Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 247 (2024) 1267–1276

Procedia Computer Science

www.elsevier.com/locate/procedia

The 11th International Conference on Applications and Techniques in Cyber Intelligence

Secure Multi-party Key-Value Data Statistics Against Malicious Models

Jilong Zhao[a,b], Yuhao Zhang[a,b], Changhui Hu[a,b,*]

[a]School of Cyberspace Security Hainan University, haikou, 570000,China
[b]Key Laboratory of Internet Information Retrieval of Hainan Province, haikou, 570000,China

Abstract

Recently, key-value storage has become the primary paradigm for managing associative data due to its simplicity and efficiency, and is widely used in various fields. However, key-value data often contains sensitive information such as identity, health data, and transaction records. Once this data is collected, the risk of privacy breaches for users significantly increases. Differential privacy is commonly used to compute aggregated statistical data while protecting user privacy, but traditional differential privacy relies on a trusted third party, which is often not available in reality. Secure Multi-Party Computation (MPC) can analyze key value data statistics without the necessity of a central trusted authority, but it involves high computational and communication costs and risks from malicious participants. To address these issues, we design an MPC scheme for malicious models based on the SPDZ protocol. This scheme combines secure multi-party computation and differential privacy to securely compute the frequency and mean of key value data without a trusted third party, achieving differential privacy protection comparable to centralized differential privacy. Upon security analysis, it has been established that our protocol is secure within the Universal Composability (UC) framework. Once a violation of the protocol by any participant is detected, the protocol will terminate, providing resistance against disruptions from malicious participants. Comparative experiments demonstrate that this scheme performs comparably under a malicious security model as it does under a semi-honest security model, verifying its feasibility and effectiveness.

© 2024 The Authors. Published by Elsevier B.V.
This is an open access article under the CC BY-NC-ND license (https://creativecommons.org/licenses/by-nc-nd/4.0)
Peer-review under responsibility of the scientific committee of the 11th International Conference on Applications and Techniques in Cyber Intelligence

Keywords: key value statistics; multi-party computation; differential privacy; malicious model

* Corresponding author. Tel.: +86-898-66252802; fax: +86-898-66252802.
E-mail address: chu@hainanu.edu.cn

1877-0509 © 2024 The Authors. Published by Elsevier B.V.
This is an open access article under the CC BY-NC-ND license (https://creativecommons.org/licenses/by-nc-nd/4.0)
Peer-review under responsibility of the scientific committee of the 11th International Conference on Applications and Techniques in Cyber Intelligence
10.1016/j.procs.2024.10.152

1. Introduction

As the widespread collection and storage of big data escalate, the risks of data breaches and privacy invasions have surged, making privacy protection particularly crucial. In recent years, key-value storage [1] has become the primary management paradigm for associated data. This data structure is extensively utilized in numerous domains, including mobile applications, e-commerce, healthcare platforms, and machine learning, owing to its simplicity and efficiency. However, key-value pairs may contain users' personal sensitive information, such as identity details and transaction records. Once these data are collected and stored on the server side, users lose control over their privacy, facing the risk of data privacy breaches.

Traditional centralized differential privacy [2] for protecting key value data rely on a trusted third party to coordinate and manage data processing. However, in reality, an absolutely trusted central authority does not exist. A viable method to address this issue is the use of local differential privacy. The PrivKV local differential privacy protection method proposed by Ye et al. [3] suffers from reduced accuracy in frequency and mean estimates due to the frequent partitioning of privacy costs [4]. The KVOH method introduced by Sun et al. [5] neglects the impact of the entire value domain of the key-value, also facing issues with low accuracy in frequency and mean estimates. Furthermore, when applying differential privacy schemes to analyze key-value data, the correlations between key-values might be compromised, resulting in reduced accuracy in the estimated frequencies and means [6]. Compared to single differential privacy schemes, MPC schemes have significantly high computational and communication costs [7]. Additionally, malicious participants may breach the protocol [8], attempting to tamper with input or output data, disrupt the computation process, or steal other participants' private information.

To address the challenges mentioned above, we design an MPC scheme for a malicious model based on SPDZ [9]. By encoding key-value pairs into arrays and securely calculating the true frequencies and means of key-value pairs using secure multi-party computation, the scheme validates the integrity and accuracy of the results with Message Authentication Codes (MAC) integrated into the SPDZ protocol. After verifying the results, differential privacy noise is added to achieve the frequency and mean estimates. By comparing MPC schemes under different protocols, this scheme demonstrates effectiveness comparable to those under semi-honest security models when implemented in a malicious security model. The main contributions of this scheme can be summarized as follows:

- The fusion of secure multi-party computation with differential privacy yields privacy protection comparable to centralized differential privacy, eliminating the necessity for a central trusted authority.
- Data security for all participants is ensured through abortive malicious security features, which terminate the protocol immediately upon detecting malicious activities or protocol violations.
- The overall computational cost in the MPC process is reduced by selecting a subset of computation parties, using random matrix selection, to participate in the computation.

2. Preliminaries

2.1. SPDZ

Secure Multi-party Computation [7] is designed to enable multiple parties to collaboratively compute a function without revealing their individual inputs. This means that even though the parties may not trust each other, they can still ensure the privacy of the computation process and the accuracy of the results. MPC typically employs the UC framework [10], which provides a robust security model for secure protocols. The UC framework allows secure protocols to be modularly combined in complex systems, building more intricate and multifunctional computing systems while maintaining overall security. The SPDZ protocol [9] is a significant protocol in the MPC field, which implements secure multi-party computation based on technologies like homomorphic encryption [12], oblivious transfer [13], and secret sharing [11]. It supports secure computation over finite fields.

The SPDZ protocol primarily consists of two stages [20]: the preprocessing phase and the online phase.The preprocessing phase primarily involves generating secret-shared values including multiple sets of Beaver triples $([a],[b],[c])$, secret-shared random numbers, and a global MAC key α using Oblivious Transfer [14] or Homomorphic Encryption [9]. These auxiliary data, required for the online phase, are independent of the input data used during the online computation phase, allowing them to be computed in advance. In the online phase, various

addition and multiplication calculations are completed based on secret sharing, and MACs are used to verify intermediate values and results to ensure privacy and correctness. Under the SPDZ protocol framework, a certified secret value $x \in \mathsf{F}$ is defined as follows:

$$[x] = (x_1, ...x_n, m_1^{(x)}, ..., m_n^{(x)}, \Delta_1, ..., \Delta_n), \quad (1)$$

where Δ is the global key, the MAC value $m^{(x)} = x\Delta$, and each participant P_i holds an array of additive secret shares $[x]_i = (x_i, m_i^{(x)}, \Delta_i)$, satisfying:

$$x = \sum_{i=1}^{n} x_i, x\Delta = \sum_{i=1}^{n} m_i^{(x)}, \Delta = \sum_{i=1}^{n} \Delta_i. \quad (2)$$

As shown in Fig. 1 of the MAC check protocol [9], when performing MAC authentication on the secret value $[x]$, all participants P_i individually compute $\sigma_i = [m_i^{(x)} - x\Delta_i]$ and broadcast their secret-shared portions σ_i. They then check if $\sigma_1 + \sigma_2 + ... + \sigma_n = 0$ holds true. If it does, the process continues; otherwise, the program is aborted, meaning an incorrect MAC value would cause the check to fail.

Protocol MAC check

Each Computation party P_i uses $x_i, m_i^{(x)}, \Delta_i$ in the following way:

1. Compute $\sigma_i \leftarrow m_i^{(x)} - \Delta_i x_i$
2. Call F_{Commit} with $(Commit, \sigma_i)$ to receive handle τ_i.
3. Broadcast σ_i to all parties by calling F_{Commit} with $(Open, \tau_i)$.
4. If $\sigma_1 + \sigma_2 + ... + \sigma_n \neq 0$ then abort and output $\perp$; otherwise continue.

Fig. 1. MAC Checking Protocol

Since secret addition is linear, two secret-shared parts can perform addition calculations locally [22]. However, when multiplying two secret values, $[x]$ and $[y]$, participant interaction is necessary to verify and secure the computation. Calculating $[x] \cdot [y]$ requires the use of a Beaver triplet (a, b, c), where $c = a \cdot b$. Each participant possesses a secret share of the Beaver triple [15] $([a], [b], [c])$. First, all participants collaboratively compute $[\varepsilon] = [x] - [a] = [x - a]$ and $[\rho] = [y] - [b] = [y - b]$; then $[z] = [x \cdot y] = [c] + \varepsilon[b] + \rho[a] + \varepsilon\rho$ is computed locally [20].

In the UC framework, the SPDZ protocol has been proven to be secure and can be combined with other secure protocols [16]. SPDZ employs Message Authentication Codes to check if participants are computing honestly. If a breach of protocol is detected, the protocol execution is halted immediately to prevent the leakage of private data, thereby ensuring abort security [24]. With n participants, it can withstand up to $n-1$ participants maliciously violating the protocol or conspiring together. To enhance computational efficiency, the SPDZ protocol allows for the linear combination of multiple secret-shared values before outputting the final result, and conducts a one-time batch MAC verification [21]. This batch processing technique significantly reduces the frequency of MAC verifications, thus improving the overall computational efficiency.

3. Problem statement

3.1. Threat model

This paper consider m honest data parties $DP_i, i \in [1, ..., m]$ and $\ell(\ell \geqslant 3)$ untrusted computation parties $CP_i, i \in [1, 2, 3, ..., \ell]$, where untrusted servers can serve as computation parties [23], with up to $\ell - 1$ of these computation parties potentially being malicious.

Each data party $DP_i, i \in [1, ...m]$ possesses a collection of key-value pairs, S_i, represented as $< k_{ij}, v_{ij} >$, where each data party $DP_i, i \in [1, ...m]$ holds the global j-th key-value pair, with each $k_{ij} \in K$ and each $v_{ij} \in V$. This can be represented as $< k_j, v_j >$ for the global j-th pair of key-values. The collections of key-value pairs S_i owned by all data parties form a multiset S. Data parties send their data to a collection of untrusted servers,, which collectively perform statistical computations on the data. After receiving data shared by data parties, the computation parties perform secure multiparty computation under a malicious model, and add differential privacy noise to the final

result to protect privacy. During the computation process, participants may attempt to violate the protocol, tamper with input or output data, disrupt the computation process, or steal other participants' private information.

3.2. Problem description and objective design

This paper primarily focuses on frequency estimation and mean estimation of key value data held by multiple data parties. In this paper, frequency estimation denotes the count of occurrences of a key k within the set S, denoted as q_k, with the frequency vector for all keys represented as $q(S)$. Mean estimation refers to the average value of all values associated with key k in the set S, denoted as μ_k, i.e.,

$$\mu_k(S) = \frac{\sum_{A_k} v_{ij}}{q_k(S)}, \tag{3}$$

where A_k represents the collection of all key-value pairs associated with key k, and we use $\mu(S)$ to denote the mean vector for all keys.We are committed to achieving the following objectives:

- Leverage MAC to guarantee the precise computation of key-value pair frequencies and averages, even when confronted with malicious actors.
- Attain privacy protection on par with differential privacy by implementing a Multi-Party Computation (MPC) scheme, eliminating the necessity for a trusted third party.
- Lower the overall computational cost of MPC process by employing random matrices to select a subset of computation parties for participation.

4. The proposed protocol

4.1. Overview

This protocol for differential privacy statistics of key-value data is divided into four stages: *i*) *initialization phase*; *ii*) *offline phase*; *iii*) *data collection phase*; *iv*) *online computation phase*.

To simplify comprehension, we illustrate the data flow in Fig. 2, where solid lines represent plaintext transmission and dashed lines represent ciphertext transmission. The data parties secretly share data to a random selection of t computation parties based on matrix E. Upon receiving the data, the computation parties locally aggregate the same key-value pairs and then securely compute the frequency and mean estimates of the key-value pairs through Protocol 1 and Protocol 2.

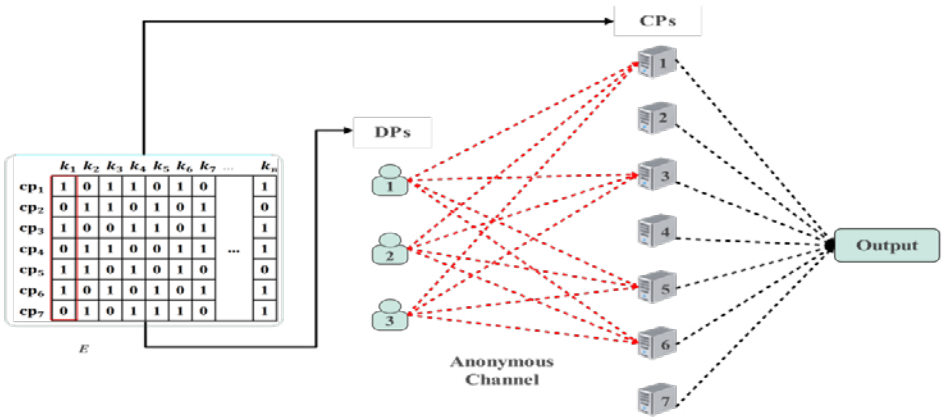

Fig. 2. Data flow

4.2. Initialization phase

During this phase, the parties must deliberate on the data structures, parameters, and other components employed in the protocol. This stage is required only once at system initialization. Initially, all participating parties must reach a consensus on a finite field F_p, which will be used as the foundation for data representation, secret sharing, and all subsequent computations.

To ensure that the secure multiparty computation outputs statistics q_k and μ_k meet differential privacy requirements, the following settings must be made for the data in set S:

- The sensitivity of the statistics can be limited by restricting the value range to an interval of size 2R, where each value $v_{ij} = [x - R, x + R]$, with x being a certain value.
- Simplification of processing can be achieved by having clients pre-aggregate data for different keys, with each key appearing no more than once in S_i.
- Each user may have at most λ key-value pairs for a more precise analysis, i.e., $|S_i| \leq \lambda$, with $\lambda = n$ in the worst case.
- The sensitivity of the mean calculation is limited by ensuring the frequency of each key is at least γ, where $\gamma \geq 1$.

Each participant maintains two arrays of length n, arrays $\boldsymbol{k}$ and $\boldsymbol{v}$, where n represents the total count of all potential key-value pairs. Initially, all elements of these two arrays are set to '0'. The global keys are numbered, denoting the j-th key-value pair as $< k_j, v_j >$. If a data party possesses $< k_j, v_j >$, then the value v_j is stored in the j-th position of array $\boldsymbol{v}$, and the j-th position of array $\boldsymbol{k}$ is set to '1', serving as an indicator that the key-value pair exists; thus, $k_j = 1$ indicates that a data party holds the key-value pair $< k_j, v_j >$. During computations, these arrays are directly manipulated to ensure the association between keys and values remains unchanged, thereby simplifying the data handling process.

In SPDZ, each secretly shared value has an associated MAC to verify the integrity and authenticity of the shared value. The MAC prevents malicious parties from committing fraud when sharing input data. To ensure the correctness of secure multiparty computation under a malicious model, when conducting secure multiparty computation on a certain key k, it is necessary for all data parties to choose the same computation party. To facilitate the selection and management of computation parties during the computation process, we number the l computation parties from 1 to ℓ. Before officially starting, a random matrix E consisting of '0's and '1's is generated and distributed to all participants. In this matrix E, each row corresponds to a computation party, and each column represents the set of computation parties chosen to process a key-value pair, with each column containing t '1's and $(\ell - t)$ '0's, where $2 < t \leq \ell - 1$. By generating this random matrix E, the participation of each computation party in handling different key-value pairs is clearly displayed. This method effectively manages and selects computation parties and achieves load balancing and simplification of the computation process in practical applications.

4.3. Offline phase

This phase only involves computation parties, generating Beaver triples, random values, and other auxiliary data through the SPDZ preprocessing protocol. These auxiliary data are independent of the input data for the online computation phase, allowing them to be computed and stored in advance. By generating these data ahead of time, the online phase can achieve efficient computation while maintaining high security.

4.4. Data collection phase

Each data party $DP_i, i \in [1, ..., m]$, when processing the global j-th key-value pair (k_{ij}, v_{ij}), selects t computation parties $(2 < t \leq \ell - 1)$ guided by the j-th column of matrix E, and secretly shares (k_{ij}, v_{ij}) with the corresponding computation parties using the SPDZ scheme, where ℓ indicates the total number of computation parties. Each column in the random matrix E contains t '1's and $(\ell - t)$ '0's; if a position in the column is '1', it indicates that the

computation party corresponding to that row is selected to process the key-value pair; if it is '0', it indicates that the computation party corresponding to that row is not selected.

When a computation party $CP_i, i \in [1,...,\ell]$ receives data shared by a data party, it only needs to add the secret shares $[k_j]$ and $[v_j]$ to the corresponding positions in its local arrays $\boldsymbol{k}$ and $\boldsymbol{v}$, thereby updating the locally held $([k_j],[v_j])$. As introduced in Section 2.1 about SPDZ, the two secret values $[x]_i = (x_i, m_i^{(x)}, \Delta_i)$ and $[y]_i = (y_i, m_i^{(y)}, \Delta_i)$ held by computation party $CP_i, i \in [1,...,\ell]$ can be computed locally to obtain $[x]+[y] = [x+y] = (x_i + y_i, m_i^{(x)} + m_i^{(y)}, \Delta_i)$. Secret sharing addition is completed locally without any interaction, and during this process, the computation party does not know the actual data values.

4.5. Online computation phase

After completing data collection, each $CP_i, i \in [1,...,\ell]$ holds shares of all key-value pairs $([k_j],[v_j])$ from all data parties, where $[k_j] = (k_{j_1},...k_{j_t}, m_1^{(k_j)},..., m_t^{(k_j)}, \Delta_1,..., \Delta_t)$ and $[v_j] = (v_{j_1},...v_{j_t}, m_1^{(v_j)},..., m_t^{(v_j)}, \Delta_1,..., \Delta_t)$. Therefore, in this phase, the computation parties only need to perform basic operations such as addition, multiplication, and division [17] to complete the corresponding computations.

4.5.1 Frequency estimation

As introduced in Section 4.2 regarding key flag for key k, the data parties use digits '1' and '0', for key flags. After data collection, computation parties hold secret shares of the key flags for key k_j. For the frequency estimation of the global j-th key, as shown in Fig. 3, the computation parties execute Protocol 1. Guided by matrix E, t computation parties involved in the computation are selected, and they compute the secret shared portion of the frequency $[q_k]$ for key k_j. Subsequently, the corresponding computation parties perform a MAC [8] check. If the check passes, noise is added to the result using the method described in [18] to obtain the frequency estimation for key k_j; otherwise, the protocol is halted. Here, $\Delta = \lambda$, where λ represents the maximum number of keys k that a data party possesses, and this value is publicly known.

Protocol 1 Frequency estimation

Input: A set of $[k_j]$, An array id j, A matrix E, A sensitivity Δ.

Output: $[q_k]+[\xi]$.

1. Compute the frequency $[q_k] = \sum_t [k_j]$ of key k_j.

2. Check: Invoking "MAC check" Protocol with $[q_k]$.

3. Generate a random variate [ξ], drawn from $Lap(\Delta / \varepsilon_F)$ by invoking Protocol [23] with $C(t) = \frac{\varepsilon_F}{2\Delta}\int_{-\infty}^{t} \exp(-(\varepsilon_F |s|)/\Delta)ds$.

4. **Return** $[q_k]+[\xi]$.

Fig. 3. Frequency estimation protocol

4.5.2 Mean estimation

Similar to frequency estimation, after data collection, the computation parties possess the secret shares of the key value associated with key k_j. When calculating the mean estimation, each computation party executes Protocol 2 as shown in Fig. 4. Following the guidance of matrix E, t computation parties are selected. They calculate the secret shares of the frequency $[q_k]$ and the key value $[s_k]$ for key k_j. Then, they compute the true global mean $[\mu_k]$ for the j-th key k_j and perform a MAC verification on this mean. If the verification passes, noise is added to the result [18] to obtain the estimated mean for the j-th key k_j; otherwise, the protocol execution is halted. In Protocol 2, $\Delta = \lambda/\gamma 2R$, where λ represents the maximum number of keys k that a data party possesses, γ is the minimum frequency of any key, and $2R$ is the size of the value range. These values are all publicly known.

Protocol 2 Mean estimation

Input: A set of $[k_j],[v_j]$, An array id j, A matrix E, A sensitivity Δ.

Output: $[\mu_k]+[\xi]$.

1. Compute the frequency $[q_k]=\sum_t[k_j]$ of key k_j.
2. Compute the key value $[s_k]=\sum_t[v_j]$ of key k_j.
3. Invoking Protocol [24] with input $[q_k],[s_k]$ to obtain $[\mu_k]=[s_k/q_k]$.
4. Check: Invoking "MAC check" Protocol with $[\mu_k]$
5. Generate a random variate $[\xi]$, drawn from $Lap(\Delta/\varepsilon_M)$ by invoking Protocol [23]

with $C(t)=\frac{\varepsilon_M}{2\Delta}\int_{-\infty}^{t}\exp(-(\varepsilon_M|s|)/\Delta)ds$.

6. **Return** $[\mu_k]+[\xi]$.

Fig. 4. Mean estimation protocol

5. Security analysis

For the global j-th key-value pair $<k_j,v_j>$ among the data parties, if it exists, then $k_j=1$ and the key value is v_j; if it does not exist, then $k_j=0$ and the key value is also 0. Therefore, the key frequency $[q_k]$ and the key value $[s_k]$ calculated through Protocols 1 and 2 are correct. These computations are performed locally by the computing parties, can be completed without interaction, and the addition of local secret-shared parts does not reveal any additional information to the computing parties. According to the properties of Protocol [18], the noise generation in Protocols 1 and 2 satisfies statistical privacy, and Catrina and Saxena [19] provide a more detailed description of this property. Our protocol is an extension of the SPDZ framework, which has been rigorously established as secure under the UC framework [10]. It restricts computation parties to only gain insights from differentially private outputs, ensuring confidentiality. Additionally, any malfeasance affecting computational integrity is promptly identified through MAC validations [8]. In the event of a participant's withdrawal, the protocol ceases instantly, aborting all computations and thus preserving its security integrity.

6. Experimental evaluation

6.1. Setup

To comprehensively assess the feasibility of our protocol, we deployed the Ubuntu 22.04.3 LTS operating system on a server with an x86_64 architecture. This server is equipped with an Intel Xeon Silver 4215R processor (with a base frequency of 3.2GHz and powerful capabilities of 8 cores and 16 threads) and 65.48GB of system memory to ensure the stability and efficiency of the testing environment. Subsequently, we conducted several benchmark tests on the server to analyze the impact of the number of computation parties ℓ and the number of key-value pairs N on the runtime.

For a fair comparison, we re-implemented other schemes using Python based on MP-SPDZ, including the protocol by Wu et al. [23], which is used for generating Laplace random variables, and performed all cryptographic operations using OpenSSL 3.0.2. MP-SPDZ integrates features released in repositories of previous generations of SPDZ protocols and can be used to benchmark various MPC protocols across different security models, including honest and dishonest majorities, as well as semi-honest (passive) and malicious (active) corruption models. The underlying technologies include secret sharing, homomorphic encryption, and garbled circuits. As shown in Table 1, this experiment conducted the same computations across different protocols to compare performance. The MASCOT protocol, implemented using oblivious transfer, and Overdrive, implemented using homomorphic encryption, both resist malicious participants. The Shamir protocol is secure under semi-honest conditions.

Table 1. Protocol security model.

Protocols	Security model
Shamir	Semi-honest
MASCOT	Malicious
Overdrive (Low Gear)	Malicious

6.2. Selective MPC vs. General MPC

To demonstrate that the selective MPC scheme under a malicious model can effectively reduce the overall running time of MPC, we conducted 50 single key-value pair tests for both the selective MPC scheme and the conventional MPC scheme under the Overdrive (Low Gear) protocol, comparing their average running times during the online phase. The experimental results, as shown in Fig. 5a, indicate that under a malicious model, the online times of both schemes are similar when the number of computing parties is small. However, as the number of computing parties increases, the superiority of the selective MPC scheme becomes apparent, effectively reducing the overall computational overhead during the process. Additionally, as illustrated in Fig. 5b, since the selective MPC scheme involves only a subset of computing parties in the computation, the volume of data transmitted during the online phase is also less than that of the traditional MPC scheme.

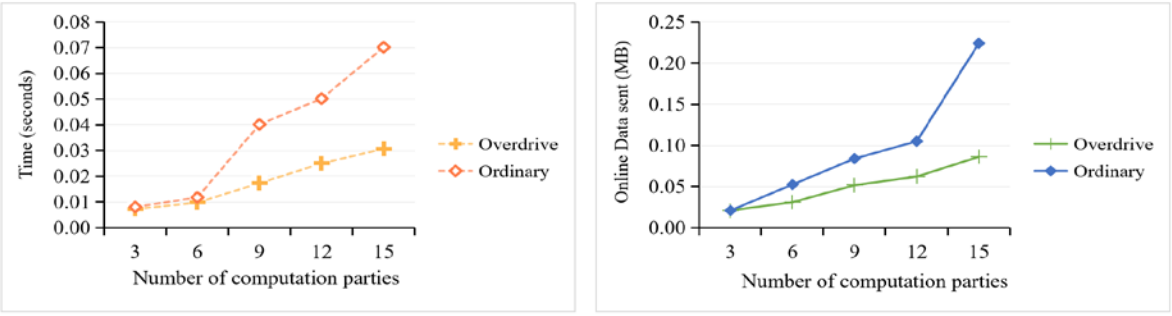

Fig. 5. (a) single key-value pair online phase time ; (b) data transmission during the online phase for a single key-value pair.

6.3. Number of computation nodes

To compare the impact of the number of computing parties on the average runtime under malicious and semi-honest models, we conducted single key-value pair test experiments under the Shamir, MASCOT, and Overdrive (Low Gear) protocols. Each protocol was tested 50 times, comparing their average running times during the online phase. The experimental results, as shown in Fig. 6a, indicate that the online runtime of the Overdrive scheme under the malicious model is similar to that of the Shamir scheme under the semi-honest model, while the MASCOT scheme, implemented through oblivious transfer, has an online runtime more than twice as long as the previous two.

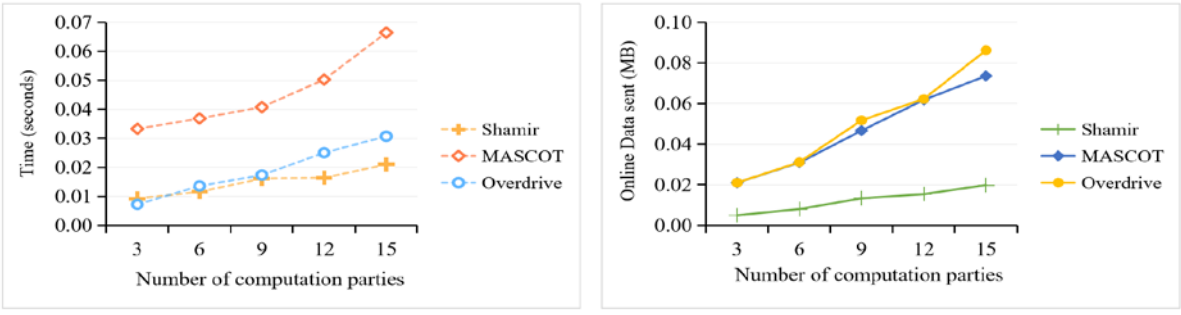

Fig. 6. (a) single key-value pair online phase time ; (b) data transmission during the online phase for a single key-value pair.

As shown in Fig. 6b, since the Overdrive and MASCOT schemes require higher security in the malicious model, including additional verification, encryption, and decryption operations, these operations increase the cost of data transmission, leading to a higher amount of data sent during the online phase compared to the Shamir scheme.

6.4. Varying key domain sizes

To validate the effectiveness of the selective MPC scheme for multiple key-value pair computations under a malicious model, we conducted tests with a default setup of three data parties and five computing nodes, for different numbers of key-value pairs N ($N \in [20, 50, 100, 500, 1000]$), under the Shamir, MASCOT, and Overdrive schemes. The experimental results, as depicted in Fig. 7, show that the online running time of our selective MPC scheme under the malicious model is close to, and even less than, the online running time of the selective MPC scheme under the semi-honest model, especially in scenarios with a large number of key-value pairs.

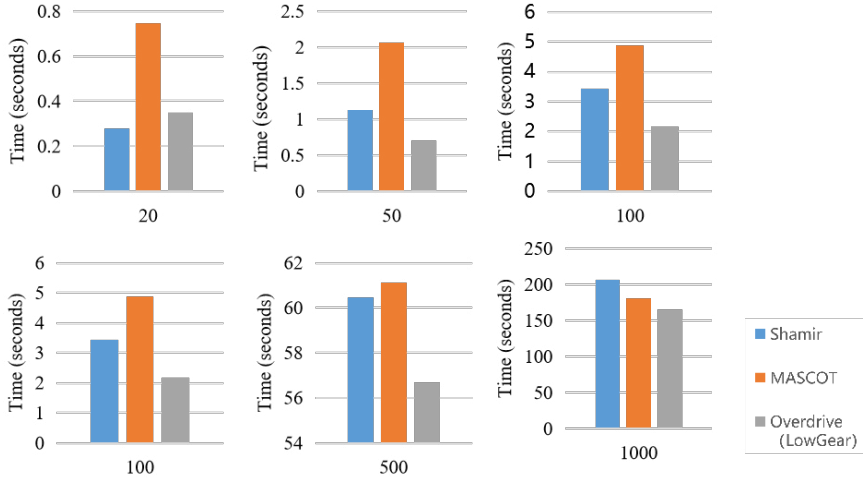

Fig. 7. multi-key-value pair online phase time.

7. Conclusion

We integrate MPC and differential privacy technologies to develop a selective MPC scheme within a malicious framework, effectively minimizing the overall computational overhead during the MPC process. At the same time, it achieves privacy protection effects similar to centralized differential privacy without the need for a trusted third party. This scheme can be used not only to compute frequency estimates and mean estimates of key-value pairs but also envisions future applications in the field of distributed machine learning. Here, it aims to achieve efficient and secure operations even in environments with malicious models.

Acknowledgements

This work was supported by the National Natural Science Foundation of China [Grant NO. 62262012] and the Foundation of Hainan University [Grant NO. KYQD22094].

References

[1] Zhang, Baoquan, Haoyu Gong, and David H. C. Du. (2023) "PMDB: A Range-Based Key-Value Store on Hybrid NVM-Storage Systems." *IEEE Transactions on Computers* **72(5)**: 1274−1285.

[2] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. (2006) "Calibrating Noise to Sensitivity in Private Data Analysis." *In Theory of Cryptography.* 265−284.

[3] Qingqing Ye, Haibo Hu, Xiaofeng Meng, Huadi Zheng. (2019) "PrivKV: Key-Value Data Collection with Local Differential Privacy." *IEEE Symposium on Security and Privacy*: 317-331.

[4] Hui Zhu, Xiaohu Tang, Laurence Tianruo Yang, Chao Fu, Shuangrong Peng. (2023) "Key-value data collection and statistical analysis with local differential privacy." *Information Sciences* 640: 119058.

[5] Lin Sun, Jun Zhao, Xiaojun Ye, Shuo Feng, Teng Wang, Tao Bai. (2019) "Conditional Analysis for Key-Value Data with Local Differential Privacy." *CoRR abs*/1907.05014.

[6] Duchi, John C., Michael I. Jordan, and Martin J. Wainwright. (2013) "Local privacy and statistical minimax rates." *In Allerton 2013*, page 1592.

[7] Sameer Wagh, Xi He, Ashwin Machanavajjhala, Prateek Mittal. (2021) "DP-cryptography: marrying differential privacy and cryptography in emerging applications." *Communications of the ACM* **64(2)**: 84-93.

[8] Thomas Humphries, Rasoul Akhavan Mahdavi, Shannon Veitch, Florian Kerschbaum. (2022) "Selective MPC: Distributed Computation of Differentially Private Key-Value Statistics." *CCS 2022:* 1459-1472.

[9] Marcel Keller, Valerio Pastro, Dragos Rotaru. (2018) "Overdrive: Making SPDZ Great Again." *EUROCRYPT* (3) 2018: 158-189.

[10] Ran Canetti. (2001) "Universally Composable Security: A New Paradigm for Cryptographic Protocols." *In Proceedings of the 2001 IEEE International Conference on Cluster Computing*, *pages* 136−145.

[11] Massoud Hadian Dehkordi, Seyed Taghi Farahi, and Samaneh Mashhadi. (2024) "LWE-based Verifiable Essential Secret Image Sharing Scheme ((t, s, k, n)$({t, s, k, n})$ - VESIS)." *IET Image Processing* **18(4)**: 1053–1072.

[12] Haibo Tian, Yanchuan Wen, Fangguo Zhang, Yunfeng Shao, and Bingshuai Li. (2024) "Lattice Based Distributed Threshold Additive Homomorphic Encryption with Application in Federated Learning." *Computer Standards & Interfaces* **87**: 103765.

[13] Even, S, Goldreich O, and Lempel A. (1985) "A Randomized Protocol for Signing Contracts." *Communications of the ACM* **28(6)**: 637–647.

[14] Keller, Marcel, Emmanuela Orsini, and Peter Scholl. (2016) "MASCOT: Faster Malicious Arithmetic Secure Computation with Oblivious Transfer." *In ACM CCS*, 830−842.

[15] Donald Beaver. (1991) "Efficient Multiparty Protocols Using Circuit Randomization." *In CRYPTO*, 420−432.

[16] Changhui Hu, Jin Li, Zheli Liu, Xiaojie Guo, Yu Wei, Xuan Guang, Grigorios Loukides, Changyu Dong. (2021) "How to Make Private Distributed Cardinality Estimation Practical, and Get Differential Privacy for Free." *USENIX Security Symposium 2021*: 965-982.

[17] Ivan Damgård, Valerio Pastro, Nigel P. Smart, Sarah Zakarias. (2012) "Multiparty Computation from Somewhat Homomorphic Encryption." *CRYPTO*: 643-662

[18] Genqiang Wu, Yeping He, Jingzheng Wu, and Xianyao Xia. (2016) "Inherit differential privacy in distributed setting: Multiparty randomized function computation." *In 2016 IEEE Trustcom/BigDataSE/ISPA, pp.* 921−928.

[19] Octavian Catrina and Amitabh Saxena. (2010) "Secure computation with fixed-point numbers." *In International Conference on Financial Cryptography and Data Security*. 35–50.

[20] Damgård I., Keller M., Larraia E., et al. (2013) "Practical covertly secure MPC for dishonest majority - or: Breaking the SPDZ limits." *In Proceedings of the European Symposium on Research in Computer Security. Berlin: Springer, pages* 1-18.

[21] Zvika Brakerski, Craig Gentry, Vinod Vaikuntanathan. (2014) "(Leveled) Fully Homomorphic Encryption without Bootstrapping." *ACM Transactions on Computation Theory* **6(3)**: 13:1-13:36..

[22] Two meaningful secret image sharing schemes based on integer wavelet transform and LWE[19] Octavian Catrina and Amitabh Saxena. (2010) "Secure computation with fixed-point numbers." *In International Conference on Financial Cryptography and Data Security.* 35–50.

[23] Satheesh K. S. V. A. Kavuri, Gangadhara Rao Kancherla, Basaveswara Rao Bobba. (2014) "Data authentication and integrity verification techniques for trusted/untrusted cloud servers." *ICACCI 2014*: 2590-2596.

[24] Marcel Keller. (2020) "MP-SPDZ: A Versatile Framework for Multi-Party Computation." *CCS 2020*: 1575-1590.