2025 International Conference on Intelligent Computing
July 26-29, Ningbo, China
https://www.ic-icc.cn/2025/index.php

Efficient Delegated Multi-Party Private Set Intersection Protocol for Large-Scale Datasets

Ou Ruan[1,2][0000-0001-8189-3258], Huiwen Miao[1,2][0009-0004-2274-856X], and Changwang Yan[2][0009-0005-9925-440X]

[1] School of Computer Science, Hubei University of Technology, Wuhan, Hubei Province 430000, China
[2] Hubei Provincial Key Laboratory of Green Intelligent Computing Power Network

Abstract. Private set intersection (PSI) as a core research direction in modern cryptography can accurately obtain the intersection information of multiple parties' data while ensuring the confidentiality of the original set of each participant. With the advantages of cloud platforms in storage and computation, cloud-based PSI schemes are getting more and more attention. Based on Paillier homomorphic encryption algorithm and pseudo-random function, this paper proposes an efficient delegated multi-party private set intersection protocol suitable for large-scale data sets. The protocol transforms the dataset intersection problem into a polynomial rooting problem and uses random polynomial blinding methods and homomorphic encryption techniques to ensure the security of the protocol. We give a rigorous formal security proof of the protocol and implement it using the C++ programming language. Our advantages can be demonstrated from the experimental analysis as follows: (a) the protocol is more suitable for large-scale dataset scenarios than the relevant protocols. The time complexity of ours' server is while it's in other protocols where d is the size of the dataset; (b) our protocol is much more efficient. The running time for clients of our protocol is 1/3 of that of the others; (c) the protocol does not depend on the existence of a secure channel while the comparison protocol needs.

Keywords: Private Set Intersection, Cloud Delegation, Set Polynomial Representation, Large-Scale Datasets.

1 Introduction

With the deep integration of big data, artificial intelligence, and cloud computing, data has become a fundamental driver of socio-economic development. However, the tension between data sharing and privacy protection is increasingly pronounced. On one hand, data silos constrain the potential of cross-domain knowledge discovery; on the other, direct exposure of raw data poses risks of sensitive information leakage and legal non-compliance. Thus, achieving secure and efficient multi-party collaborative computing while ensuring data privacy has become a critical challenge for both academia and industry [1].

Private Set Intersection (PSI) provides a key technological path to resolve the above contradiction. PSI allows two or more participants to output the intersection content without revealing the non-intersecting elements of their respective sets. For example, medical institutions can identify common mutation sites through PSI protocols without sharing all the data [2]; banks can compare lists of defaulters with each other through PSI technology without the need for the intervention of trusted third-party organizations [3]; airports compare passenger biometrics with multinational security databases through PSI technology without leaking users' biometric privacy [4]. Since Meadows [5] first proposed the concept of PSI, its technical system gradually moved from theoretical conception to engineering landing (Freedman et al. [6]; Pinkas et al. [7]). However, with the rapid development of the big data era, the existing PSI protocols still face multiple challenges of efficiency, scalability and security.

In order to solve this problem, cloud-commissioned PSI protocols have been proposed one after another. Cloud has two major advantages: the cloud storage advantage and the cloud computing advantage [8]. Compared with traditional local PSI computation, cloud-commissioned PSI has two significant differences [9]: security model and computation model. The computational side of traditional local PSI trusts its own local resources, while cloud servers are untrustworthy to users; traditional local PSI performs computation and storage on local devices, while cloud servers of cloud-commissioned PSI share the computation overhead.

Moving to multi-party PSI introduces additional complexity and opportunities based on the Cloud Delegated PSI protocol. Multi-party PSI is about collaboratively computing the intersection of multiple participants without disclosing non-intersection elements while ensuring privacy and security [10]. Multi-party systems require a higher degree of cloud resource delegation and fine-grained orchestration. With the flexibility of cloud computing, multi-party PSI protocols are able to handle large-scale data and increase the efficiency of collaboration, but securing data in an untrusted cloud environment remains a key concern. In 2022, Abadi introduced Feather [27], a lightweight cloud-based multi-party PSI protocol using Bloom filters for verification, improving efficiency. However, it still relies on secure channels [28], and the server has relatively low computing efficiency in scenarios involving large-scale data.

Based on the analysis above, we propose an efficient delegated multi-party private set intersection protocol for large-scale datasets (DMPSI-LD). Our protocol offers several improvements over O-PSI [22] and Feather [27]. It maintains scalability, ensuring that the number of clients does not impact individual clients or the querying party, thereby sustaining performance as the system expands. Additionally, the server’s computational complexity increases linearly with the number of participants, ensuring predictable performance. Unlike O-PSI and Feather, our protocol eliminates the need for a secure communication channel, reducing both complexity and potential vulnerabilities. Furthermore, it optimizes the server’s ability to handle larger datasets, improving efficiency without sacrificing performance. In summary, DMPSI-LD retains their advantages while addressing key limitations, such as the requirement for a secure channel and server performance, particularly for large datasets.

2 Related work

In 1976, Diffie and Hellman [11] proposed the Diffie-Hellman key exchange protocol to provide a theoretical basis for subsequent DH-based PSI schemes. In 2004, Freedman [8] proposed the DH-based hash comparison PSI protocol, which utilizes a shared key to generate a matchable hash value. In 2012, Huang et al. [12] compared the efficiency of the Diffie-Hellman hash comparison with the obfuscated circuit scheme and proved the advantage of the Diffie-Hellman scheme under large-scale data. In 2014, Pinkas [13] combined DH hashing with OT scaling techniques to propose an efficient PSI protocol to support billion-scale data. In 2016, Abadi et al. [14] proposed verifiable DH hash comparison schemes to support third-party delegated computation scenarios. In 2017, Rindal et al. [15] optimized the DH hash comparison protocol for malicious adversary models to enhance security through dual execution, and Kiss et al. [16] extended the DH scheme to support PSI with unequal-length sets to optimize the performance of mobile applications. In 2021, Rosulek et al. [17] optimize DH hash comparison for small-scale datasets and propose compact and malware-resistant protocols.

Traditional privacy set intersection (PSI) protocols rely on client-side computation, constrained by storage and processing power, making large-scale data handling inefficient. Cloud computing introduces a new paradigm, enabling outsourced data storage and computation. In 2012, Kerschbaum proposed the first cloud-based PSI protocol [18] for encrypted data outsourcing and cloud computation but lacked reusability and had security flaws. He later optimized it with Bloom filters and homomorphic encryption [19], improving efficiency but still requiring local data storage. In 2014, Liu et al. introduced a lightweight PSI protocol [20] with double encryption, though vulnerable to aggregate base leakage. Kamara et al. [21] enhanced security using blinding and zero-knowledge proofs but restricted cloud computation. In 2017, Abadi et al. proposed two two-party PSI protocol with homomorphic encryption [22], enabling fully outsourced encrypted storage with linear complexity. The first protocol, O-PSI, computes the intersection efficiently using the Paillier homomorphic encryption algorithm that displaces the blinding factor of the set polynomial. The second protocol, EO-PSI, removes the time-consuming cryptographic algorithms and greatly improves the efficiency of the protocol by running an efficient pseudo-random function implementation. The second protocol, EO-PSI, removes the time-consuming cryptographic algorithms and greatly improves the efficiency of the protocol through an operationally efficient pseudo-random function implementation. However, both protocols rely on secure channels in order to ensure security [23,24]. Later optimizations [24] used hash indexing and pseudo-random functions to reduce overhead but retained secure channel reliance. In 2018, Mehd et al. [24] introduced an asymmetric encryption scheme improving security. Yang et al. [25] replaced homomorphic encryption with RSA, increasing computational efficiency by 40%, while Tajima et al. [26] used fully homomorphic encryption (FHE) for provable security but at a high computational cost due to ciphertext expansion.

Multi-party PSI is also widely used in various domains by ensuring the data privacy of multiple participants while performing efficient intersection computation. Despite

its progress in security, protocol design, and efficiency, it still faces challenges such as scalability, generalization, and malicious protection.

In 2022, Abadi extended the second protocol EO-PSI in [22] to multiparty (Feather [27]) and used a Bloom filter to enhance the performance of the protocol. Feather inherits the feature of high efficiency of EO-PSI, the operation efficiency of users and queriers is not affected by the number of users, and the operation efficiency of servers is linearly correlated with the number of users. However, it still has two aspects that need to be improved: (a) the protocol still does not address the need for secure channels; and (b) the efficiency of the protocol is less efficient with large-scale datasets. To address this issue, our protocol retains the advantages of the Feather protocol and further improves the processing capability of the server when the amount of data increases, while no longer relying on the secure channel and requiring only a small amount of encryption to complete the computation under the open channel, which improves the efficiency and applicability of the protocol while guaranteeing privacy and security.

3 Specific Protocol

In this subsection, the process and details of the protocol will be elaborated specifically.

3.1 System Model

System Framework

The cloud-commissioned multi-party private aggregate intersection protocol contains three types of participants: t users $A_i(1 \le i \le t)$, one query party B, and one cloud server C. The system framework is shown in Fig. .

Fig. 1. System framework diagram

Define the following function: $f(\perp, S_i, \ldots, S_t, S_b)$,where $\perp$ is the empty string, $S_1, \ldots, S_t$represents the data set of user $A_1, \ldots, A_t$,$S_b$represents the data set of querying party B, and Enc represents the encryption algorithm. $A_1, \ldots, A_t$and B encrypt or blind their respective data sets and upload them to C, which computes the encrypted or blinded data to find the intersection $Enc(S_1 \cap \cdots \cap S_t \cap S_b)$ of the encrypted or blinded set, and sends it to the querying party B. The querying party B can decrypt or unblind this data set to find the intersection.

System Goals and Requirements

Based on the definition of the cloud delegation PSI protocol given in the article [28], the objectives of the system given in this paper are as follows:

(1) Each delegating user $A_i(1 \le i \le t)$ is able to securely store their set $S_i(1 \le i \le t)$ on the cloud server C;

(2) Query user B is able to securely store his set S_b on cloud server C;

(3) Cloud server C is able to unite user $A_i(1 \le i \le t)$ and querying party B to compute intersection $S_1 \cap \cdots \cap S_t \cap S_b$ efficiently.

The requirements of the system are as follows:

(1) Delegated user $A_i(1 \le i \le t)$ knows only their private set $S_i(1 \le i \le t)$;

(2) Cloud server C cannot know any private information;

(3) Querying party B can only know his set S_b and the intersection information $S_1 \cap \cdots \cap S_t \cap S_b$ of all users;

(4) Cloud server C handle most of the computation and communication tasks;

(5) User $A_i(1 \le i \le t)$ and querying party B should not interact directly or should keep such interaction minimal.

3.2 Overview of the Protocol

The protocol is divided into 3 phases: the initialization phase, the data uploading phase, and the intersection calculation phase. The initialization phase works as the cloud server generates the necessary parameters for the entire protocol, and the users prepare their data sets. The data uploading phase works as all the users store their data on the cloud server, which utilizes the storage capacity of the cloud server so that the data can be deleted locally to save storage space. The intersection computation phase is performed by the querying party's request, and the other users cooperate, and the cloud server performs the main computation work.

The protocol uses point-valued representation of polynomials for the computation. The point-valued representation of polynomials reduces the time complexity of multiplication of polynomials to linear. The protocol blinds the roots by adding a random number to the polynomial after blinding the random polynomial. A polynomial plus a random number means that the curve of this polynomial is shifted up and down on the axes, and if this random value is not exposed, then the roots of the original polynomial are not exposed. With the protocol in this paper, the need for a secure communication channel can be eliminated, optimizing the server's ability to handle larger datasets and increasing efficiency without sacrificing performance.

3.3 The Details of the Protocol

The detailed flowchart of the protocol is shown in Fig. .

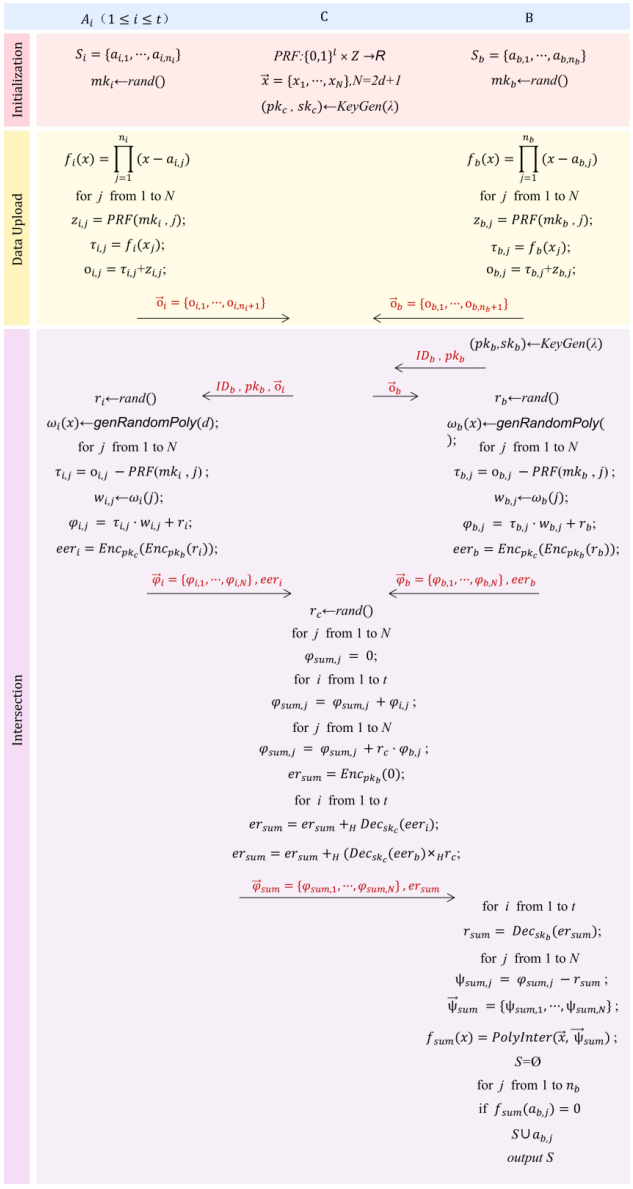

Fig. 2. Protocol Flow Chart

Setup Phase

Step 1: A finite field Fp is constructed with large prime p, followed by the generation of a pseudo-random function $PRF: \{0,1\}^l \times Z \rightarrow R$ where l denotes secret key

bit-length. The system configures the user-defined set cardinality bound d, selects security parameter λ, and generates point vector $\vec{x} = \{x_1, \ldots, x_n\}, N = 2d + 1$. All parameters are ultimately published by cloud server C.

Step 2: User A_i generates a master key $\mathrm{mk_i}$ using a cryptographically secure random number generator (RNG), which serves as the seed for the PRF $rand()$.The user maintains a private dataset $\overrightarrow{S_i} = \{a_{i,1}, \ldots, a_{i,n}\}, n_i \leq d$.

Step 3: Query party B generates a master key $\mathrm{mk_b}$ using a cryptographically secure random number generator (RNG), which serves as the seed for the PRF $rand()$.The B maintains a private dataset $\overrightarrow{S_b} = \{a_{b.1}, \ldots, a_{b,n_b}\}, = n_b \leq d$.

Initialization Phase

Step 1: Executes the Paillier cryptosystem's key generation algorithm to produce a public-private key pair $(pk_c, sk_c) \leftarrow KeyGen(\lambda)$ The public-private keys are represented by $\mathrm{pk_c}$and $\mathrm{sk_c}$, respectively, and λ stands for a security parameter in this procedure.

Step 2: User A_i represents the dataset as a polynomial $f_i(x) = \prod_{j=0}^{n_i}(x - a_{i,j})$, and then substitutes the point vector $\vec{x}$ into it to obtain the point value vector. Utilizes the PRF and master key mk_i to generate N random numbers, forming a random vector $\vec{Z}_i$.Computes a blinded point-value vector $\vec{o}_i$ by summing the original and random vectors, then uploads $\vec{o}_i$ to C.

$$\vec{\tau}_i = \{\tau_{i,1}, \ldots, \tau_{i,N}\} = \{f_i(a_{i,1}), \ldots, f_i(a_{i,N})\} \quad (1)$$

$$\vec{z}_i = \{z_{i,1}, \ldots, z_{i,N}\} = \{PRF(mk_i, 1), \ldots, PRF(mk_i, N)\} \quad (2)$$

$$\begin{aligned} \vec{o}_i &= \vec{\tau}_i + \vec{z}_i = \{\tau_{i,1} + z_{i,1}, \ldots, \tau_{i,N} + z_{i,N}\} \\ &= \{f_i(a_{i,1}) + PRF(mk_i, 1), \ldots, f_i(a_{i,N}) + PRF(mk_i, N)\} \end{aligned} \quad (3)$$

Step 3: B represents the dataset as a polynomial $f_b(x) = \prod_{j=0}^{n_b}(x - a_{b,j})$, and then substitutes the point vector $\vec{x}$ into it to obtain the point value vector.Utilizes the PRF and master key mk_b to generate N random numbers, forming a random vector $\vec{Z}_b$.Computes a blinded point-value vector $\vec{o}_b$ by summing the original and random vectors, then uploads $\vec{o}_b$ to C.

$$\vec{\tau}_b = \{\tau_{b,1}, \ldots, \tau_{b,N}\} = \{f_b(a_{b,1}), \ldots, f_i(a_{b,N})\} \quad (4)$$

$$\vec{z}_b = \{z_{b,1}, \ldots, z_{b,N}\} = \{PRF(mk_b, 1), \ldots, PRF(mk_b, N)\} \quad (5)$$

$$\begin{aligned} \vec{o}_b &= \vec{\tau}_b + \vec{z}_b = \{\tau_{b,1} + z_{b,1}, \ldots, \tau_{b,N} + z_{b,N}\} \\ &= \{f_b(a_{b,1}) + PRF(mk_b, 1), \ldots, f_b(a_{b,N}) + PRF(mk_b, N)\} \end{aligned} \quad (6)$$

Intersection Computation Phase

Step 1: Generates a Paillier public-private key pair $(\mathrm{pk_b}, \mathrm{sk_b}) \leftarrow \mathrm{KeyGen}(\lambda)$ and transmits its identity $\mathrm{ID_b}$ and public key $\mathrm{pk_b}$ to C to request intersection computation. Returns the dataset $\vec{o}_i$ stored by user A_i to A_i and the dataset $\vec{o}_b$ stored by B

to party B.Transmits auxiliary parameters pk_b and ID_b to A_i to explicitly identify the requesting party initiating the intersection query.

Step 2: Restores $\vec{\tau}_i$. Generate a randomized d-order polynomial $\omega_i(x)$. Construct a point-value vector $\vec{w}_i$ for the randomized polynomial. A random number r_i is generated, followed by the computation of parameters $\vec{\varphi}_i$ and eer_i, $\vec{\varphi}_i$ and eer_i are transmitted to the cloud server C for further processing.

$$\vec{\tau}_i = \vec{o}_i - \vec{z}_i = \{o_{i,0} - z_{i,0}, \ldots, o_{i,N} - z_{i,N}\} \quad (7)$$

$$\omega_i(x) \leftarrow genRandomPoly(d) \quad (8)$$

$$\vec{w}_i = \{w_{i,1}, \ldots, w_{i,N}\}, w_{i,j} = \omega_i(j), 1 \le j \le N \quad (9)$$

$$\varphi_{i,j} = \tau_{i,j} \times w_{i,j} + r_i, 1 \le j \le N \quad (10)$$

$$\vec{\varphi}_i = \{\varphi_{i,1}, \ldots, \varphi_{i,N}\} \quad (11)$$

$$eer_i = Enc_{pk_c}(Enc_{pk_b}(r_i)) \quad (12)$$

Step 3: Restores $\vec{\tau}_b$.Reduce the polynomial $f_b(x)$ via $\vec{\tau}_b$ and Lagrange interpolation, factorize $f_b(x)$ using the Jenkins-Traub algorithm and restore $\vec{S}_b$. Generate a randomized d-order polynomial $\omega_b(x)$. Construct a point-value vector $\vec{w}_i$ for the randomized polynomial. A random number r_b is generated, followed by the computation of parameters $\vec{\varphi}_b$ and eer_b, $\vec{\varphi}_b$ and eer_b are transmitted to the cloud server C for further processing.

$$\vec{\tau}_b = \vec{o}_b - \vec{z}_b = \{o_{b,0} - z_{b,0}, \ldots, o_{b,N} - z_{b,N}\} \quad (13)$$

$$\omega_b(x) \leftarrow genRandomPoly(d) \quad (14)$$

$$\vec{w}_b = \{w_{b,1}, \ldots, w_{b,N}\}, w_{b,j} = \omega_b(j), 1 \le j \le N \quad (15)$$

$$\varphi_{b,j} = \tau_{b,j} \times w_{b,j} + r_b, 1 \le j \le N \quad (16)$$

$$\vec{\varphi}_b = \{\varphi_{b,1}, \ldots, \varphi_{b,N}\} \quad (17)$$

$$eer_b = Enc_{pk_c}(Enc_{pk_b}(r_b)) \quad (18)$$

Step 4: A random number $\mathrm{r_c}$ is generated. Subsequently, the point-value vectors of all users are aggregated by summing their respective polynomials, yielding the resultant vector $\vec{\varphi}_{\mathrm{sum}}$. Decrypt the parameters $eer_i(1 \le i \le t)$ and eer_b to get $er_i(1 \le i \le t)$ and $\mathrm{er_b}$ and then compute $\mathrm{er_{sum}}$.

$$\varphi_{sum,j} = \varphi_{1,j} + \cdots + \varphi_{t,j} + r_c \cdot \varphi_{b,j}, 1 \le j \le N \quad (19)$$

$$\vec{\varphi}_{sum} = \{\varphi_{sum,1}, \ldots, \varphi_{sum,N}\} \quad (20)$$

$$er_i = Dec_{sk_c}(eer_i), 1 \le i \le t \quad (21)$$

$$er_b = Dec_{sk_c}(eer_b) \quad (22)$$

$$er_{sum} = r_c \times_H er_b +_H er_0 +_H \cdots +_H er_t \quad (23)$$

Step 5: Recover r_{sum} and computes $\vec{\psi}_{sum}$.Construct a polynomial using Lagrange interpolation method for the set $f_{sum}(x)$, and substitute each data point of the set $\vec{\mathrm{S}}_{\mathrm{b}}$

into the polynomial. If the value of the polynomial is 0, then it is considered as an element in the intersection.

$$r_{sum} = Dec_{sk_b}(er_{sum}) \tag{24}$$

$$\psi_{sum,j} = \varphi_{sum,j} - r_{sum}, 1 \le j \le N \tag{25}$$

$$\vec{\psi}_{sum} = \{\psi_{sum,1}, \ldots, \psi_{sum,N}\} \tag{26}$$

$$f_{sum}(x) = PolyInter(\vec{x}, \vec{\psi}_{sum}) \tag{27}$$

4 Scheme Analysis

4.1 Correctness Analysis

From Section 3, it can be known that the protocol can obtain the intersection of all users including and the querying party B at the end. This section will derive the result of the protocol execution to prove the correctness of our protocol at the end. For the $i \in \{1, \ldots, t\}, j \in \{1, \ldots, N\}$:

$$\begin{aligned} eer_i &= Enc_{pk_c}(Enc_{pk_b}(r_i)) \\ eer_b &= Enc_{pk_c}(Enc_{pk_b}(r_b)) \end{aligned} \tag{28}$$

$$\begin{aligned} Dec_{sk_c}(eer_i) &= Dec_{sk_c}(Enc_{pk_c}(Enc_{pk_b}(r_i))) = Enc_{pk_b}(r_i) \\ Dec_{sk_c}(eer_b) &= Dec_{sk_c}(Enc_{pk_c}(Enc_{pk_b}(r_b))) = Enc_{pk_b}(r_b) \end{aligned} \tag{29}$$

$$\begin{aligned} r_{sum} &= Dec_{sk_b}(er_{sum}) \otimes \\ &= Dec_{sk_b}(r_c \times_H er_b +_H er_0 +_H \cdots +_H er_t) \\ &= Dec_{sk_b}(r_c \times_H Dec_{sk_c}(eer_b) +_H Dec_{sk_c}(eer_0) +_H \cdots +_H Dec_{sk_c}(eer_t)) \\ &= Dec_{sk_b}(Enc_{pk_b}(r_c \cdot r_b + r_0 + \cdots + r_t)) \\ &= \sum_{i=1}^{t}(r_i) + r_c \cdot r_b \end{aligned} \tag{30}$$

$$\begin{aligned} \varphi_{sum,j} &= \varphi_{1,j} + \cdots + \varphi_{t,j} + r_c \cdot \varphi_{b,j} \\ &= \tau_{1,j} \cdot w_{1,j} + r_i + \cdots + \tau_{t,j} \cdot w_{t,j} + r_t + r_c(\tau_{b,j} \cdot w_{b,j} + r_b) \\ &= \sum_{i=1}^{t}(\tau_{i,j} \cdot w_{i,j}) + r_c \cdot \tau_{b,j} \cdot w_{b,j} + \sum_{i=1}^{t}(r_i) + r_c \cdot r_b \end{aligned} \tag{31}$$

$$\begin{aligned} \psi_{sum,j} &= \varphi_{sum,j} - r_{sum} \\ &= \sum_{i=1}^{t}(\tau_{i,j} \cdot w_{i,j}) + r_c \cdot \tau_{b,j} \cdot w_{b,j} + \sum_{i=1}^{t}(r_i) + r_c \cdot r_b - \sum_{i=1}^{t}(r_i) + r_c \cdot r_b \\ &= \sum_{i=1}^{t}(\tau_{i,j} \cdot w_{i,j}) + r_c \cdot \tau_{b,j} \cdot w_{b,j} \end{aligned} \tag{32}$$

For each term of the polynomial point-value form of each user A_i, it is multiplied by the corresponding term of the point-value form of the random polynomial (denoted as $\omega_i(x)$) generated by itself. The final result is:

$$\begin{aligned} f_{sum}(x) &= PolyInter(\vec{x}, \vec{\psi}_{sum}) \\ &= f_1(x)\cdot\omega_1(x)+\cdots+f_t(x)\cdot\omega_t(x)+r_c\cdot f_b(x)\cdot\omega_b(x) \end{aligned} \tag{33}$$

When query party B substitutes each data element into the protocol, if and only if the element belongs to the intersection set, will all polynomial terms be set to zero, thereby resulting in the polynomial evaluating to zero.

4.2 Security Analysis

Theorem: If the Paillier homomorphic encryption scheme and the pseudo-random function PRF are secure, then the protocol DMPSI-LD is secure in the semi-honest adversary model.

Proof:

We prove the security of our protocol by considering three scenarios. For each scenario, we compare whether the adversary's simulated view and the real view can be distinguished, thereby determining whether the protocol is secure.

Case 1: Cloud Server C Controlled by Adversary

In a real protocol, the view of the server can be represented as follows:

$$\begin{aligned} &View_C^{\Pi}(\perp, \vec{S}_1, \ldots, \vec{S}_t, \vec{S}_b) \\ &= \{\vec{o}_1, \ldots, \vec{o}_t, \vec{o}_b, \vec{\varphi}_1, \ldots, \vec{\varphi}_t, \vec{\varphi}_b, eer_1, \ldots, err_t, err_b, er_1, \ldots, er_t, er_b, er_{sum}, \vec{\varphi}_{sum}\} \end{aligned} \tag{34}$$

To simulate real-world views, we construct a simulator $\mathrm{Sim_c}$ to emulate scenarios where cloud server C is compromised by adversaries, where index i ranges from 1 to t and j ranges from 1 to N. The simulation process executes the following steps:

Step 1: Create an empty view $Sim_c(\perp, \perp)$.

Step 2: Simulate the initialization process of $A_1, \ldots, A_t, B$'s steps and generate $\mathrm{t}+1$ random numbers $\mathrm{n}'_1, \ldots, \mathrm{n}'_\mathrm{t}, \mathrm{n}'_\mathrm{b}$ to represent the cardinality of $A_1, \ldots, A_t, B$ set. Create $\mathrm{t}+1$ random $\mathrm{n}'_1, \ldots, \mathrm{n}'_\mathrm{t}, \mathrm{n}'_\mathrm{b}$-dimensional row vectors $\vec{S}'_1, \ldots, \vec{S}'_t, \vec{S}'_b$ to represent the data set of $\mathrm{A}_1, \ldots, \mathrm{A}_\mathrm{t}, \mathrm{B}$, and express it as polynomial $f'_1(x), \ldots, f'_t(x), f'_b(x)$. Then, substitute each value $\mathrm{x_j}$ in the vector $\vec{\mathrm{x}}$ and convert these data sets into vector $\vec{\mathrm{T}}'_1, \ldots, \vec{\mathrm{T}}'_t, \vec{\mathrm{T}}'_b$ in point-value form. Then, generate $\mathrm{t}+1$ random N-dimensional row vectors $\vec{Z}'_1, \ldots, \vec{Z}'_t, \vec{Z}'_b$ to blindify the point-value vector, obtaining $\vec{o}'_1, \ldots, \vec{o}'_t, \vec{o}'_b$. The simulator will add this vector to the simulation view. Here, $0 < n'_i \le d$, $\vec{S}'_1 = \{a'_{i,1}, \ldots, a'_{i,n'_i}\}$, $f'_i(x) = \prod_{j=1}^{n'_i}(x - a'_{i,j})$, $\vec{\mathrm{T}}'_i = \{\vec{\mathrm{T}}'_{i,1}, \ldots, \vec{\mathrm{T}}'_{i,N}\}$, $\vec{\mathrm{T}}'_{i,j} = f'_i(x_j)$, $\vec{z}'_i = \{z'_{i,1}, \ldots, z'_{i,N}\}$, $\vec{o}'_i = \{o'_{i,1}, \ldots, o'_{i,N}\}$, $o'_{i,j} = \mathrm{T}'_{i,j} + z'_{i,j}$.

Step 3: Simulate the calculation process of the intersection of sets $A_1, \ldots, A_t, B$'s steps, generate $t+1$ random numbers $r'_1, \ldots, r'_t, r'_b$, then generate $t+1$ N-dimensional random row vectors $\vec{w}'_1, \ldots, \vec{w}'_t, \vec{w}'_b$, and then calculate $\vec{\varphi}'_1, \ldots, \vec{\varphi}'_t, \vec{\varphi}'_b$ based on $\vec{\mathrm{T}}'_1, \ldots, \vec{\mathrm{T}}'_t, \vec{\mathrm{T}}'_b$, $r'_1, \ldots, r'_t, r'_b$, and $\vec{w}'_1, \ldots, \vec{w}'_t, \vec{w}'_b$ .Simultaneously, calculate $eer'_1, \ldots, eer'_t, eer'_b$ based on $r'_1, \ldots, r'_t, r'_b$., and

adds $\vec{\varphi}'_1,\ldots,\vec{\varphi}'_t,\vec{\varphi}'_b$ and $eer'_1,\ldots,eer'_t,eer'_b$ to the simulation view. Here, $\vec{w}'_i = \{w'_{i,1},\ldots,w'_{i,N}\}$, $\vec{\varphi}'_i = \{\varphi'_{i,1},\ldots,\varphi'_{i,N}\}$, $\varphi'_{i,j} = \mathrm{T}'_{i,j}\cdot w'_{i,j} + r'_i$, $eer'_i = Enc_{pk_c}(Enc_{pk_b}(r'_i))$.

Step 4: Simulate the calculation process of the intersection of sets on the cloud server's steps, generate random numbers $\mathrm{r_c}$, calculate $\vec{\varphi}'_{sum}$, $er'_0,\ldots,er'_t,er'_b$, er'_{sum}, and add them to the simulation view. Here, $\vec{\varphi}'_{\mathrm{sum}} = \{\varphi'_{\mathrm{sum},1},\ldots,\varphi'_{\mathrm{sum},N}\}$, $\varphi'_{sum,j} = \varphi'_{i,j}+\ldots+\varphi'_{t,j} + r_c\varphi'_{b,j}$, $er'_i = Dec_{sk_c}(eer'_i)$, $er'_{sum} = r'_c \times_H er'_b +_H er'_1 +_H \cdots +_H er'_t$.

After completing the above steps, all the protocol steps of cloud server C have been executed. The view of the simulator can be obtained as follows:

$$Sim_C(\perp,\perp) = \{\vec{o}'_1,\ldots,\vec{o}'_t,\vec{o}'_b,\vec{\varphi}'_1,\ldots,\vec{\varphi}'_t,\vec{\varphi}'_b,eer'_1,\ldots,err'_t,err'_b,er'_1,\ldots,er'_t,er'_b,er'_{sum},\vec{\varphi}'_{sum}\} \quad (35)$$

By comparing the real view and the simulated view, both vectors $\vec{\mathrm{o}}$ and $\vec{\varphi}$ in the two views have been blinded using random numbers. The values eer and er have been encrypted by random numbers and encrypted using public keys. Moreover, the cloud server cannot completely decrypt them. Therefore, $View_C^{\Pi}$and $Sim_c$are computationally indistinguishable, $View_C^{\Pi}(\perp,\vec{S}_1,\ldots,\vec{S}_t,\vec{S}_b) \simeq Sim_c(\perp,\perp)$.

Case 2: User $A_i(1 \le i \le t)$ Controlled by Adversary

In a real protocol, the view of user $A_i(1 \le i \le t)$ can be represented as follows:

$$View_{A_i}^{\Pi}(\perp,\vec{S}_1,\ldots,\vec{S}_t,\vec{S}_b) = \{\vec{S}_i, f_i(x),\vec{\tau}_i,\vec{z}_i,\vec{o}_i,r_i,\vec{w}_i,\vec{\varphi}_i,er_i,eer_i\} \quad (36)$$

The steps to simulate a real view are similar to case 1, including creating an empty view, simulating data upload, and simulating set intersection computation. The difference is that when creating the empty view, $\vec{S}_i$ is added to the view, $\vec{S}_i = \{a_{i,1},\ldots,a_{i,n_i}\}$.After completing the above steps, all the protocol steps of user A_i have been executed. The view of the simulator can be obtained as follows:

$$Sim_{A_i}(\perp,\vec{S}_i) = \{\vec{S}_i, f_i(x),\vec{\tau}_i,\vec{z}'_i,\vec{o}'_i,r'_i,\vec{w}'_i,\vec{\varphi}'_i,er'_i,eer'_i\} \quad (37)$$

By comparing the real view and the simulated view,, $View_{A_i}^{\Pi}$ and Sim_{A_i} are indistinguishable in calculation, and $View_{A_i}^{\Pi}(\perp,\vec{S}_1,\ldots,\vec{S}_t,\vec{S}_b) \simeq Sim_{A_i}(\perp,\vec{S}_i)$.

Case 3: Query Party B Controlled by Adversary

In a real protocol, the view of B can be represented as follows:

$$View_B^{\Pi}(\perp,\vec{S}_1,\ldots,\vec{S}_t,\vec{S}_b) = \{\vec{S}_b, f_b(x),\vec{\tau}_b,\vec{z}_b,\vec{o}_b,r_b,\vec{w}_b,\vec{\varphi}_b,er_b,eer_b,\vec{\varphi}_{sum},er_{sum},r_{sum},\vec{\psi}_{sum},f_{sum}(x),\vec{S}\} \quad (38)$$

The steps to simulate a real view are similar to case 2, including creating an empty view, simulating data upload, and simulating set intersection computation. After completing the above steps, all the protocol steps of B have been executed. The view of the simulator can be obtained as follows:

$$Sim_B(\perp,\vec{S}_b) = \{\vec{S}_b, f_b(x),\vec{\tau}_b,\vec{z}'_b,\vec{o}'_b,r'_b,\vec{w}'_b,\vec{\varphi}'_b,er'_b,eer'_b,\vec{\varphi}'_{sum},er'_{sum},r'_{sum},\vec{\psi}'_{sum},f'_{sum}(x),\vec{S}'\} \quad (39)$$

By comparing the real view and the simulated view,$View_B^{\pi}$ and Sim_B are indistinguishable in calculation, and $View_B^{\pi}(\perp, \vec{S}_1, \ldots, \vec{S}_t, \vec{S}_b) \simeq Sim_B(\perp, \vec{S}_b)$.

From the above three scenarios we know that our protocol is secure in the semi-honest adversary model.

5 Realization and Performance Analysis

In this section, the implementation of DMPSI-LD is described. Moreover, in the same environment, a comparison of running efficiency and communication complexity with O-PSI [22] and Feather [27] is conducted.

5.1 Protocol implementation

The experiment implemented our protocol using C++. The experimental environment was a 12th Gen Intel(R) Core (TM) i9-12900H with a clock speed of 2.50 GHz, 16GB RAM, Ubuntu 18.04, and 64-bit. The libraries used in the experiment included the large number operation library NTL and the encryption library Crypto++. The experiment used the NTL library to implement the operation of large data and the Paillier encryption algorithm. At the same time, we used the AES-CTR 128bit algorithm of the Crypto++ library as the PRF call.

5.2 Performance analysis

This section presents a detailed analysis of the performance of our protocol in comparison with O-PSI and Feather, as shown in Tables 1, 2 and 3. d is the dataset size and t is the number of clients.

Table 1. Comparison of the protocol

Protocol		Secure communication channel	Time complexity	Communication complexity
O-PSI [22]	User A	√	$O(N)$	$O(N)$
	Query Party B		$O(N^2)$	$O(tN)$
	Server C		$O(tN^2)$	$O(tN)$
Feather [27]	User A	√	$O(N^2)$	$O(N)$
	Query Party B		$O(N^2)$	$O(tN)$
	Server C		$O(tN^2)$	$O(tN)$
DMPSI-LD	User A	×	$O(N^2)$	$O(N)$
	Query Party B		$O(N^2)$	$O(N)$

Server C		$O(tN)$	$O(tN)$

Table 2. Runtime vs. set size under fixed participant count (Unit: seconds)

Protocol / Set size		2^{11}	2^{12}	2^{13}	2^{14}	2^{15}
O-PSI [22]	User A	0.446	0.888	1.808	3.602	7.259
	Query Party B	52.901	112.181	256.396	631.003	1721.517
	Server C	27.983	57.542	127.273	290.823	746.561
Feather [27]	User A	0.394	1.580	6.213	24.936	99.496
	Query Party B	4.819	21.036	81.126	308.392	1340.440
	Server C	1.570	6.225	24.704	99.869	411.918
DMPSI-LD	User A	0.264	1.063	4.130	16.945	65.736
	Query Party B	3.213	13.176	53.669	219.830	917.323
	Server C	0.030	0.031	0.033	0.038	0.041

Table 3. Runtime vs. participant count under fixed set size (Unit: seconds)

Protocol / Participant Count		2^8	2^9	2^{10}	2^{11}	2^{12}	2^{13}	2^{14}
O-PSI [22]	A	0.223	0.224	0.224	0.226	0.226	0.226	0.226
	B	25.112	25.253	25.339	25.478	25.389	25.560	2.367
	C	65.736	143.758	316.931	673.173	1423.146	2973.292	5998.584
Feather [27]	A	0.106	0.104	0.113	0.106	0.108	0.122	0.138
	B	1.168	1.234	1.310	1.644	1.752	1.914	2.367
	C	25.826	51.423	106.86	218.723	449.75	913.426	1839.17
DMPSI -LD	A	0.079	0.081	0.079	0.079	0.080	0.079	0.081
	B	0.832	0.841	0.813	0.832	0.867	0.824	0.860
	C	1.530	3.063	6.019	12.376	25.082	50.310	98.786

Table 1 presents a comparison of the three protocols in terms of time and communication complexity, as well as whether a secure channel is required.

Table 2 shows the impact of the number of user data on the time changes of each participant in the protocol operation. The experiment fixes the number of participants t as 5 (where the number of users is 4 and the number of querying parties is 1), and the maximum amount of aggregated data d is from 2^{11} to 2^{15}.

Table 3 shows the impact of the number of user data on the time changes of each participant in the protocol operation. The set size d of users and query parties is fixed to 1024, the number of participants t is increased from 2^8 to 2^{14}.

From Table 1, DMPSI-LD has the following two advantages over O-PSI and Feather in terms of design principle: (a) DMPSI-LD does not require a secure channel. In DMPSI-LD, users blind point-valued polynomials using random polynomials generated by themselves, and the random numbers of the blinded roots use double encryption to prevent the querying party B from eavesdropping on the channel to obtain them; (b) the computational complexity of the server in DMPSI-LD is linear in the size of the maximum set of users, which is an order of magnitude lower than O-PSI and Feather.

From Table 2, it can be seen that with the same parameter settings, as the data set size increases exponentially, the total run time is O-PSI > Feather > DMPSI-LD. From the perspective of each participant, DMPSI-LD has three main advantages: (a) In the case where the data volume is 2^{11}, the running time of DMPSI-LD users is 2/3 of that of Feather and O-PSI users; (b) the running time of the DMPSI-LD query party is 2/3 of that of Feather, and when the dataset size is 2^{11}, the running time of the DMPSI-LD query party is 1/16 of that of O-PSI; (c) the running time of the DMPSI-LD server is basically unaffected. From the data, when the dataset size is 2^{15}, the running time of the DMPSI-LD server is 1/10047 of that of Feather and 1/18209 of that of O-PSI.

From Table 3, it can be seen that with the increase in the number of clients, the running times of users and queryors for the three protocols are not affected, but the running time of the server shows a linear growth. Specifically, the advantages of the DMPSI protocol over the O-PSI protocol and Feather are as follows: (1) Although the number of participants does not affect the running times of users and queryors, the running time of users for DMPSI-LD is approximately 1/2 of that for Feather and 1/4 of that for O-PSI. The running time of queryors for DMPSI-LD is approximately 1/2 of that for Feather and 1/31 of that for O-PSI. (2) The running time of the server for DMPSI-LD increases more gently with the increase in the number of participants. From the specific data analysis, approximately for every 100 additional users, the server running time of the DMPSI-LD protocol increases by 0.006 seconds, while that of Feather increases by 0.113 seconds and that of O-PSI increases by 0.348 seconds. The server running time of DMPSI-LD is 1/18 of that of Feather and 1/60 of that of O-PSI.

From the above analysis, the following conclusions can be drawn:

(1) DMPSI-LD does not rely on secure channels, while O-PSI and Feather must be secured in an environment where secure channels are used.

(2) DMPSI-LD is more suitable for large-scale dataset scenarios than O-PSI and Feather. The time complexity of DMPSI-LD's server is $O(N)$ when the dataset size is the variable, while the time complexity of O-PSI's server and Feather's server are $O(N^2)$.

(3) The running time of DMPSI-LD users and queryors is independent of the number of participants and is highly efficient. Under the condition that the dataset size is 1024 in all cases, the running time of DMPSI-LD users is approximately 1/4 of that of O-PSI and 1/2 of that of Feather. The running time of DMPSI-LD queryors is approximately 1/31 of that of O-PSI and 1/2 of that of Feather.

6 Conclusion

This paper presented an efficient delegated multi-party private set intersection protocol for large-scale datasets (DMPSI-LD) in a cloud environment, which overcome the deficiency of the security channel in the Feather protocol. The protocol achieved the characteristic of one-time delegation and multiple intersection computations without the need for a secure channel. Moreover, the running time of users and query parties was independent of the number of participants, and the computational cost of the server increases linearly with the number of participants and the size of the data set. The article detailed the protocol flow, provided a rigorous security proof and experimental analysis, and offered beneficial references for research in the PSI field.

Acknowledgments. This work is funded by the Hubei Provincial Department of Education Project (No. Q20231409).

References

1. Dwork, C.: Differential privacy. In: Proceedings of the 33rd International Colloquium on Automata, Languages and Programming, pp. 1–12. Springer, Venice, Italy (2006)
2. Chen, H., Laine, K., Rindal, P.: Fast private set intersection from homomorphic encryption. In: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, pp. 1243–1255. ACM, Dallas, TX, USA (2017)
3. Kales, D., Mariani, C., Paterson, K.G.: Private contact discovery in mobile messaging systems with stronger security guarantees. In: Proceedings of the 2019 ACM Asia Conference on Computer and Communications Security, pp. 183–197. ACM, Auckland, New Zealand (2019)
4. Zhang, Y., Li, S., Wang, X., et al.: Post-quantum PSI via lattice-based authenticated key exchange. IEEE Access 8, 142301–142312 (2020)
5. Meadows, C.: A more efficient cryptographic matchmaking protocol. In: Proceedings of the 1986 IEEE Symposium on Security and Privacy, pp. 134–147. IEEE, Oakland, CA, USA (1986)
6. Freedman, M.J., Nissim, K., Pinkas, B.: Efficient private matching and set intersection. In: Advances in Cryptology – EUROCRYPT 2004, pp. 1–19. Springer, Interlaken, Switzerland (2004)
7. Pinkas, B., Rosulek, M., Trieu, N., et al.: PSI from PaXoS: Fast, malicious private set intersection. In: Eurocrypt 2020, pp. 739–767. Springer (2020)
8. Marston, S., Li, Z., Bandyopadhyay, S., Zhang, J., Ghalsasi, A.: Cloud computing—The business perspective. Decis. Support Syst. 51(1), 176–189 (2011)
9. Abadi, M., Terzis, S., Metere, R., Dong, C.: Efficient Delegated Private Set Intersection on Outsourced Private Datasets. IEEE Trans. Secure Comput. 16(4), 608–624 (2017)
10. Zhang, J., Kang, X., Liu, Y., et al.: A secure and lightweight multi-party private intersection-sum scheme over a symmetric cryptosystem. Symmetry 15(2), 319 (2023). https://doi.org/10.3390/sym15020319.
11. Diffie, W., Hellman, M.: New directions in cryptography. IEEE Trans. Inf. Theory 22(6), 644–654 (1976)
12. Huang, Y., Evans, D., Katz, J.: Private set intersection: Are garbled circuits better than custom protocols? In: NDSS 2012, pp. 1–15 (2012)

13. Pinkas, B., Schneider, T., Zohner, M.: Faster private set intersection based on OT extension. In: USENIX Security Symposium, pp. 797–812 (2014)
14. Abadi, A., Terzis, S., Dong, C.: VD-PSI: Verifiable delegated private set intersection on outsourced private datasets. In: International Conference on Financial Cryptography and Data Security, pp. 149–168. Springer (2016)
15. Rindal, P., Rosulek, M.: Malicious-secure private set intersection via dual execution. In: USENIX Security Symposium, pp. 1229–1246 (2017)
16. Kiss, Á., Liu, J., Schneider, T., et al.: Private set intersection for unequal set sizes with mobile applications. Proc. Priv. Enhancing Technol. 2017(4), 177–197 (2017)
17. Rosulek, M., Trieu, N.: Compact and malicious private set intersection for small sets. In: Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security, pp. 1166–1181. ACM (2021)
18. Kerschbaum, F.: Collusion-resistant outsourcing of private set intersection. In: Proceedings of the 27th Annual ACM Symposium on Applied Computing (SAC 2012), pp. 1451–1456. ACM, Trento, Italy (2012)
19. Kerschbaum, F.: Outsourced private set intersection using homomorphic encryption. In: Proceedings of the 7th ACM Symposium on Information, Computer and Communications Security (ASIACCS 2012), pp. 85–86. ACM, Seoul, South Korea (2012)
20. Liu, F., Zhang, W., Giang, D.H., Han, S.: Encrypted set intersection protocol for outsourced datasets. In: 2014 IEEE International Conference on Cloud Engineering (IC2E 2014), pp. 135–140. IEEE, Boston, MA (2014)
21. Kamara, S., Mohassel, P., Raykova, M., Sadeghian, S.: Scaling private set intersection to billion-element sets. In: Proc. Int. Conf. Financial Cryptography and Data Security, pp. 195–215 (2014)
22. Abadi, M., Terzis, S., Metere, R., Dong, C.: Efficient Delegated Private Set Intersection on Outsourced Private Datasets. IEEE Trans. Secure Comput. 16(4), 608–624 (2017)
23. Ruan, O., Wang, Z.H., Mi, J., Zhang, M.W.: New approach to set representation and practical private set-intersection protocols. IEEE Access 15(7), 64897–64906 (2019)
24. Oliaee, M., Delavar, M., Ameri, M.: On the security of O-PSI: A delegated private set intersection on outsourced datasets. Inf. Secur. Cryptol, 77–81 (2017)
25. Yang, X.Y., et al.: Improved outsourced private set intersection protocol based on polynomial interpolation. Concurr. Comput. Pract. Exp. (2018)
26. Tajima, A., Sato, H., Yamana, H.: Outsourced private set intersection cardinality with fully homomorphic encryption. In: 2018 6th International Conference on Multimedia Computing and Systems, pp. 1–8 (2018)
27. Abadi, A., Dong, C., Steven, J.: Multi-party updatable delegated private set intersection. In: Proc. Int. Conf. Financial Cryptography and Data Security: 26th International Conference (FC 2022), vol. 13411. Springer, Grenada (2022)
28. Hu, J., Liu, Z., Zuo, C.: Delegated multi-party private set intersection from secret sharing (2025). https://eprint.iacr.org/2025/030, last accessed 2025/01/10