International Journal of Software and Informatics, ISSN 1673-7288
http://www.ijsi.org, ijsi@iscas.ac.cn, +86-10-62661048
IJSI, 2022, 12(1): 107–129, doi: 10.21655/ijsi.1673-7288.00273

International Journal of Software and Informatics

Research Article

A Secure Multi-party Data Federation System

Shuyuan Li (李书缘)[1,2], Yudian Ji (季与点)[3], Dingyuan Shi (史鼎元)[1,2], Wangdong Liao (廖旺冬)[1,2], Lipeng Zhang (张利鹏)[1,2], Yongxin Tong (童咏昕)[1,2], Ke Xu (许可)[1,2]

[1] (State Key Laboratory of Software Development Environment (Beihang University), Beijing 100191, China)
[2] (School of Computer Science and Engineering, Beihang University, Beijing 100191, China)
[3] (Information Center of Ministry of Science and Technology, Beijing 100862, China)
Corresponding author: Yongxin Tong, yxtong@buaa.edu.cn

Abstract In the era of big data, data is of great value as an essential factor in production. It is of great significance to implement its analysis, mining, and utilization of large-scale data via data sharing. However, due to the heterogeneous dispersion of data and increasingly rigorous privacy protection regulations, data owners cannot arbitrarily share data, and thus data owners are turned into data silos. Since data federation can achieve collaborative queries while preserving the privacy of data silos, we present in this paper a secure multi-party relational data federation system based on the idea of federated computation that "data stays, computation moves." The system is compatible with a variety of relational databases and can shield users from the heterogeneity of the underlying data from multiple data owners. On the basis of secret sharing, the system implements the secure multi-party operator library supporting the secure multi-party basic operations, and the resulting reconstruction process of operators is optimized with higher execution efficiency. On this basis, the system supports query operations such as Summation (SUM), Averaging (AVG), Minimization/Maximization (MIN/MAX), equi-join, and θ-join and makes full use of multi-party features to reduce data interactions among data owners and security overhead, thus effectively supporting efficient data sharing. Finally, experiments are conducted on the benchmark dataset TPC-H. The experimental results show that the system can support more data owners than the current data federation systems SMCQL and Conclave and has higher execution efficiency in a variety of query operations, exceeding the existing systems by as much as 3.75 times.

Keywords data federation; database system; secure multi-party computation

Citation Li SY, Ji YD, Shi DY, Liao WD, Zhang LP, Tong YX, Xu K. A secure multi-party data federation system. *International Journal of Software and Informatics*, 2022, 12(1): 107–129. http://www.ijsi.org/1673-7288/273.htm

With the advent and rapid development of the era of big data, data as a factor of production

This is the English version of the Chinese article "面向多方安全的数据联邦系统. 软件学报, 2022, 33(3): 1111–1127. doi: 10.13328/j.cnki.jos.006258".
Funding items: National Key Research and Development Program of China (2018AAA0101100); National Natural Science Foundation of China (61822201, U1811463, 62076017); the CCF-Huawei Database System Innovation Research Plan (CCF-HuaweiDBIR2020008B); State Key Laboratory of Software Development Environment (Beihang University) Open Program (SKLSDE-2020ZX-07)
Received 2021-07-01; Revised 2021-07-31; Accepted 2021-09-13; IJSI published online 2022-03-28

has become more and more important in the application of various industries, and it is of great significance to realize data sharing. The Opinions of the Central Committee of the Communist Party of China and the State Council on Building Improved Systems and Mechanisms for Market Allocation of Factors stated that China should accelerate the cultivation of data factor markets. Specifically, it emphasizes the need to promote data opening and sharing and cultivate new industries in the digital economy. It is clear that data sharing and data value mining are of strategic importance to the future economic and social development of the country. The benefits of data sharing are obvious. For example, when a patient has medical records and examination reports filed in several hospitals, it will not only help the patient avoid repeated examinations but also provide a more comprehensive understanding of the patient's medical history and other conditions if these data can be shared among hospitals; when government data are stored in different information systems of different government departments, we can optimize government services if we break the barriers to achieve the integration of big data on government services.

However, data sharing is severely limited in reality. Data is scattered and heterogeneous among a large number of individuals, which makes data aggregation extremely difficult and limits the sharing of data, a phenomenon known as the problem of "data silo". Thus, corresponding data integration technologies have been developed[1] to address this problem. However, data privacy protection has become a widespread consensus worldwide in recent years. For instance, the General Data Protection Regulation (GDPR) was issued by the EU on May 25, 2018, and the Law on Personal Information Protection (Draft for Second Deliberation) and the Data Security Bill (Draft for Second Deliberation) were published by the Standing Committee of the Chinese National People's Congress on April 30, 2021. All of these regulatory acts impose restrictions on the processing and circulation of data, making it difficult for data sharing to be widely implemented.

The increasingly stringent privacy protection requirements further exacerbate the "data silo" phenomenon, in which data owners can only use the small-scale data they hold, and data can hardly converge through sharing and is unable to give full play to its value. Only by connecting silos and breaking sharing barriers among data can allow data to be used as a factor of production to drive economic and social development effectively[2]. Under this condition, the concept of federated computing was born, in which federation refers to a collection of independent and autonomous data owners. The data owners in the federation ensure that the original sensitive data does not leave original data owners to protect privacy. The core idea of privacy protection in federated computing is that "data stays, computation moves", namely that each data owner first computes the data in each party and then aggregates the intermediate computations to obtain the final results instead of sharing the data directly. This mode splits the computation to each party to avoid the local flow of data, and thus the idea that "data stays, computation moves" in federated computing can guide the data sharing mode under the requirement of privacy protection. On the premise of "no data leaving original data owners" in federated computing, the following challenges exist in building a data federation system under the above data sharing mode.

- It is difficult to guarantee the privacy requirements for data sharing. Effective data sharing cannot be achieved without the participation of multiple data owners; however, the underlying security operation design is a challenge for the federation system to support multiple data owners.
- It is difficult to guarantee the efficient and secure querying of data sharing. The data sharing process in a federation scenario needs to protect data privacy and security, which requires a secure design of the federation system. However, secure computing is costly, and ensuring efficient sharing of large-scale data also poses a challenge.
- It is difficult to guarantee heterogeneous multi-party collaboration for data sharing. In

a data sharing scenario, heterogeneity problems exist among data owners, including database system heterogeneity and data schema heterogeneity. Thus, the system should also adapt to heterogeneous parties while protecting privacy.

No system has been able to solve the above three challenges well. In the 1980s, federated databases were essentially middleware coordinating multiple databases to accomplish a query. However, those systems focused on solving the heterogeneity of multiple databases and the disassembly and rewriting of federated queries and paid no attention to the privacy issues in federated queries. Until the beginning of the 21 st century, some secure multi-party computation tool libraries[3, 4] were gradually open-sourced for easy development and use. As a result, data federation systems begin to develop since 2017, and unlike federation databases, a data federation system emphasizes more on data privacy protection for data owners. Some scholars have attempted to combine secure multi-party computation technologies with federation ideas to build privacy-preserving data federation systems[5, 6]. However, those efforts are limited by the use of secure multi-party computation tool libraries that can only support the participation of two or three data owners.

To address the above challenges, we present a secure multi-party data federation system. This system consists of the system adaptation, secure multi-party operator library, query engine, and interactive interface. It supports multiple heterogeneous databases, multiple secure query operations including θ-join, and a unified user-oriented SQL Language and graphical user interface. In this way, this system enables high efficiency of secure multi-party data sharing, and its main contributions are as follows.

- The data federation system for multi-party security is built. The system can support secure data sharing with multiple data owners with more than three parties. Meanwhile, it is equipped with an adaptation interface to shield the system from the heterogeneity of each data owner's database system. It also provides a user-friendly graphical interface, supports SQL queries, and can adapt to various query interfaces.
- It implements a secure multi-party operator library with high efficiency that supports addition, subtraction, multiplication, division, and comparison. The implementation of the operator library is based on the secret sharing framework, and the basic operations covered can support basic data query operations. Efficient result reconstruction is realized considering the features in secret sharing.
- An efficiently secure multi-party query engine including θ-join is achieved. On the basis of the aforementioned basic operators, the query engine can carry out query operations such as SUM, AVG, MIN/MAX, equi-join, and θ-join. The design process also takes into account the characteristics of multiple parties and uses the transferability of the join process to ensure the efficiency of query operations.

The structure of this paper is as follows: Section 1 introduces the background of the system. Section 2 demonstrates the system architecture and system workflow. Sections 3 and 4 illustrate the secure multi-party operator library and query engine, respectively. Section 5 presents the system adaptation and interactive interfaces. Section 6 shows the system performance verification results on a standard test set. Section 7 introduces related work. Finally, Section 8 concludes the paper with an outlook.

1 Background

In the context of increasingly strict data privacy and security protection, as well as serious "data silos" problems, data federation is an important idea to achieve data sharing. Specifically, the data federation F can be considered as a collection of n data owners and the central server C, where $n \geq 3$. In data federation, the databases of different data owners are heterogeneous.

In addition, for practical application scenarios, the number (n) of data owners is usually more than three. For example, in an epidemiological survey, the cooperation of multiple data owners such as hospitals, map applications, mobile payment, and online taxi platforms may be required. In a taxi union, the number of taxi companies participating in the union may range from a dozen to several dozen. In some scenarios, the computation task of the central server of the data federation can be rotated among data owners.

For the i-th data owner P_i in the data federation, the data owned by P_i is assumed to be d_i, and the safety model followed by each data owner is presumed to be semi-honest[7]. In the semi-honest model, the participants are "honest-but-curious", namely that they follow the protocol and the code to be executed honestly, but they infer information from the data obtained during the execution. If they can infer the information that should be protected, the system is insecure. The semi-honest model is a widely used assumption in the field of security, and it has been adopted by existing representative secure multi-party database systems[5, 6].

For data sharing in federation scenarios, the data federation system should support secure multi-party federation query operations. The definition is as follows:

Definition 1 (Federated query operation)**.** For federated query operation $f\colon D^n \to R$, its result shall be consistent with the result of the corresponding conventional database query operation, i.e., $f(d_1, d_2, \cdots, d_n) = f'(d_1 \cup d_2 \cup \cdots \cup d_n)$, and in the computation, data d_i of the data owner P_i shall not be disclosed to any other data owner.

The basic federated query operations include federated SUM (f-SUM), federated AVG (f-AVG), federated MIN/MAX (f-MIN/MAX), federated equi-join (f-equi-join), and federated arbitrary join (f-θ-join). These query operations shall support multi-party heterogeneous databases and ensure security. In other words, each data owner shall ensure that the original data does not leave its original data owner when data owners are computing together. The data sharing under this security guarantee requires that each data owner should not send its original data directly to other parties for computation during the execution of query operations. This ensures that each data owner can jointly obtain the results of the query operation after completing the query operation but cannot obtain the original data of other data owners, thus protecting the data privacy of each party.

The following example is the application in the medical field in combination with the above definition. For example, if a patient has medical records and examination reports filed in multiple hospitals, sharing these data among hospitals will not only help the patient avoid duplicate examinations but also provide a more comprehensive understanding of the patient's medical history. In this application case, each hospital, a data owner, has a large amount of medical data on this patient, and all these together constitute a data federation. To protect the data privacy of the patient, multiple hospitals need to share data securely. For instance, the average value of the patient's previous liver function test indexes can be queried by first f-equi-join and then f-AVG of the liver function test data table according to the ID number of this patient on the premise of no patient data leaving its original data owner.

In the above scenario, the main objectives of system design are the data security objective due to the federation scenario, the efficiency objective due to the resulting computation overhead control, and the usability objective due to the heterogeneity of multiple databases. The following three system objectives, i.e., security, efficiency, and usability objectives, correspond to the three challenges mentioned above, namely, the difficulty of guaranteeing privacy and security, the difficulty of efficient and secure query, and the difficulty of heterogeneous multi-party collaboration, respectively.

(1) Security objective

In a federation scenario, multiple parties cannot trust each other, and data cannot leave

its original data owner. Other data owners may make inferences in view of public information available. For this reason, neither the querying user nor any parties can infer any additional information from the public information, except that query statements and query results are publicly accessible to all data owners. For example, the user and all data participants should only know the final SUM result in a secure multi-party SUM operation but cannot infer the exact value of each party's respective participation in the SUM. In this way, the data owner's data can be prevented from being stolen during the computation. This is a common requirement for query operations in secure multi-party database systems[5, 6].

(2) Efficiency objective

In a federation scenario, the computation overhead of query operations has two main sources. Firstly, the protected object of security operations is data, and hence the amount of protected data is large, especially in join operations where the protected object is a data column, and a large amount of data leads to a high protection cost. Secondly, the query operation in the data federation scenario requires the participation of multiple parties, and solving the heterogeneity problem of each data owner and the collaboration problem of multiple parties will also bring overhead. To ensure the efficiency of the system, we need to design optimization modules to address these sources of overhead. For the overhead caused by the large amount of protected data, we need to design efficient security operators to optimize the operation efficiency; for the overhead caused by the coordination of multiple parties, we should flexibly take advantage of the parallelism of multiple parties to improve the efficiency. The operating efficiency of the system will affect its performance in the application. In a taxi union, the operating efficiency determines the response time of the union platform to orders; the operating efficiency of the system determines the time spent on case tracing and close contact investigation in an epidemiological survey.

(3) Usability objective

In a federation scenario, the database system of each data owner is heterogeneous, and therefore the system shall realize the adaptation and unification of heterogeneous databases. In addition, a unified SQL query interface and graphical management interface shall be built for users to facilitate the coordination of multiple parties.

The system is designed to achieve the above objectives, and the specific architecture and process design are described in Section 2.

2 System Overview

2.1 System architecture

The architecture of the proposed system is shown in Figure 1. The system is divided into four layers, from the bottom multi-party database to the top, which are the system adaptation, secure multi-party operator library, query engine, and interactive interface, respectively. At the bottom, there are the databases of the different data owners in the data federation. In order to screen the heterogeneity of the databases, we write adaptation modules for different kinds of databases to complete the system adaptation, which corresponds to the objective design in Section 1.2. On top of the system adaptation, the secure multi-party operator library and the query engine are the core modules of the system, which are designed to meet the security and efficiency objectives in Section 1.2, respectively, and thus the system can support secure and efficient multi-party data sharing. The graphical interface and the SQL query interface in the interactive interface are designed to facilitate multi-user coordination, and together with the system adaptation, they accomplish the usability objective design in Section 1.2. The specific functions of each layer are as follows.

The bottom layer of the system is designed for each data owner and is comprised of the

database system of each data owner. Considering the heterogeneity among these database systems, the proposed system can support multiple database systems.

System adaptation: The proposed system can adapt to various query computations issued by the upper secure multi-party operator library and to the underlying multi-party heterogeneous data. The system shall first adapt to database systems. At present, it has been adapted to various database systems such as MySQL, openGauss, SQL Server, and PostgreSQL. Then, adaptations shall be made to different data schemas to build a unified user-oriented data view.

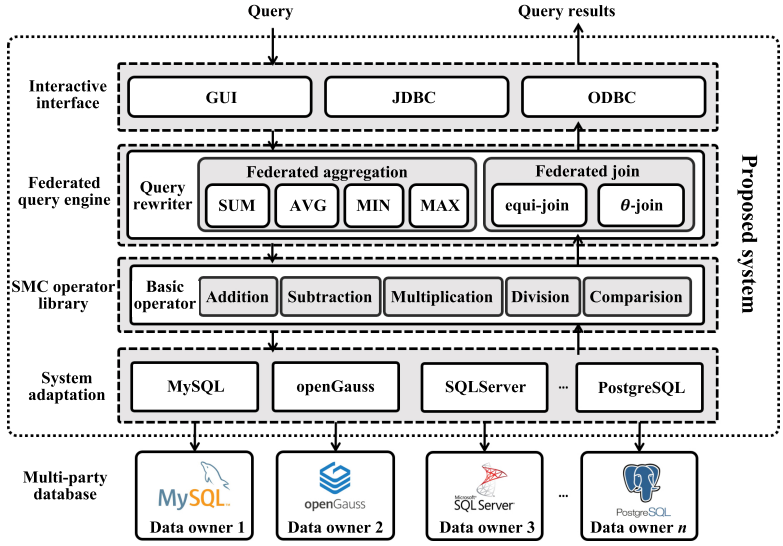


Figure 1 Architecture of the data federation system based on secure multi-party computation

Secure multi-party operator library: This operator library mainly contains basic operators for multi-party security. Each basic operator is mainly used for simple computations involving multiple data owners, such as the SUM and the quadrature operators for multi-party security. During the computation process, each data owner will be called to query the local database through the adaptation interface. The secure multi-party operator library is the basis for developing subsequent federated query operations.

Query engine: This query engine takes over the queries obtained from the upper-layer interactive interface and then parses and rewrites them into the corresponding federated query operations. There are essentially two types of federated aggregations and federated joins involved in multi-party federated computations. The federated aggregation operation supports SUM, AVG, and MIN/MAX on the basis of the lower layer, and the federated joins support both equi-join and θ-join.

Interactive interface: This interface provides users with a graphical interface to improve the usability of the system. It receives queries from users through the interface and passes them to the lower-layer query engine. The system supports the use of SQL query statements by users.

As previously mentioned, the secure multi-party operator library and the query engine are the core components of the system, and their implementation is described in Sections 3 and 4, respectively.

2.2 System workflow

The workflow of the proposed system is shown in Figure 2. Users first enter an SQL query according to the unified view provided by the system. The central server of the system

receives the query, parses it into a syntax tree, and performs predicate push-down optimization on the SQL query to be closer to the data source under the filtering condition. Then, for query operations that involve the joint participation of multiple data owners, the query is rewritten as a multi-party security-oriented federated query operation method. The server then sends the execution plan of the rewritten federated query operation to each data owner, and the data owner executes the rewriting process in two parts: firstly, it executes the query operation on the local database and obtains the corresponding query results; secondly, it uses basic operators for multi-party security to execute the part involving joint computation by multiple parties according to the protocol. Finally, the federated query results are sent to the server, and the final query results are obtained and returned to the user with methods such as secret sharing and secure set merging on the premise of protecting the source of each federated query result.

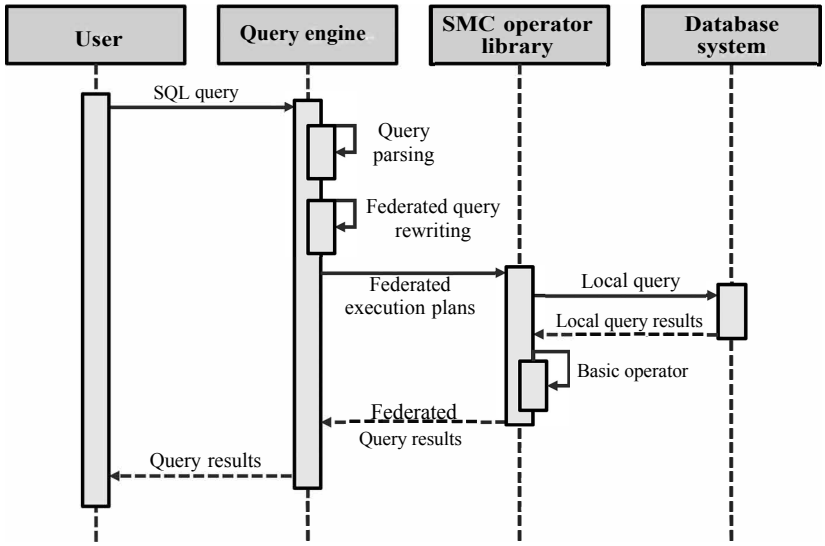


Figure 2 System workflow of query processing

3 Secure Multi-party Operator Library

The secure multi-party operator library of this system supports multi-party security-oriented computations, including addition, subtraction, multiplication, division, and comparison, and thus it helps realize a variety of query operations in the system. Each operator ensures data security with the secure multi-party computation technology based on secret sharing. The main reason for adopting this technology instead of a garbled circuit is that secret sharing can better support multiple parties. The existing data federation systems such as SMCQL[5] and Conclave[6] are based on garbled circuit tool libraries such as ObliVM[3] or Obliv-C[4], which cannot involve more than three parties to participate in the computation. In contrast, secure computation protocols based on secret sharing can well support multi-party participation. Sharemind[8] and MP-SPDZ[9] are the main tool libraries for secret sharing technology. The former is not open-sourced, which limits its application; interface encapsulation of the latter is mainly computation-oriented, and its multi-party interaction mode is not compatible with data federation systems. Therefore, it is necessary to customize the basic computation library based on secret sharing and multi-party security for the proposed system. In this paper, we introduce the basic framework of this operator library and the execution process of operators in Sections 3.1 and 3.2, respectively, and elaborate on the example of the SUM operator.

3.1 Framework of operator library

The system is designed to implement the basic operators on the basis of secret sharing of the Shamir(t, n) model[10]. This model ensures that when a secret is scattered among n data owners, at least t data owners need to participate to reconstruct the secret together. On this basis, the framework of the secure multi-party operator library is divided into three phases: secret distribution, local computation, and result reconstruction, which are described below.

(1) Secret distribution

In the secret distribution phase, firstly, each data owner is assigned a number (No.); then, according to the set threshold t, each party generates a polynomial with the highest count term as the threshold, and the input data of that party is used as the constant term of the polynomial. Next, the No. of each data owner is brought into the polynomial to calculate its sub-secret that will be further sent to the data owner of that No.

(2) Local computation

After each data owner received the sub-secret from all other parties, the sub-secret is calculated according to different basic operators, and the result is the sub-secret of the final result.

(3) Result reconstruction

Any data owners not less than t send the sub-secrets of the final result obtained in the previous phase to the central server. Since the final objective result is a constant term of a higher-order polynomial, and the order of the polynomial increases with the number of participating data owners, the sub-secret received by the central server is a point on the polynomial. Therefore, the final result of the basic operator can be reconstructed by the central server with the Lagrange interpolation method by solving the higher-order polynomial according to the sub-secret.

Among the above three phases, the high expenditure of the result reconstruction phase is the bottleneck of a secure multi-party operator library as the Lagrange interpolation method used to solve higher-order polynomials in this phase leads to two problems. Firstly, solving higher-order polynomials in the result reconstruction phase gives rise to a high computation overhead, while only simple random number generation and polynomial computation are involved in the secret distribution and local computation phases; secondly, the common Lagrange interpolation method produces errors in solving higher-order polynomials, and these errors generally increase with the order of the polynomials. In view of these two problems, the secure multi-party operator library is improved by applying a divide and conquer approach to obtain the following computational framework. For a basic operator with n data owners involved, the n data owners are first divided into groups, and then the computational results of each group are regarded as the intermediate results of the basic operator through the above three phases. Next, the intermediate results of each group are considered as the new intermediate results through the above three phases, and the process is repeated until the final results of the operator are obtained. This framework reduces the number of participants in each secure multi-party computation by grouping, thus reducing the number of polynomials in it, which can significantly decrease the error associated with the basic operators and improve their operation efficiency.

3.2 Operator execution flow

Given the above secure multi-party operator library framework, the operator library of this system implements basic operators such as addition, subtraction, multiplication, division, and comparison. Taking the addition operator as an example, we introduce it as follows: Algorithm 1 describes the computation process of the addition operator, where lines 1–5 describe the execution steps of the secret distribution phase, and each party randomly generates a $(t-1)$-order polynomial, calculates the sub-secret by substituting the No., and then distributes it;

lines 6–8 describe the execution steps of the local computation phase, where each party sums up the sub-secret for the addition operator; line 9 describes the execution steps of the result reconstruction phase, where the following Eq. (1) is used to calculate the sub-secret for each party to obtain the final result R by substituting this equation into tR_i, respectively.

$$R = \sum_{i=1}^{t} R_i \cdot \prod_{j=1, j\neq i}^{t} \frac{x_j}{x_j - x_i} \tag{1}$$

Algorithm 1. Federated SUM operator

Input: Data $d_1, d_2, \cdots, d_n$ with the No. $x_1, x_2, \cdots, x_n$ of data owners $P_1, P_2, \cdots, P_n$, respectively
Threshold value t
Output: The SUM result of the parties, in which $R = d_1 + d_2 + \cdots + d_n$
1. **for** each data owner P_i **do**
2. Generate a random polynomial $f_i(x) = d_i + a_1 x + a_2 x^2 + \cdots + a_{t-1} x^{t-1}$
3. Compute the sub-secret S_i, where
4. Distribute the sub-secret and send $S_i[j]$ to P_j
5. **end for**
6. **for** each data owner P_i **do**
7. $R_i \leftarrow \sum\{S_j[i] \mid 1 \le j \le n, j \neq i\}$
8. **end for**
9. Summarize R_i of any t parties, and solve the final results according to Eq. (1)
10. **return** R

Figure 3 illustrates the operation flow of the addition operator with the participation of three data owners. The data of each data owner i is d_i, and the No. is x_i. First, each data owner generates two random numbers a_{i1} and a_{i2} and obtains a random polynomial f_i. Then, the Nos. are substituted into f_i for computation, and the sub-secret $f(x_i)$ is sent to the i-th party to complete the secret distribution phase. In the local computation phase, each data owner sums up the obtained sub-secrets, i.e., $f_1(x_i) + f_2(x_i) + f_3(x_i)$, which is equivalent to constructing a polynomial S with $\sum d_i$ as constant term. In the final result reconstruction phase, the polynomial S is solved with $S(x_1)$, $S(x_2)$, and $S(x_3)$, and the constant term is the result of the SUM computation after solving the polynomial.

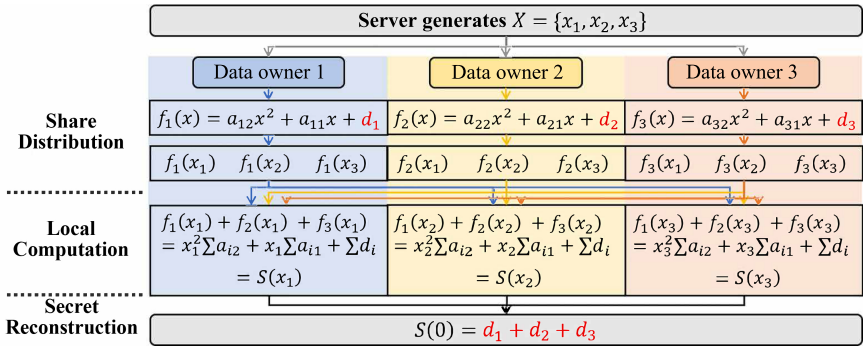


Figure 3 Operation flow of SUM operator in secure multi-party operator library

4 Query Engine

4.1 Design principles

Unlike a conventional database system, the query of this system is oriented to the data federation formed by multiple data owners, and the data security and privacy of each party

should be protected. Therefore, when a computation involves only the data of the data owner in the process of query processing, the query can be done directly in plaintext locally without considering the data privacy of that party. However, when a computation involves all data owners and requires multi-party interactions, a large number of security operations have to be performed according to security protocols to protect the data privacy of each party. Therefore, the efficiency of multi-party interactions is the key to the performance of the query engine in this system. Hence, the query engine of this system is designed to **reduce the overhead of multi-party interactions** and for the three steps of query parsing, query plan generation, and query plan execution in the query processing flow. The query engine of this system is designed with the corresponding parsing unit, rewriting unit, and execution unit to complete the above process. The specific design of the three units is as follows.

4.2 Design of parsing unit

A parsing unit receives the SQL query input from the user through the interface, and the central server is responsible for its computation. This unit first parses the input SQL query to produce a query syntax tree. Under the design concept of the query engine, predicates in the resulting syntax tree are pushed down to reduce the interaction overhead of multiple parties. For example, the filtering conditions are pushed down so that the filtering conditions can be executed locally on the plaintext by the data owner as early as possible without considering security issues. In this way, the amount of data is reduced in subsequent query operations involving multiple data owners, and the overhead of subsequent multi-party interactions decreases. Figure 4 shows the flow and effect of the parsing unit with a concrete example: pushing down the operations can lower the amount of data and computation overhead involved in secure multi-party interactions.

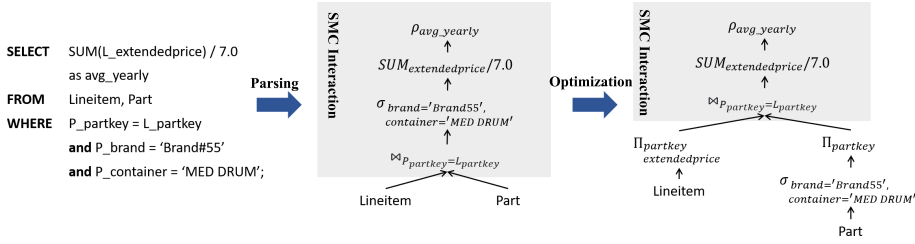


Figure 4 Workflow of parsing unit in this system

4.3 Design of rewriting unit

A rewriting unit is designed to take over the output of the previous parsing unit and generate a query execution plan on the basis of the syntax tree. Since the query operation of some nodes in the syntax tree requires the participation of multiple data owners, the data privacy of all parties should be considered simultaneously. Therefore, the rewriting unit mainly rewrites the query operation involving multiple parties into a federated query operation to protect the data privacy of multiple parties according to the design concept of the query engine, namely, reducing the interaction overhead of multiple parties. The following example is the classical database join operation JOIN, which introduces the rewriting unit design.

(1) Federated JOIN query operation

Since the underlying multi-party database storage of the system is transparent to users, users can initiate a conventional join query operation $L.c \bowtie_{condition} R.c$ according to the unified view provided by the central server. In other words, tuples in the table L and table R are joined by referring to the data column c that meets the condition. The rewriting unit should be oriented to the underlying multiple data owners, and the tables L and R considering views of users are

scattered among multiple parties. Since multiple data owners hold parts of tables L and R, when the JOIN operation is jointly computed by multiple parties, any two of them are required to perform the JOIN operation once. Then, all the intermediate results of joins are aggregated to obtain the final join result. Thus, the global result of the federated query operation is obtained by aggregating the results of multiple two-party joins. Using one of the two joins as an example, we present how to rewrite it in a data federation scenario in the next paragraph.

(2) Two-party JOIN query operation

When two data owners make a secure join to protect data privacy, the data owners P_1 and P_2 hold a part of the data L_1 and R_2 in the table L and table R, respectively. If the entire data tables L_1 and R_2 are joined directly as input to a secure protocol, it will result in a significant overhead of interactions between the two parties[5]. Therefore, according to the design concept of the query engine, this rewriting unit rewrites the JOIN operation flow to reduce the communication traffic between the two parties considering the high overhead. Firstly, the data column c is ordered in plaintext on the data owner's side, and then the value of the i-throw of c in L_1 is compared with the value of the j-th row of c in R_2 in a secure way to determine whether the *condition* is satisfied. In this way, we can determine whether the tuples in the i-throw and j-th row of the two data tables need to be joined without revealing the data values to each other. If the secure comparison result meets the *condition*, the tuples can be sent to the central server, and the central server can join the two tuples; if the secure comparison result does not meet the *condition*, the two tuples do not need to be joined. We can repeat the above steps to compare the value of the i-throw of c in L_1 with the other rows of c in R_2 until all the values of c in L_1 are compared with those of R_2.

The above process is designed to reduce the interaction overhead between two parties, and thus only one data column of the data table is involved in the computation of the secure comparison protocol between two data owners. The system is based on the idea of transferability to further reduce the interaction overhead of the joins between two parties. Since the data columns of the JOIN operation are ordered locally by the two parties, the value of the i-throw of the data column c in the data table L_1 can be located by a binary search without traversing all values of the data column c in the data table R_2. The binary search lessens the number of secure comparisons that need to be made between the two joins, thus reducing the interaction overhead.

(3) Global JOIN query operation

The above rewriting process can safely complete the JOIN operation between two data owners; however, in a federated join query, every data owner P_i holds data tables L_i and R_i, and hence there is still a need to design how to obtain the global join operation result from the aggregation of multiple two-party joins. This unit is also based on the idea of transferability to reduce the communication traffic among multiple parties in the aggregation process. In the case of equi-joins, when the data owner P_1 makes a secure join with both the data owner P_2 and the data owner P_3, the data owner P_1 can infer that both P_2 and P_3 have the intersection part from the intersection of the two joins. Then, the secure join operation between P_2 and P_3 does not need to repeat the secure comparison of this part. By organizing the JOIN operations among multiple parties in this way, the amount of computation involved in the secure join between the two parties is reduced and the interaction overhead is cut down.

4.4 Design of execution unit

An execution unit takes over the federated query plan developed in the previous unit and schedules each data owner to execute the plan. Each federated query operation in the plan is rewritten in the previous rewriting unit, and these federated query operations are executed by calling the corresponding secure multi-party basic operators in the operator library. For

example, for a federated JOIN query operation, a comparison operator is called to determine whether the join constraints are met; for a federated SUM query operation, after each data owner locally executes the SUM operation, a secure SUM operator is called on the resulting value to generate the result of the federated SUM query operation.

The flow of basic operators and query operations designed by the above unit has good parallelizability. In this execution unit, each data owner is scheduled to parallelize some of the computation operations in the corresponding process to improve the operational efficiency of the system. For the execution of basic operators, since operators in a multi-party security operator library are designed in view of a secret sharing framework, they are computed in groups according to the idea of divide and conquer. Therefore, the computation of operators can be executed in parallel in groups. For the federated aggregation query operation, each data owner first performs the corresponding database query operation locally, thus this local computation can be executed in parallel by all parties, and then each party calls the operator from the library to aggregate the intermediate results from multiple parties, which can be parallelized with the aforementioned basic operator. For the federated join query operation, since the global join result consists of multiple two-party join results, each two-party execution of the JOIN operation can be parallelized on a two-party basis.

4.5 Security analysis on query operations

This secure multi-party data federation system is developed for data federation; the original data of each data owner must be protected locally, namely that each party can know the result after executing the query operation process but cannot know the original data of other parties. The security analysis on basic operators, aggregation operations, and join operations is as follows.

Security analysis on basic operators: The secure multi-party operator library of this system is based on the secure multi-party computation technology of secret sharing. According to the definition of secure multi-party computation, such operators are employed to ensure that each party does not obtain the result of the function $f(x_1, x_2, \cdots, x_n)$ without the collaboration of other parties x_i[11]. Therefore, operations in the secure multi-party operator library of this system can ensure that the data of each data owner is not revealed to other parties, and meanwhile, the computational results are obtained by completing the computation requirements. In this way, the security requirements of the data federation scenario are fulfilled.

Security analysis on federated aggregation query operations: This system supports aggregation operations, and the computation process of such operations is comprised of local computations and multi-party interactions. The former is realized by each data owner on their own data, involving no security issues of data privacy leakage. The latter directly calls a basic operator to perform an aggregation calculation for the final aggregation result, and thus its security is guaranteed by the basic operator, which has been analyzed in the previous paragraph.

Security analysis on federated join query operations: This system supports join operations. The computation process is mainly divided into two-party joins of all parties, which are then aggregated to obtain global join results. When two parties are joined, a secure comparison operator is first called on the data columns based on the join to determine whether the join conditions are met, and in this determination process, the secure comparison operator ensures that both parties are informed of the result without knowing each other's data. Then, both parties send the eligible tuples to the central server for the join in light of the determination result, and these tuples are not protected as join results, which complies with the system security requirements. When aggregating the global join results, the aggregation process does not need to be protected because the computation calls only the join results obtained by each party, and those join results are included in the final global join results. When the federated join operation

is followed by the aggregation operation, instead of sending tuples to the central server for the join, each party will perform the federated aggregation operation directly on the tuple that is determined to be eligible for the join to avoid join result leakage. In summary, the federated join query operation meets the system security requirements.

5 System Adaptation and Interactive Interface

5.1 System adaptation

The proposed system provides interfaces for different database systems and different data table schemas to address the problem of data heterogeneity among multiple data owners. For different database systems of the underlying data owners, this system has been adapted to the operation interface of various database systems such as MySQL, openGauss, SQL Server, and PostgreSQL. For various data table schemas of the underlying data owners, this system provides the upload interface of a data table schema for each data owner in the view building step, and each party transmits the data schema shared by the data federation to the central server through the interface. The central server builds a unified data view for users according to the data schema of each party and constructs a mapping relationship between the unified view and the data schema of each party to shield the users from the underlying data heterogeneity.

5.2 Interactive interface

The proposed system provides a user-friendly graphical interface. The interface has two main functions: one is to show a unified data view of the data federation to users for queries based on this data view, and the other is to provide users with query operation functions. This system supports SQL statement queries, and thus users can enter SQL queries in the corresponding operation box to make a joint query of multiple data (Figure 5).

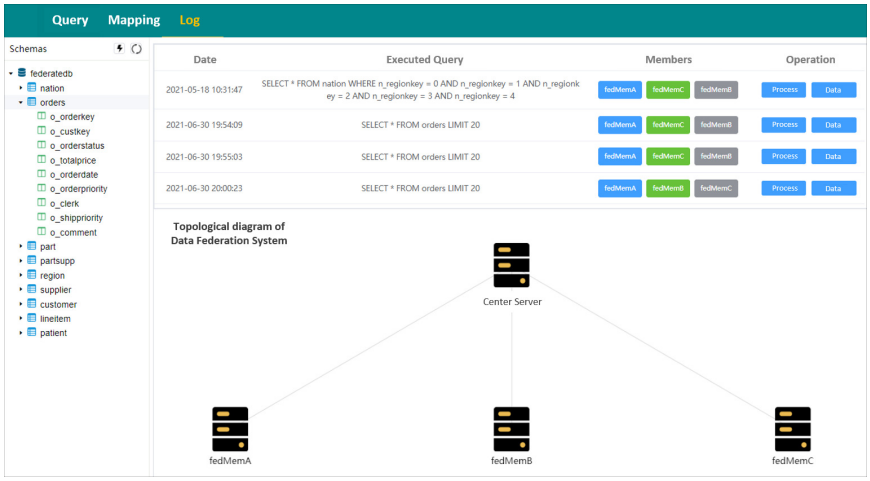


Figure 5 Interactive interface of proposed system

6 System Performance Verification

6.1 Experimental setup

6.1.1 *Experimental environment*

This experiment requires a federation scenario, where multiple machines simulate the participation of multiple data owners. Each machine is equipped with one CPU (model: Intel®

Xeon® Platinum 8269CY CPU T 3.10 GHz) and 32 GB of memory, and the operating system is Ubuntu 18.04.5 LTS (Bionic Beaver). Specifically, one machine was chosen to run the central server process, and each of the other machines ran a data ownership process and built its database for each data owner. During the execution of the query operation, the processes collaborated with each other to complete the computation. Since different data participants are located on different machines, the data of each party is physically decentralized for storage. Unlike the simulation of a federation scenario on a single machine by means of virtual machines, the decentralized storage of data makes it easier to verify that "no data leaves the original data owner" and the experimental setup is more realistic.

6.1.2 *Test data*

The test dataset chosen for the experiment is TPC Benchmark™H (TPC-H)[12]. This dataset consists of data tables containing information on various products. We chose 1 GB of data to construct the database of each data owner. The TPC-H dataset is widely used in the database system performance test and analysis[13], and its experimental results are useful for reference.

6.1.3 *Evaluation index*

The execution time of the query statements was used as the quantitative evaluation index of the system operation efficiency in the experiment.

6.1.4 *Comparison system*

The two most representative data federation systems that support multi-party security, i.e., SMCQL[5] and Conclave[6], were selected for this experiment. In addition, the secure multi-party computation tool library, MP-SPDZ[9], was adopted as the comparison system to further verify the efficiency and scalability of the proposed system.

6.2 System performance comparison

Firstly, we analyzed the performance difference between the proposed system and the representative secure multi-party data federation systems SMCQL and Conclave. Specifically, single query operations and compound query statements were compared to examine the application performance of the system comprehensively.

6.2.1 *Performance comparison of single query operation system*

The maximum number of data owners supported by SMCQL and Conclave systems is two and three, respectively, and with appropriate modifications, SMCQL can support single query operations on up to three data owners. Therefore, a performance comparison for the single query operations on SUM, AVG, MIN/MAX, equi-join, and θ-join is shown in Figure 6 in the case of three participants, where the left table size for the JOIN operation is 200,000 rows. The table size refers to the sum of the table sizes of all data owners involved in the JOIN operation.

The experimental comparison results show that the time consumed for the JOIN operation increases significantly with the increase in the size of the running data, while the time consumed for SUM, AVG, and MIN/MAX operations increases slightly. However, our implemented system can always maintain optimal performance. The time overhead is only 0.3% and 29.7% of that of SMCQL and Conclave, respectively.

6.2.2 *System performance comparison of compound query statements*

A compound query statement is a combination of single query operations. Due to the complexity and implementation issues of the SMCQL system, it is not easy to directly modify it to support multi-party scenarios with more than three data owners[13]. Therefore, this section

focuses on the comparison of the proposed system with Conclave. The compound query statements based on the TPC-H dataset are shown in Table 1 below, and the execution performance comparison between the two systems is shown in Figure 7.

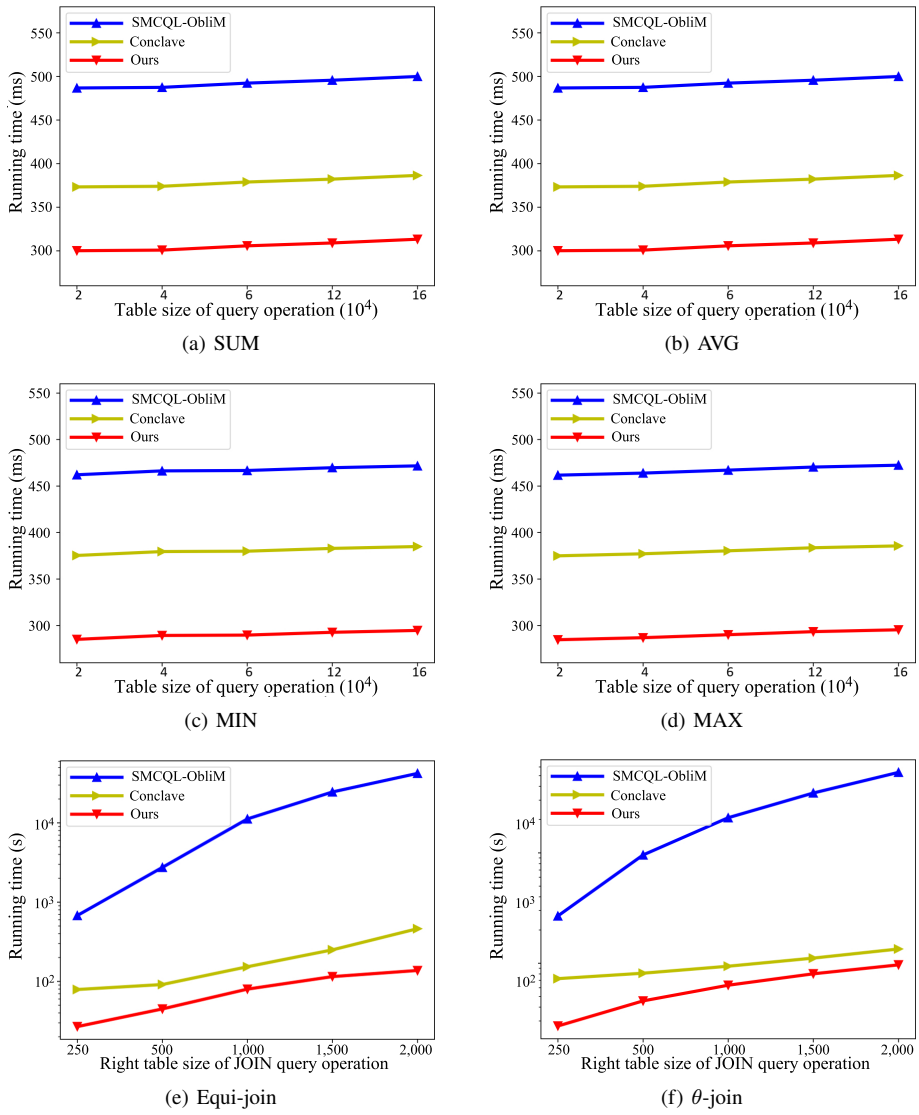


Figure 6 Comparison of single query operations amongproposed system, SMCQL, and Conclave

Table 1 SQLquery based on TPC-H dataset

SELECT	SUM(L_extended price) / 7.0 asavg_yearly
FROM	Lineitem, Part
WHERE	P_partkey = L_partkey
	and P_brand = 'Brand#55'
	and P_container = 'MED DRUM';

The experimental results show that the efficiency of the proposed system is significantly better than that of Conclave, with an efficiency advantage of up to 3.75 times.

In summary, the proposed system achieves the best performance among data federation systems that can support multi-party security. Furthermore, another advantage of this system is that it can support data query operations with more than three parties, which will be verified in the following experiments.

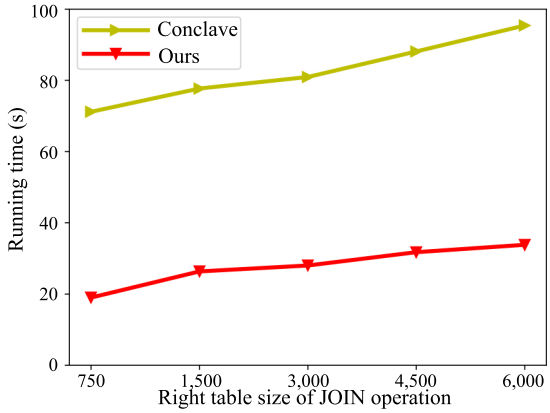


Figure 7 Comparison of compound statement query between our proposed system and MP-SPDZ (left table size is 200,000 rows)

6.3 Multi-party performance analysis

Both SMCQL and Conclave systems can no longer be used when there are more than three data owners. For this reason, the secure multi-party computation tool library MP-SPDZ was adopted[9]. As a comparison system, MP-SPDZ can achieve secure algebraic operations that support multiple parties and can be used as a comparison object for the basic operators of the proposed system. The performance of query operations of the proposed system in the multi-party case was further verified experimentally. The SUM and AVG operations are the simple superposition of basic operators, and thus we focus on the performance of equi-join and θ-join operations.

6.3.1 *Multi-party performance analysis on JOIN operations*

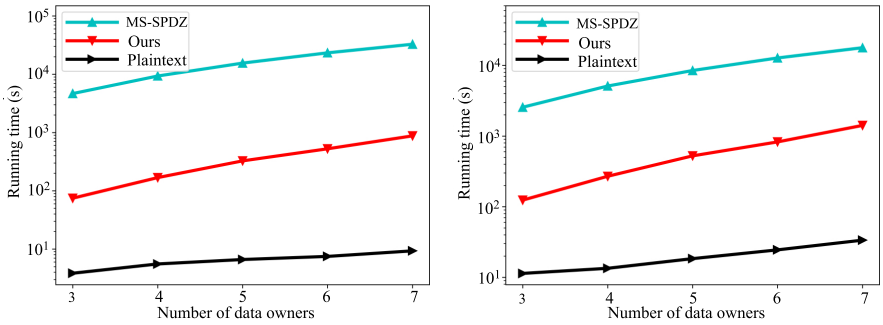
This section verifies experimentally the performance of the system's join operations in multi-party scenarios. The variation of time overhead for equi-join and θ-join operations of the proposed system with the increasing number of data owners is shown in Figure 8.

It can be seen that the proposed system remains relatively fast with a different number of data owners and can effectively protect data security with little difference in computational overhead from the direct plaintext.

6.3.2 *Multi-party performance analysis on basic operators*

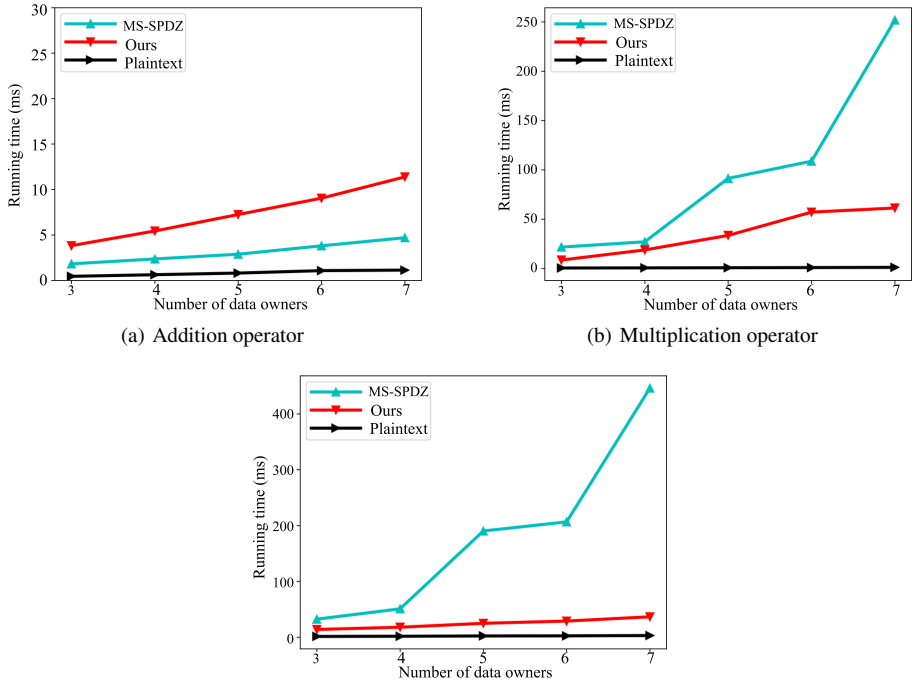
The performance comparison of basic operators of the proposed system for addition, multiplication, and comparison operations with MP-SPDZ and plaintext computation is shown in Figure 9. It is easy to convert between subtraction and addition and between division and multiplication, and thus the relevant comparison is omitted.

It can be seen that the time overhead increases as the number of data participants grows in the performance comparison of the three basic operators. For multiplication and comparison operators, the execution efficiency of the proposed system is always faster than that of MP-SPDZ



(a) Equi-join (b) θ-join

Figure 8 Comparison of two join query modes among proposed system, MP-SPDZ, and plaintext computation



(a) Addition operator (b) Multiplication operator

(c) Comparison operator

Figure 9 Comparison of operator efficiency among MP-SPDZ, plaintext, and proposed secure multi-party operator library

and is closer to that of plaintext computation. For the addition operator, the efficiency of the proposed system is slightly slower than MP-SPDZ, but the difference is small. Therefore, this system can fulfill the data security protection requirement with less time overhead.

In summary, the proposed system maintains better performance even when the number of data owners rises. This indicates that the system can not only outperform the existing SMCQL and Conclave systems in terms of execution efficiency but also significantly outstrip the existing systems in terms of scalability.

7 Related Work

The system implemented in this paper is a secure multi-party data federation system in a federation scenario. Unlike the federated database concept that emerged in the 1980s, such systems center on database heterogeneity and aim to achieve collaboration among heterogeneous database systems. In recent years, the concept of a secure multi-party database system in a federation scenario has emerged with the enhancement of user data security protection, which is also called a data federation system. Such systems not only support the collaboration of heterogeneous databases but also emphasize data security protection in multi-party collaboration, and they rely on secure multi-party computation technology to achieve data security protection. The federated database technology, secure multi-party data federation technology, and secure multi-party computation technology are described in this section.

7.1 Federated database technology

The concept of a federated database system[14] was introduced in the 1980s, which is not a physical database but a middleware between underlying multiple databases and upper users. As a virtual database running upon each database, it can screen the heterogeneity between underlying multiple databases and behave as a unified database for users.

Multiple heterogeneous databases at the bottom of the federation database system are independent of each other, but they also collaborate. Unlike traditional distributed databases, federated databases need to work together to store, query, and modify data. The underlying databases of the federation database system are autonomous to some extent, as they operate locally and independently. The application background of a federated database system is derived from the data management upon the acquisition of a company. For example, when Company A acquires Company B, the data of Company B needs to be managed by Company A, but this process may be hampered by the incompatibility of the two companies' databases. Federated database technology can solve this problem. By building a federated database management system and connecting Company B's database system to it, Company A only needs to process the data jointly with Company B through the federated database system rather than migrate all of Company B's data to its own database system. The DB2 data management system[15] introduced by IBM supports this federated database.

The concept of federation in such federated database technology implies the characteristics of multi-party collaboration and heterogeneous autonomy of all parties; however, it does not involve the data privacy and security issues in multi-party data, and thus it is considerably different from the federated scenarios and secure multi-party database concept discussed in this paper. The federation scenario and its related research are described in Section 7.2.

7.2 Secure multi-party data federation technology

With the increasing requirements for user data privacy protection worldwide, the traditional approach of sharing data directly without considering data security in a federation database is no longer applicable. Under this scenario, secure multi-party data federation technologies have emerged to meet the challenge of data security. Such systems enable multiple data owners to cooperate in query operations while ensuring that sensitive data involved in the computation are not leaked, thus protecting data security.

In 2017, Bater *et al*. proposed the SMCQL data federation system[5] that can support secure querying on two data owners' databases. The SMCQL system divides the access rights of each column of a data table into public and private ones. Different access rights correspond to different shared security requirements. For example, the values of public columns can be shared by default, and hence if the operation in the query only involves public column data, it

can be calculated directly in plaintext, the same way as the traditional database calculation. If the query involves private columns, it is necessary to ensure that the original private data does not leave the original data owner during the query operation. Thus, the security requirement of no data leakage is achieved. Although this system can guarantee the security of private data, it has a huge runtime overhead due to security protection and can only support two data owners. Its efficiency and scalability of the participant size limit the application of this system.

In 2019, Volgushev *et al*. developed the Conclave data federation system[6], which proposes query optimizations such as the push up, push down, and hybrid protocol to control the time overhead caused by the security protection. The core idea of the optimization is to complete the computation of query operations within the database of the owner as much as possible, so as to reduce data interactions between data owners and further reduce the time overhead caused by security protection. This system is more efficient than SMCQL, but it supports at most three data owners and has limited application scenarios.

At present, the idea of implementing secure operations in the data federation system is basically the same, i.e., converting SQL statements into security operation primitives and then executing them. Such security operation primitives are based on secure multi-party computation technology that can realize the joint computation of multiple parties without leaking sensitive data to each other, which complies with the security requirements of data federation. The related work on secure multi-party computation technology is described below.

7.3 Secure multi-party computation technology

Secure multi-party computation is a collection of cryptographic protocols proposed to solve the problem of secure computation in the development of modern cryptography. It solves the problem of joint computation of a function among a number of parties with mutual distrust, which was first introduced by the millionaire problem proposed by Andrew Chi-Chih Yao in 1982[11]. The focus in secure multi-party computation can be formalized as follows: n computation participants hold data $x_1, x_2, \cdots, x_n$, respectively, and the purpose of the protocol is to compute a pre-agreed function $y_1, y_2, \ldots, y_n = f(x_1, x_2, \cdots, x_n)$ with the secret data of each party, where any party can obtain the corresponding result y_i but cannot obtain any other information. Two common technologies, namely Garbled Circuit (GC)[16] and secret sharing[17] can solve this problem. The details of the two technologies are described as follows.

7.3.1 *Garbled circuit*

The GC technology can convert a computational function into a Boolean circuit and cryptographically garbles the truth table for computation, which is a general framework for solving secure computation problems between two parties. This technology can be applied to verifiable computation[18], Key-Dependent Message (KDM) security[19], etc. It can also be widely used in real life for medical diagnosis[20], auction mechanism[21], information retrieval[22], etc. to protect privacy. This model was first proposed by Turing Award winner Andrew Chi-Chih Yao in 1986 under the semi-honest model of Yao's circuit[23], which was used to solve the millionaire problem. The main work of Yao's circuit is to transform an arbitrary function into a Boolean circuit with two participants, namely Alice and Bob. Alice generates the truth table according to the logic circuit and then performs two-fold symmetric encryption and decryption operations for each circuit gate with the string corresponding to the cable as the key, and thus the garble decipher text table is generated; Bob calls the Oblivious Transfer (OT) protocol[24] to obtain the string form of the input and attempts to decrypt the garbled cipher text table line by line with this string and the string sent by Alice as the key. Recently, GC-based tool libraries ObliVM[3] and Obliv-C[4] were proposed by Chang Liu and Samee Zahur, respectively. Those tool libraries build a developer-oriented high-level programming language that compiles the template code

written by the developer directly into a GC, thus greatly reducing the development efforts. In fact, the aforementioned SMCQL system uses ObliVM to build GCs for secure computation, while Conclave partially uses Obliv-C. However, those two tool libraries with GCs as secure back ends perform slightly poorly in terms of computational efficiency; for example, the ObliVM tool library can only load less than 100 Kb of data input[13].

7.3.2 *Secret sharing*

Secret sharing is another important approach in secure multi-party computation, which is based on the idea of splitting a secret into multiple parts (sub-secrets) and then sending each part to the corresponding participant. In this way, only a subset of the authorized participant set can collaboratively reconstruct the original secret, while other arbitrary subsets of non-authorized participants cannot reconstruct the original secret. In addition to secure multi-party computation, secret sharing was originally used for secure information storage[25], key distribution[26], access control[27], and other applications in real life, such as electronic voting[28], copyright enforcement[29], and machine learning privacy protection[30]. Secret sharing was first introduced by Shamir[10] and Blakley[31], who proposed the threshold secret sharing, namely that any subset of the participant set with a size larger than a threshold can construct the original secret. The secret sharing computation process of the Shamir model mainly uses the Lagrange interpolation principle: firstly, a polynomial is randomly selected, and the input of the distributor is used as the constant term of the polynomial. Then, this polynomial is used to calculate the sub-secret distributed to each participant, and the original polynomial can be reconstructed when the size of the participant set is larger than a specified threshold by Lagrange interpolation. In this way, the original secret is obtained. There are also secure multi-party computation tool libraries based on secret sharing, such as Sharemind[8] and MP-SPDZ[9]. The data federation system Conclave uses Sharemind, but this tool library supports up to three computational parties and is not open-sourced, which restrains its use.

8 Summary and Outlook

In this paper, we designed and implemented a federated data federation system for multi-party security under the premise of increasingly strict data privacy protection and data sharing requirements. The system achieves the system adaptation, secure multi-party operator library, query engine, and interactive interface from the bottom up, which can screen underlying database heterogeneity, provide users with a unified and friendly operation interface, and realize data sharing under the premise of data privacy and security protection. Specifically, the work in this paper is summarized as follows.

- We implemented the secure multi-party operator library and query engine of this system. On the basis of the secret sharing framework, we achieved efficient and secure basic operators including addition, subtraction, multiplication, division, and comparison. On this basis, the query engine was built to support basic database query operations including SUM, AVG, MIN/MAX, equi-join, and θ-join, with full consideration of multi-party collaboration characteristics. The query engine implemented the idea of giving priority to local operations and reducing transmission redundancy in the whole process of query statement parsing, query plan generation, and query operation execution. This minimizes data interactions between data owners, thus greatly reducing security overhead and improving system performance.
- We implemented the system adaptation and the interactive interface. The system adaptation interface is designed to screen the database heterogeneity due to multiple data owners, while the interactive interface provides a unified SQL query interface and a graphical interactive interface for users to facilitate the unified coordination and management of

the data sharing process and improve usability.

- We verified the system performance experimentally. The experimental results on the benchmark dataset TPC-H show that, compared with the current data federation systems SMCQL and Conclave, the proposed system can support data federation scenarios with more than three data owners, and it shows higher efficiency in various basic query operations and compound queries. The efficiency and scalability of this system show that it has higher practicality than the current mainstream data federation systems.

The future work of this system is as follows.

- We will extend the scope of the system to support further query operations. The system is based on the idea of the basic secure multi-party computation of secret sharing, and it implements operators such as aggregation and join, but the functions supported by the system are limited compared with those of mature database systems. The system can be further expanded on the basis of the existing system on views and other operations.
- We will achieve more complex data mining analysis algorithms in view of existing basic operators. This system already supports simple query operations, but it can develop more complex data mining algorithms such as clustering and regression algorithms in light of existing SUM and product algorithms to further explore the value contained in multiple data while protecting privacy.

References

[1] Doan AH, Halevy A, Ives Z. Principles of Data Integration. Elsevier, 2012.

[2] Shi DY, Wang YS, Zheng PF, *et al*. Cross-silo federated learning-to-rank. Ruan Jian Xue Bao/Journal of Software, 2021, 32(3): 669–688 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/6174.htm. [doi: 10.13328/j.cnki.jos.006174]

[3] Liu C, Wang XS, Nayak K, *et al*. Oblivm: A programming framework for secure computation. Proc. of the 2015 IEEE Symposium on Security and Privacy. IEEE, 2015. 359–376.

[4] Zahur S, Evans D. Obliv-C: A language for extensible data-oblivious computation. IACR Cryptology ePrint Archive, 2015: 1153.

[5] Bater J, Elliott G, Eggen C, *et al*. SMCQL: Secure query processing for private data networks. Proc. of the 2017 VLDB Endowment. 2017, 10(6): 673–684.

[6] Volgushev N, Schwarzkopf M, Getchell B, *et al*. Conclave: Secure multi-party computation on big data. Proc. of the 14th EuroSys Conf. ACM,2019. 1–18.

[7] Hastings M, Hemenway B, Noble D, *et al*. Sok: General purpose compilers for secure multi-party computation. Proc. of the 2019 IEEE Symp. on Security and Privacy. IEEE, 2019. 1220–1237.

[8] Bogdanov D, Laur S, Willemson J. Sharemind: A framework for fast privacy-preserving computations. Proc. of the 2008 European Symp. on Research in Computer Security. Berlin, Heidelberg: Springer, 2008. 192–206.

[9] Keller M. MP-SPDZ: A versatile framework for multi-party computation. Proc. of the 2020 ACM SIGSAC Conf. on Computer and Communications Security. ACM, 2020. 1575–1590.

[10] Shamir A. How to share a secret. Communications of the ACM, 1979, 22(11): 612–613.

[11] Yao AC. Protocols for secure computations. Proc. of the 23rd Annual Symp. on Foundations of Computer Science. IEEE, 1982. 160–164.

[12] http://www.tpc.org/tpch/

[13] Wang Y, Yi K. Secure Yannakakis: Join-aggregate queries over private data. Proc. of the 2021 Int'l Conf. on Management of Data. ACM, 2021. 1969–1981.

[14] Sheth AP, Larson JA. Federated database systems for managing distributed, heterogeneous, and autonomous databases. ACM Computing Surveys, 1990, 22(3): 183–236.

[15] Josifovski V, Schwarz P, Haas L, *et al*. Garlic: A new flavor of federated query processing for DB2. Proc. of the 2002 ACM SIGMOD Int'l Conf. on Management of Data. ACM, 2002. 524–532.

[16] Bellare M, Hoang VT, Rogaway P. Foundations of garbled circuits. Proc. of the 2012 ACM Conf. on Computer and Communications Security. 2012. 784–796.

[17] Beimel A. Secret-sharing schemes: A survey. Proc. of the 2011 Int'l Conf. on Coding and Cryptology. Berlin, Heidelberg: Springer, 2011. 11–46.

[18] Setty S, Vu V, Panpalia N, *et al*. Taking proof-based verified computation a few steps closer to practicality. Proc. of the 21st USENIX Security Symposium. 2012. 253–268.

[19] Applebaum B. Key-dependent message security: Generic amplification and completeness. Proc. of the 2011 Annual Int'l Conf. on the Theory and Applications of Cryptographic Techniques. Berlin, Heidelberg: Springer, 2011. 527–546.

[20] Chen F, Cheng S, Mohammed N, *et al*. Precise: Privacy-preserving cloud-assisted quality improvement service in healthcare. Proc. of the 8th Int'l Conf. on Systems Biology. IEEE, 2014. 176–183.

[21] Kolesnikov V, Sadeghi AR, Schneider T. Improved garbled circuit building blocks and applications to auctions and computing minima. Proc. of the 2009 Int'l Conf. on Cryptology and Network Security. Berlin, Heidelberg: Springer, 2009. 1–20.

[22] Kim HJ, Kim HI, Chang JW. A privacy-preserving kNN classification algorithm using Yao's garbled circuit on cloud computing. Proc. of the 2017 IEEE 10th Int'l Conf. on Cloud Computing. IEEE, 2017. 766–769.

[23] Yao ACC. How to generate and exchange secrets. Proc. of the 27th Annual Symp. on Foundations of Computer Science. IEEE, 1986. 162–167.

[24] Kilian J. Founding crytpography on oblivious transfer. Proc. of the 20th Annual ACM Symp. on Theory of Computing. 1988. 20–31.

[25] Huang W, Langberg M, Kliewer J, *et al*. Communication efficient secret sharing. IEEE Trans. on Information Theory, 2016, 62(12): 7195–7206.

[26] D'Souza R, Jao D, Mironov I, *et al*. Publicly verifiable secret sharing for cloud-based key management. Proc. of the 2011 Int'l Conf. on Cryptology in India. Berlin, Heidelberg: Springer, 2011. 290–309.

[27] Naor M, Wool A. Access control and signatures via quorum secret sharing. IEEE Trans. on Parallel and Distributed Systems, 1998, 9(9): 909–922.

[28] Schoenmakers B. A simple publicly verifiable secret sharing scheme and its application to electronic voting. Proc. of the 1999 Annual Int'l Cryptology Conf. Berlin, Heidelberg: Springer, 1999. 148–164.

[29] Zhu Y, Yang YT, Sun ZW, *et al*. Ownership proofs of digital works based on secure multiparty computation. Ruan Jian Xue Bao/Journal of Software, 2006, 17(1): 157–166 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/17/157.htm. [doi: 10.1360/jos170157]

[30] Tan ZW, Zhang LF. Survey on privacy preserving techniques for machine learning. Ruan Jian Xue Bao/Journal of Software, 2020, 31(7): 2127–2156 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/6052.htm. [doi: 10.13328/j.cnki.jos.006052]

[31] Blakley GR. Safeguarding cryptographic keys. Proc. of the Int'l Workshop on Managing Requirements Knowledge. IEEE Computer Society, 1979. 313–313.

Shuyuan Li, Ph.D. candidate. Her research interests include big data analysis and processing, as well as privacy protection.

Dingyuan Shi, master's candidate. His research interests include federal learning, spatio-temporal big data analysis and processing, crowd sourcing computing, swarm intelligence, and privacy protection.

Yudian Ji, Ph.D., engineer. His research interests include spatio-temporal data analysis and processing, data compression, and data mining.

Wangdong Liao, master's candidate. His research interests include big data analysis and processing, as well as privacy protection.

Lipeng Zhang, master's candidate. His research interests include federated databases and privacy protection.

Ke Xu, professor, Ph.D. supervisor. His research interests include algorithms and artificial intelligence.

Yongxin Tong, Ph.D., professor, Ph.D. supervisor, senior member of CCF. His research interests include federal learning, spatio-temporal big data analysis and processing, crowd sourcing computing, swarm intelligence, and privacy protection.