A}_R$ obtains $\widehat{\mathcal{J}} = \{r_j, R(r_j)\}_{j \in \widehat{\mathcal{T}}}$.

Fig. 7: Simulator against a corrupted receiver in Π_{MOMPE}.

then the output received by the receiver is also indistinguishable, which establishes the indistinguishability of $\mathcal{E}$. In both the real and ideal worlds, the polynomials $R(x)$ and $\widehat{R(x)}$ pass through the point $(0, result)$. Assumption APA states that the real-world $\mathcal{E}$ and the ideal-world $\hat{\mathcal{E}}$ are indistinguishable. Since $\mathcal{E}$ is the output of a UC-secure OT protocol, it is also indistinguishable thus the real-world $\mathcal{J} = \{r_j, R(r_j)\}_{j \in \mathcal{T}}$ and the ideal-world $\hat{\mathcal{J}}$ are indistinguishable.

C. Corrupted ρ Receiver

In the negotiation stage, the agreement and verification of random numbers are identical in both the real and ideal worlds. The case where multiple receivers are corrupted during the OT stage is analogous to the case where a single receiver is corrupted. Thus, when multiple receivers are compromised, the real and ideal worlds are the same.

VI. Implementation and Evaluation

We have implemented our protocol and evaluated its computational efficiency in local area network (LAN) and wide area network (WAN) settings. In the LAN setting, the network bandwidth is 1 Gbps, the latency is less than 1 ms, and the machine running the protocol has ten cores, each with a base frequency of 3.7 GHz. In the WAN setting, the bandwidth is five Mbps, the latency is 75 ms, and the machines running the protocol all have eight cores, each operating at a frequency of 2.2 GHz. Due to pipelining, we did not observe any issues in terms of memory usage. All reported timing results are calculated as the average of 10 executions. Our code is written in Python, and the OT scheme used is the one proposed in reference [38].

Our protocol, distinct from the traditional two-party OPE protocol, is a secure computation technique involving multiple parties. Therefore, we compare it with OPE protocols and secure multiparty computation protocols allowing multiple-party participation. Considering the availability of experimental code for comparison, we also compare our protocol with the most efficient maliciously secure multiparty computation protocol available, Overdrive [39], [40].

A. Compared with Overdrive

1) Input: In the performance comparison with the Overdrive protocol, we conducted three experiments, each with three participants holding secrets. Since our MOMPE is limited to polynomial evaluation, we set the target computation methods as three different polynomials. In Exp1, the target polynomial only requires addition; in Exp2, the target polynomial only requires multiplication; and in Exp3, both multiplication and addition are required. The inputs used by the participants in the MOMPE protocol are shown in Table I. In Exp1, Exp2, and Exp3, the polynomials held by the sender are $P1_s$, $P2_s$, and $P3_s$, respectively, and the random numbers α_1, α_2, α_3 are used as the secret data held by the receivers R_1, R_2, and R_3, respectively. The comparison of the input data used is listed in Table I.

TABLE I: The input of each participant when conducting an experimental comparison with Overdrive

	polynomial	α_1	α_2	α_3
Ext1	$P1_s(x_1, x_2, x_3) = x_1 + x_2 + x_3$	20	35	6
Ext2	$P2_s(x_1, x_2, x_3) = x_1 * x_2 * x_3$	7	26	93
Ext3	$P3_s(x_1, x_2, x_3) = (x_1 + x_2 + x_3)^3$	3	61	24

2) Running time: The time consumption of our protocol includes data preprocessing time and communication time. The time consumption calculation for the Overdrive scheme is also the same. Table II records the time consumption of our protocol and the Overdrive protocol in both LAN and WAN environments. From Table II, we can observe that our scheme takes 36.5% of the time required by the Overdrive scheme to compute $P1_c$ in the LAN, 16.8% of the time required by the Overdrive scheme to compute $P2_c$, and 27% of the time required by the Overdrive scheme to compute $P3_c$. In the WAN environment, our MOMPE protocol takes 5.5% of the time required by the Overdrive scheme to compute $P1_c$, 2.4% of the time required by the Overdrive scheme to compute P_c^2, and 7.2% of the time required by the Overdrive scheme to compute $P3_c$. These data demonstrate that our scheme is much more efficient than the Overdrive scheme in computing polynomials. The different percentages of time consumption for the three experiments are because $P1_c$ only involves addition, $P2_c$ involves multiplication, and $P3_c$ involves both addition and multiplication. As for the specific calculation

process, since the two protocols employ completely different calculation methods, there is no need for further analysis.

B. Compared with OPE

When comparing the performance with OPE, due to the limitations of OPE protocols allowing only two participating parties, we introduce a third party in the negotiation phase of the MOMPE protocol to facilitate a fair comparison. This third party sends random numbers to the receiver during the negotiation stage, mimicking an honest receiver, but does not participate in the subsequent OT stage. The input data used for computation is presented in Table III. We conduct three experiments, with $P4_s$, $P5_s$ and $P6_s$ as the target polynomials to be evaluated in each experiment. The receiver's private data consists of random numbers. Exp4 involves only addition, Exp5 only multiplication, and Exp6 includes addition and multiplication.

TABLE III: The input of each participant when conducting an experimental comparison with OPE

	Sender	Receiver
Ext4	$P4_s(x) = x + 7$	78
Ext5	$P5_s(x) = 3x^2$	45
Ext6	$P6_s(x) = 7x^4 + 2x^2 + 7$	13

1) Running time: Our protocol's time expenditure encompasses data preprocessing and communication times, which are consistent with the OPE scheme calculation methodology. Table IV documents the time consumption for our protocol and the OPE protocol to conduct three separate experiments within both LAN and WAN. It also records the time our protocol and the OPE protocol spent under the malicious model in both LAN and WAN environments. The time consumption of our MOMPE protocol is similar to that of the OPE protocol, regardless of whether it is deployed in a LAN or a WAN. Besides the time loss during the OT phase, our protocol incurs an additional cost due to the negotiation stage. Theoretically, the time expenditure for the sender remains the same, whereas the receiver experiences an extra overhead from sending a random number and verifying the seed. Consequently, the time consumption of the two protocols is comparable.

C. Scalability

We assessed the time consumption of our protocol in a LAN environment as the number of receivers increased. To better observe the trend, we utilized symmetric polynomials as the sender's input, and for each increment in the number of receivers, an additional variable was added to the sender's input. For instance, the polynomial is $P7_s(x_1, x_2) = x_1 + x_2$ for two receivers, and for three receivers, it is $P8_s(x_1, x_2, x_3) = x_1+x_2+x_3$. We conducted five sets of experiments, measuring the time expenditure for both the sender and the receivers in the MOMPE protocol as the number of receivers ranges from 2 to 6. The inputs for the parties involved in the protocol are detailed in Table V.

TABLE V: The input of MOMPE Protocol

	Polynomial	α_1	α_2	α_3	α_4	α_5	α_6
EXP7	$P7_s$	20	35				
EXP8	$P8_s$	7	26	93			
EXP9	$P9_s$	3	61	24	59		
EXP10	$P10_s$	56	81	-41	32	28	
EXP11	$P11_s$	46	50	-82	37	16	-23

1) Running time: Fig. 8 records the receiver's time consumption trend as the number of receivers increases. It can be observed that with the increase in the number of receivers, the time expenditure for each receiver gradually rises. Approximately 20 ms is added to the time consumption of a single receiver for each additional receiver. From the execution process of the protocol, it is evident that the primary reason lies in the negotiation phase. With each additional receiver during the negotiation phase, the receiver needs to engage in an extra round of communication, increasing time consumption. Under other constant conditions, the time expenditure for the receiver in the OT phase does not increase. Fig. 9 records the sender's time consumption trend as the number of receivers increases. It can be observed that with the rise in the number of receivers, the sender's time expenditure also gradually increases. However, the change is relatively subtle compared to that of the receivers. Analyzing the protocol's execution process, we can infer that the sender's increased time consumption primarily stems from the augmented local computation. Since the sender processes the data from the receivers in parallel, the time expenditure on communication does not vary with the increase in the number of receivers.

VII. CONCLUSION

This paper introduces the Multiparty Oblivious Multivariate Polynomial Evaluation (MOMPE) protocol, which overcomes the limitations of traditional Oblivious Polynomial Evaluation (OPE) protocols confined to two-party secure computation. MOMPE facilitates the secure computation of multivariate polynomials among multiple parties, where the sender possesses a multivariate polynomial P, and n receivers each hold a secret value α_i corresponding to the variables in P. The protocol ensures that, upon completion, the receivers learn the result $P(\alpha_1, \alpha_2, \ldots, \alpha_n)$, while the sender learns nothing. Under the assumption that the sender and receivers are not simultaneously corrupted by an attacker, the protocol is proven secure within the Universal Composability (UC) framework, ensuring robust security in complex multiparty interactive environments. The protocol has been implemented and its performance evaluated through experimentation. Comparative analysis with the state-of-the-art Overdrive protocol demonstrates that the time consumption of the MOMPE protocol is significantly lower than that of Overdrive, while the time consumption for the participating parties remains comparable to that of OPE. This indicates that MOMPE achieves a broader range of application scenarios while maintaining efficiency.

TABLE II: Compared the running time of our MOMPE and Overdrive in the LAN and WAN settings.

		LAN			WAN		
	Polynomial	Our MOMPE	Overdrive	Speedup	Our MOMPE	Overdrive	Speedup
Exp1	$P1_s$	95ms	96ms	2.7	425ms	7663ms	18
Exp2	$P2_s$	36ms	214ms	5.9	420ms	8294ms	19.7
Exp3	$P3_s$	60ms	215ms	3.6	601ms	8315ms	13.8

TABLE IV: Compared the running time of our MOMPE and OPE in the LAN and WAN settings.

		LAN			WAN		
	Polynomial	Our MOMPE	OPE	Slowdown	Our MOMPE	OPE	Slowdown
Exp4	$P4_s$	33ms	30ms	1.1	426ms	403ms	1.05
Exp5	$P5_s$	42ms	39ms	1.08	513ms	489ms	1.05
Exp6	$P6_s$	76ms	70ms	1.09	610ms	543ms	1.12

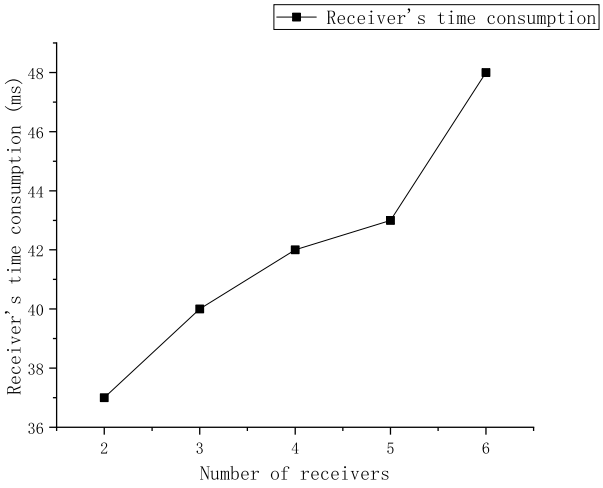

Fig. 8: Receiver's time consumption

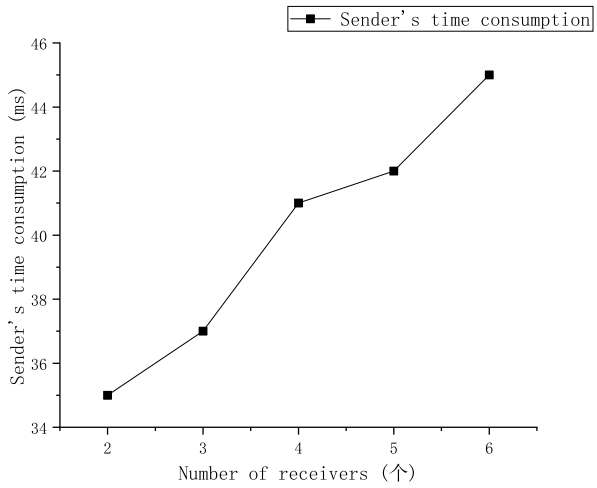

Fig. 9: Sender's time consumption

Furthermore, the MOMPE protocol has extensive potential applications in fields such as privacy-preserving machine learning and collaborative data analysis, where multiple parties need to compute and analyze data together while protecting their respective private information. The secure and efficient method offered by the MOMPE protocol allows parties to utilize data for analysis and modeling without revealing sensitive information, thereby opening up new possibilities for privacy-preserving data collaboration and analysis. Future research could focus on optimizing the negotiation process to reduce overhead, enhancing the protocol's resilience against insider collusion, expanding its applications across various industries, and advancing standardization efforts. Additionally, improving the practical value and security of the MOMPE protocol through theoretical proofs and performance enhancements is another key area for development.

REFERENCES

[1] C. Juvekar, V. Vaikuntanathan, and A. Chandrakasan, "{GAZELLE}: A low latency framework for secure neural network inference," in *27th USENIX security symposium (USENIX security 18)*, 2018, pp. 1651–1669.

[2] P. Mohassel and P. Rindal, "Aby3: A mixed protocol framework for machine learning," in *Proceedings of the 2018 ACM SIGSAC conference on computer and communications security*, 2018, pp. 35–52.

[3] W. Zheng, R. A. Popa, J. E. Gonzalez, and I. Stoica, "Helen: Maliciously secure coopetitive learning for linear models," in *2019 IEEE symposium on security and privacy (SP)*. IEEE, 2019, pp. 724–738.

[4] N. Kumar, M. Rathee, N. Chandran, D. Gupta, A. Rastogi, and R. Sharma, "Cryptflow: Secure tensorflow inference," in *2020 IEEE Symposium on Security and Privacy (SP)*, 2020, pp. 336–353.

[5] Z. Huang, W.-j. Lu, C. Hong, and J. Ding, "Cheetah: Lean and fast secure {Two-Party} deep neural network inference," in *31st USENIX Security Symposium (USENIX Security 22)*, 2022, pp. 809–826.

[6] D. Rathee, A. Bhattacharya, R. Sharma, D. Gupta, N. Chandran, and A. Rastogi, "Secfloat: Accurate floating-point meets secure 2-party computation," in *2022 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2022, pp. 576–595.

[7] W. Zheng, R. Deng, W. Chen, R. A. Popa, A. Panda, and I. Stoica, "Cerebro: A platform for Multi-Party cryptographic collaborative

learning," in *30th USENIX Security Symposium (USENIX Security 21)*. USENIX Association, Aug. 2021, pp. 2723–2740. [Online]. Available: https://www.usenix.org/conference/usenixsecurity21/presentation/zheng

[8] C. Brunetta, G. Tsaloli, B. Liang, G. Banegas, and A. Mitrokotsa, "Non-interactive, secure verifiable aggregation for decentralized, privacy-preserving learning," in *Australasian Conference on Information Security and Privacy*. Springer, 2021, pp. 510–528.

[9] D. W. Archer, D. Bogdanov, Y. Lindell, L. Kamm, K. Nielsen, J. I. Pagter, N. P. Smart, and R. N. Wright, "From keys to databases—real-world applications of secure multi-party computation," *The Computer Journal*, vol. 61, no. 12, pp. 1749–1771, 2018.

[10] M. Naor and B. Pinkas, "Oblivious transfer and polynomial evaluation," in *Proceedings of the thirty-first annual ACM symposium on Theory of computing*, 1999, pp. 245–254.

[11] C. Hazay and Y. Lindell, "Efficient oblivious polynomial evaluation with simulation-based security," *Cryptology ePrint Archive*, 2009.

[12] C. Hazay, "Oblivious polynomial evaluation and secure set-intersection from algebraic prfs," *Journal of Cryptology*, vol. 31, no. 2, pp. 537–586, 2018.

[13] M. Izabachène, A. Nitulescu, P. de Perthuis, and D. Pointcheval, "Myope: malicious security for oblivious polynomial evaluation," in *International Conference on Security and Cryptography for Networks*. Springer, 2022, pp. 663–686.

[14] H. Gajera, M. Giraud, D. Gérault, M. L. Das, and P. Lafourcade, "Verifiable and private oblivious polynomial evaluation," in *IFIP International Conference on Information Security Theory and Practice*. Springer, 2019, pp. 49–65.

[15] G. Ateniese, Ö. Dagdelen, I. Damgård, and D. Venturi, "Entangled cloud storage," *Future Generation Computer Systems*, vol. 62, pp. 104–118, 2016.

[16] Y. Liu, Q. Wang, and S.-M. Yiu, "Blind polynomial evaluation and data trading," in *International Conference on Applied Cryptography and Network Security*. Springer, 2021, pp. 100–129.

[17] A. B. Arie and T. Tassa, "Distributed protocols for oblivious transfer and polynomial evaluation," *Cryptology ePrint Archive*, 2024.

[18] T. Tassa, A. Jarrous, and Y. Ben-Ya'akov, "Oblivious evaluation of multivariate polynomials," *Journal of Mathematical Cryptology*, vol. 7, no. 1, pp. 1–29, 2013.

[19] M. O. Rabin, "How to exchange secrets with oblivious transfer," *IACR Cryptol. ePrint Arch.*, vol. 2005, p. 187, 2005. [Online]. Available: https://api.semanticscholar.org/CorpusID:15222660

[20] W.-G. Tzeng, "Efficient 1-out-n oblivious transfer schemes," in *Public Key Cryptography: 5th International Workshop on Practice and Theory in Public Key Cryptosystems, PKC 2002 Paris, France, February 12–14, 2002 Proceedings 5*. Springer, 2002, pp. 159–171.

[21] M. Naor and B. Pinkas, "Computationally secure oblivious transfer," *Journal of Cryptology*, vol. 18, pp. 1–35, 2005.

[22] W.-G. Tzeng, "Efficient 1-out-n oblivious transfer schemes," in *Public Key Cryptography: 5th International Workshop on Practice and Theory in Public Key Cryptosystems, PKC 2002 Paris, France, February 12–14, 2002 Proceedings 5*. Springer, 2002, pp. 159–171.

[23] Y.-J. Xu, S.-D. Li, and Z.-H. Chen, "Efficient oblivious transfer protocol based on bilinear pairing," *Jisuanji Gongcheng/ Computer Engineering*, vol. 39, no. 6, 2013.

[24] Y. Xu, D. Li, D. Wang *et al.*, "Oblivious transfer based on elliptic curve public key cryptosystems," *Computer Science*, vol. 40, no. 12, pp. 186–191, 2013.

[25] J. Lai, Y. Mu, F. Guo, R. Chen, and S. Ma, "Efficient k-out-of-n oblivious transfer scheme with the ideal communication cost," *Theoretical Computer Science*, vol. 714, pp. 15–26, 2018.

[26] P. Elias, "List decoding for noisy channels," 1957.

[27] Y. Ishai, M. Prabhakaran, and A. Sahai, "Secure arithmetic computation with no honest majority," in *Theory of Cryptography: 6th Theory of Cryptography Conference, TCC 2009, San Francisco, CA, USA, March 15-17, 2009. Proceedings 6*. Springer, 2009, pp. 294–314.

[28] M. Franklin and M. Yung, "Communication complexity of secure computation," in *Proceedings of the twenty-fourth annual ACM symposium on Theory of computing*, 1992, pp. 699–710.

[29] A. Kiayias and M. Yung, "Cryptographic hardness based on the decoding of reed–solomon codes," *IEEE Transactions on Information Theory*, vol. 54, no. 6, pp. 2752–2769, 2008.

[30] D. Bleichenbacher and P. Q. Nguyen, "Noisy polynomial interpolation and noisy chinese remaindering," in *International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2000, pp. 53–69.

[31] D. Boneh, "Finding smooth integers in short intervals using crt decoding," in *Proceedings of the thirty-second annual ACM symposium on Theory of computing*, 2000, pp. 265–272.

[32] M. Naor and B. Pinkas, "Oblivious polynomial evaluation," *SIAM Journal on Computing*, vol. 35, no. 5, pp. 1254–1281, 2006.

[33] S. Ghosh, J. B. Nielsen, and T. Nilges, "Maliciously secure oblivious linear function evaluation with constant overhead," in *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, 2017, pp. 629–659.

[34] L. Cianciullo and H. Ghodosi, "Efficient information theoretic multi-party computation from oblivious linear evaluation," in *IFIP International Conference on Information Security Theory and Practice*. Springer International Publishing Cham, 2018, pp. 78–90.

[35] I. Damgård, H. Haagh, M. Nielsen, and C. Orlandi, "Commodity-based 2pc for arithmetic circuits," in *Cryptography and Coding: 17th IMA International Conference, IMACC 2019, Oxford, UK, December 16–18, 2019, Proceedings 17*. Springer International Publishing, 2019, pp. 154–177.

[36] O. Goldreich, *Foundations of cryptography: volume 2, basic applications*. Cambridge university press, 2009.

[37] R. Canetti, "Universally composable security: A new paradigm for cryptographic protocols," in *Proceedings 42nd IEEE Symposium on Foundations of Computer Science*. IEEE, 2001, pp. 136–145.

[38] T. Chou and C. Orlandi, "The simplest protocol for oblivious transfer," in *Progress in Cryptology–LATINCRYPT 2015: 4th International Conference on Cryptology and Information Security in Latin America, Guadalajara, Mexico, August 23-26, 2015, Proceedings 4*. Springer, 2015, pp. 40–58.

[39] M. Keller, V. Pastro, and D. Rotaru, "Overdrive: Making spdz great again," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer International Publishing Cham, 2018, pp. 158–189.

[40] "GitHub - data61/MP-SPDZ: Versatile framework for multi-party computation — github.com," https://github.com/data61/MP-SPDZ, [Accessed 19-04-2024].