Privacy-Preserving Queries Using Multisource Private Data Counting on Real Numbers in IoT

Guanglai Guo, Yan Zhu, Cecilia E Chen, Lejun Zhang, Rongquan Feng, Di Ma

***Abstract*—In this paper, our primary focus is on the current lack of privacy-preserving queries tailored to real-number fields rather than integers. We take multisource Private Data Counting on Real numbers (R-PDC) within IoT architecture as a breakthrough point to enable diverse query services without revealing sensitive data. The advantage of this is that the parameters required for queries remain stable and minimal in the scenario of wide numerical domain and dynamic changed dataset. At first, we present a general R-PDC method based on curve approximation, in which an Approximate Query Function (AQF) is established to approximate the constructed ideal counting curve for element queries on target set. We also demonstrate curve construction process of AQF and provide the feasibility theorem of AQF for counting a target element within an allowable error. By integrating the R-PDC method with fixed-point fully homomorphic encryption, an efficient R-PDC scheme is presented to perform multiparty collaborative queries in IoT. In security aspect, the R-PDC scheme on (m, ϵ)-AQF is proved to be statistically secure against Chosen Element Attack (CEA) for cumulative error ϵ and dataset size m. Moreover, the scheme achieves $O(nm\gamma)$ computation and $O(n^2 m\gamma)$ communication complexities for n servers and γ-length fraction. Finally, as an extension of AQF over single attribute, multi-dimensional R-PDC method is applied into privacy-preserving Naive Bayes Classifier and Apriori algorithm over multiple attributes. Our work provides substantial support and insights for the advancement of privacy computation.**

***Index Terms*—Privacy-preserving Query, Multisource Private Data Counting, Curve Approximation, Approximate Query Function, Fixed-point Presentation, IoT.**

I. Introduction

In an IoT network, multisource data queries/mining enable different institutions to collaboratively analyze their shared data from various IoT devices for valuable knowledge discovery [1]. For example, multiple healthcare institutions share medical records of patients monitored by wearable devices, so as to make better medical diagnoses [2]. In smart grid, multiple electricity sales departments share their power data, collected by edge servers, to provide optimal decision of balancing grid loads [3]. Thus, it plays an important role to provide better business decisions for institutions.

However, sharing sensitive data (e.g., personal interests and hobbies, health data and location info.) has been inevitably raising concerns on data privacy and legal compliance, such as HIPAA and GDPR laws [4] [5] . Undoubtedly, privacy-preserving data sharing and analysis are one of the key technologies to ensure data security in IoT applications. As their underlying technology, multisource Private Data Counting (PDC) allows all participants to integrate shared data from multiple sources to collaboratively perform counting without leaking their private data [6]. In IoT, PDC lays the security foundation on many applications, e.g., traffic flow statistics, health monitoring and smart retail analytics [7] [8]. Thus, it is essential and important to implement an efficient PDC for secure IoT applications.

A. Motivation

There have existed some PDC methods that are applied into privacy-preserving data sharing and analysis [9]. For example, Yang *et al.* [10] presented a privacy-preserving frequency mining protocol, which utilizes the additive homomorphism of Elgamal encryption to implement secure frequency statistics. Vo-Huu *et al.* [11] presented a frequency counting protocol called EPiC based on indicator polynomials, in which the counting problem is reduced to a summation of polynomial evaluations and somewhat homomorphic encryption is used to ensure data privacy.

As evident from existing literature, almost all known PDC methods are implemented on integers. However, the data that most practical applications need to handle are real numbers. For example, IoT sensors usually deal with temperature and humidity values in weather forecasting and disaster early warning, as well as length, weight and volume measurements in a safe and reliable service condition. Thus, there is an urgent need to find a practical PDC method on real numbers for various application needs.

A straightforward method for PDC on real numbers is to transform real numbers into integers through field conversion. It is commonly applied into fixed-point representation of real numbers, thereby the PDC methods on real numbers can be reduced to PDC on integers. Informally, the fixed-point representation of a real number $\tilde{a}$ is written as $\tilde{a} = \bar{a} \cdot 2^{-\gamma}$, where $\bar{a}$ is its integer representation and γ is its fraction length. Conversely, the integer representation of $\tilde{a}$ can be obtained by amplifying it, i.e., $\bar{a} = \tilde{a} \cdot 2^{\gamma}$. However, the multiplication on

G. Guo, Y. Zhu, and C. E. Chen are with the School of Computer and Communication Engineering, University of Science and Technology Beijing, Beijing, 100083 China (e-mail: guanglai_guo@126.com; zhuyan@ustb.edu.cn; chene@ustb.edu.cn).

L. Zhang is with Cyberspace Institute Advanced Technology, Guangzhou University, Guangzhou 510006, China (e-mail: zhanglejun@gzhu.edu.cn).

R. Feng is with the School of Mathematical Sciences, Peking University, Beijing 100871, China (e-mail: fengrq@math.pku.edu.cn).

D. Ma is with Computer and Information Science Department, College of Engineering and Computer Science, University of Michigan-Dearborn, Michigan 48128, USA (dmadma@umich.edu).

Corresponding author: Yan Zhu

real numbers will cause Position Drift of Decimal Separator (PDDS). For example, for given two inputs $3.25 = 13 \times 2^{-2}$ and $1.5 = 6 \times 2^{-2}$ with $\gamma = 2$, the multiplication output is $3.25 \times 1.5 = (000011.01)_2 \times (000001.10)_2 = (0100.1110)_2 = 4.875$, where $(\cdot)_2$ denotes binary representation. This means that the fraction length of the multiplication result is changed as $\gamma = 4$. So, the position of decimal point is left shifted by 2 bits in comparison with two inputs. In computer system, two least significant bits ($\gamma = 2$) of the multiplication result might be truncated to keep the decimal point position unchanged, i.e., $3.25 \times 1.5 = (0100.1110)_2 \approx (0100.11)_2 = 19 \times 2^{-2} = 4.75$, and the output's error is 0.125. Obviously, algebraic operation on real numbers can be implemented on integers through field conversion, but this will introduce the fixed-point representation errors of real numbers due to PDDS.

Basic algebraic operations, such as addition, multiplication and exponentiation, commonly need to be performed in a given PDC computing task. To do it, Homomorphic Encryption (HE) is currently the most common form to perform these algebraic operations on encrypted data, which is collectively referred to as Ciphertext-State Computation (CSC). In other words, CSC is a process of performing computations or operations directly on data in a ciphertext form. At present, the CSC researches mainly focus on algebraic operations on integers, but CSC on real numbers is relatively few and mainly focuses on Secure Multiparty Fixed-point Computation (SMFC) protocols. A typical SMFC scheme, presented by Catrina *et al.* [12], can implement addition and multiplication HE on real numbers and solve the PDDS problem. To our best knowledge, it is currently one of the most complete frameworks of SMFC in CSC on real numbers.

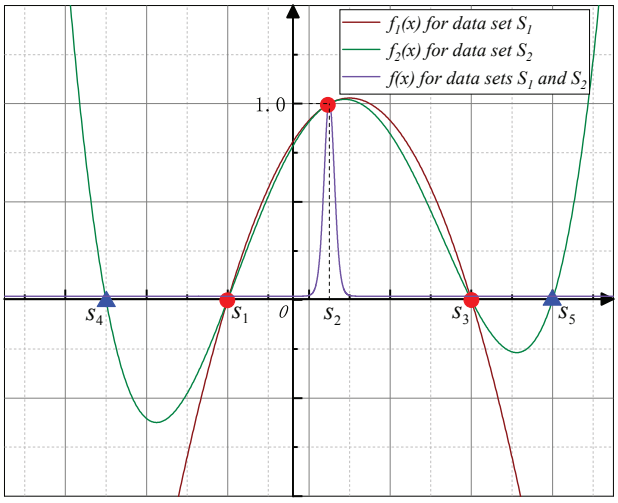


Fig. 1. Three curves for counting the same element on different data sets.

The implementation method of PDC on integers is generally based on polynomial interpolation. Formally, for given data set $S = \{s_1, s_2, \cdots, s_m\}$ and target element $w \in S$, the idea of this method is to construct an interpolating polynomial $f(x)$ on domain S such that $f(x) = 1$, when $x = w$; and $f(x) = 0$, otherwise, then scan through all elements in S to compute the sum $\sum_{i=1}^{m} f(s_i)$. The result is the number of element w in S. Based on this method, Fig.1 shows the curves of interpolating polynomials $f_1(x)$ and $f_2(x)$ for counting the target element $w = s_2$ on data sets $S_1 = \{s_1, s_2, s_2, s_3\}$ and $S_2 = \{s_1, s_2, s_2, s_3, s_4, s_4, s_5\}$, respectively. It is easy to see that the curve shapes of $f_1(x)$ and $f_2(x)$ will be different if the elements in domains S_1 and S_2 are different.

In this method, adding or removing an element to/from the data set will significantly change the curve shape of the interpolation polynomial, thereby requiring the reconstruction of it. That is, this method has poor stability and is not suitable for the data set with dynamic changes. For example, the temperature data set exhibits significant variations corresponding to different seasons. In this setting, the temperature difference in a certain area ranges from $-20\,^{\circ}\mathrm{C}$ to $40\,^{\circ}\mathrm{C}$ among four seasons. When the measurement accuracy is 0.1, a data set containing 601 elements will be generated. Thus, an interpolating polynomial of degree 601 is required to be constructed for counting on the data set. As a result, this construction form of polynomial curve suffers from excessive storage overheads and increases the complexity of application development. It means that this method is only applicable to a fixed discrete set of integers. As shown in Fig.1 (see the purple curve $f(x)$), there will be a simpler solution for the above example by using PDC on real numbers, described in the next section.

B. Our Approaches

In views of limitations of PDC on integers, we intend to present a practical PDC method on real numbers based on curve approximation (called R-PDC). This method can implement efficient counting on real numbers with an allowable error, while ensuring the privacy of multisource data. Exactly, we employ the following techniques to implement it.

- In curve construction aspect, an approximation algorithm is used to construct the curve of Approximate Query Function (AQF) for counting target elements. This kind of curve construction has a unified form to support queries for variable continuous data sets (e.g., the purple curve $f(x)$ in Fig.1 supports queries on both S_1 and S_2, simultaneously). Moreover, the parameter of AQF is adjusted to control the cumulative counting error within an allowable range for the correctness of counting results.
- In ciphertext-state computation of AQF aspect, polynomial fitting method is used to convert AQF into a polynomial with real coefficients for homomorphic computations of addition and multiplication on real numbers. By using SMFC, the coefficients of polynomial and elements of data set are distributed into all participants in the form of fixed-point representation, so as to ensure the privacy of multisource data in collaborative query process.

The curve $f(x)$ of our R-PDC method in Fig.1 is a long tail curve such that $f(x) \approx 1$, when $x = w$; and $f(x) \approx 0$, otherwise. Using this curve for queries will incur counting errors. As shown in Fig.1, the curve $f(x)$ is symmetric about the line $x = s_2$ and decreases monotonically on intervals $(s_2, +\infty)$ and $(-\infty, s_2)$, resulting in a monotonically decreasing counting error. This feature of curves ensures that the cumulative counting error is controllable on all elements of the

data set. As a result, the curve $f(x)$ is applicable to efficient queries on large-scale data sets with dynamic change.

The SMFC framework presented by Catrina *et al.* [12] is adopted to implement homomorphic computations of addition and multiplication on real numbers. Especially, the $TruncPr$ algorithm in the framework effectively solves the problem of fixed-point representation errors on real numbers. Based on SMFC, a Fixed-point Fully Homomorphic Encryption (F-FHE) scheme is constructed to implement multiparty CSC with fixed-point representation[1]. Therefore, the proposed R-PDC method can support queries not only on integers but also on real numbers, thereby meeting the business needs of most applications.

C. Our Contributions

In this paper, we address the implementation of privacy-preserving queries based on a practical and efficient R-PDC method in real-number field. To do it, we provide a method of fixed-point representation on real numbers and design a system model over IoT to implement privacy-preserving query process among three entities, i.e., owners, inquirer and multiservers, by four algorithms (Encap, Query, Answer and Reduct). On this basis, an efficient R-PDC scheme over F-FHE is presented to protect multisource data privacy in collaborative queries of multiservers. Our main contributions are as follows:

- We present a general R-PDC method based on curve approximation. In this method, an ideal counting curve is constructed from queried target set, and then an AQF is generated to approximate this curve for element counting. Especially, we demonstrate curve construction process of AQF and provide the feasibility theorem of AQF for counting a target element within an allowable error. Moreover, the constructed AQF can be converted into a fitted polynomial of variable degree to implement CSC with fixed-point representation. As an extension of AQF over single attribute, n-dimensional R-PDC method is applied into privacy-preserving Naive Bayes Classifier and Apriori algorithm over multiple attributes.
- We provide a type of curve, expressed as $f(x) = ((x - w)^2 + 1)^{-a}$, to establish (m, ϵ)-AQF. This construction can count target w for any data set S with m elements provided that $a > 3.3(\log m - \log \epsilon)$, where $\epsilon \in (0, 0.5]$ is the upper bound of allowable cumulative counting error. In security aspect, the R-PDC scheme on (m, ϵ)-AQF is proved to be statistically secure against Chosen Element Attack (CEA) for both data privacy and query privacy. Moreover, the presented scheme achieves $O(nm\gamma)$ computation and $O(n^2 m\gamma)$ communication complexities, where n is the number of servers in multiservers and γ is the fraction length of real numbers.

In the remainder of this paper, we introduce the current state of the art in Section II and basic preliminaries in Section III, followed by the definitions of private data counting and system model in Section IV. We present a general R-PDC method based on curve approximation in Section V and construct the R-PDC scheme over F-FHE in Section VI. The full security proofs and performance evaluations of our scheme are given, respectively, in Section VII and Section VIII. In Section IX, we show two applications of the multi-dimensional R-PDC method. Finally, we conclude this paper in Section X.

[1]The F-FHE construction is described in the supplementary material

II. Current State of the Art

In this section, we will introduce the current state of the art on secure computation with fixed-point numbers and the PDC methods, respectively.

A. Secure Computation on Real Numbers

In the past decade, many research works focused on secure computation with fixed-point representation since some applications required computing over real numbers. Catrina and Saxena [13] presented the first secure computation framework with real numbers using fixed-point representation, which provides some protocols for secure fixed-point arithmetic and comparison based on secret sharing. Subsequently, Catrina [14] presented some improved building blocks and protocols for secure fixed-point arithmetic based on the framework in literature [13]. After that, Liedel [15] presented a secure square root protocol to extend the fixed-point arithmetic primitive for secure computation in the framework of Catrina and Saxena. Recently, Ugwuoke *et al.* [16] presented an efficient two-party fixed-point division protocol to compute fixed-point quotient of two encrypted inputs with a linear computational complexity. Boyle *et al.* [17] presented the first function secret sharing gates for arithmetic and logical shift based on pseudorandom generator, which enables secure computation of fixed-point arithmetic, including multiplication and comparison.

In some literature, the framework of Catrina and Saxena [13] has been applied into privacy-preserving multisource data mining protocols. At first, Catrina and De Hoogh [12] presented secure multiparty linear programming protocols with fixed-point arithmetic to achieve privacy-preserving collaborative optimization problems, as well as applications in secure supply chain management [18]. After that, De Cock *et al.* [19] presented a secure protocol for performing linear regression over distributed datasets based on pre-distributed data. Moreover, the presented protocol is proved to be information-theoretically secure in the commodity-based model. Similarly, Gascón *et al.* [20] presented privacy-preserving distributed linear regression protocols over vertically-partitioned datasets, which provided security against semi-honest adversaries.

Other secure computation frameworks with fixed-point numbers have been presented in recent years. Mohassel and Zhang [21] presented SecureML, a secure two-party computation framework using secret sharing in the two-server model, aiming to provide efficient protocols for private preserving machine learning such as linear regression. SecureML supported build-in fixed-point multiplication with precomputed Beaver multiplication triples. Mohassel and Rindal [22] presented a complete three-party computation framework (ABY^3) for training linear regression in three-server model. Moreover, ABY^3 used replicated secret sharing technique of Araki *et al.*

[23] to achieve fixed-point multiplication. Li and Xu [4] presented PrivPy, an efficient framework for privacy-preserving collaborative data mining in the four-server model. the PrivPy computation engine combined the thought of SecureML and ABY^3 and provided more efficient protocols for fixed-point multiplication using 2-out-of-4 secret sharing.

Some scholars extended homomorphic encryption schemes by encoding methods to enable secure computations with real numbers. Fouque *et al.* [24] improved the Paillier cryptosystem to achieve its homomorphic properties on bounded rationals by encoding it as a fraction of two integers and decoding it from two-dimensional lattices. Jiang *et al.* [25] used the scaled numeric formats to tranform real number into integer and then encoded them into polynomial, so that homomorphic operations with encoded fixed-point real number can be performed in multi-bit fully homomorphic encryption. Dowlin *et al.* [26] introduced a binary fractional encoder for fixed-precision rational numbers, in which the digits after the decimal point was encoded as the high-degree coefficients of the polynomial.

B. Private Data Counting

Some secure protocols using encryption are designed for private data counting in data mining. Yang *et al.* [10] presented the privacy-preserving frequency mining protocols that allow a data miner to privately compute frequencies of values in the customers' data. Their protocol design was based on the additive homomorphism of a variant of ElGamal encryption [27]. Roughan and Zhang [28] used the secure sum protocol [29] to achieve secure distributed internet traffic measurement such as calculating the number of traffic flows across a set of networks. Vo-Huu *et al.* [11] presented EPiC, a practical frequency counting protocol based on indicator polynomials. EPiC transformed the problem of privacy-preserving pattern counting into a summation of polynomial evaluations and used somewhat homomorphic encryption [30] to address secure counting in a highly efficient manner.

Other solutions based on secret sharing paradigm have been also applied into private data counting. Emekci *et al.* [31] presented a privacy-preserving summation protocol based on addictive secret sharing [32], and it is applied into privacy-preserving decision tree algorithm for counting instances with specific attribute values. Bohli *et al.* [33] presented $equal^+$, an equality comparison protocol based on joint random number sharing [34] in assisting server model. Based on $equal^+$, they also presented a link-counting application protocol to sum up the number of matched records with specific conditions in cooperative network monitoring. Dolev *et al.* [35] presented an oblivious count query protocol for MapReduce based on Shamir's secret sharing scheme. The protocol was actually an extension of string-matching algorithm [36] that was executed using accumulating automata.

III. Preliminary

In this section, we recall some preliminaries. At first, some notations are defined as follows. Let $P = \{P_1, \cdots, P_n\}$ be the set of n participants. The vector and matrix are denoted as bold lower-case and capital letters, e.g., $\mathbf{x}$ and $\mathbf{X}$, respectively. $\mathbf{C}_x = (x_1, \cdots, x_n)$ is denoted as the ciphertext vector with n sub-ciphers x_k for the corresponding plaintext x, where $k \in [1, n]$. $[\mathbf{C}_x]$ is denoted as the aggregation of shared sub-ciphers from n participants. In addition, the other notations used in this paper are listed in Table I.

TABLE I
THE DEFINITION OF MAIN NOTATIONS.

Notations	Description
$\mathbb{F}_p$	the finite field of order p
$\mathbb{Z}_p$	the integer field of order p
$\mathbb{U}$	the finite set of integers
$\mathbb{P}$	the finite set of real numbers
γ	the fraction part length of fixed-point number
e	the integer part length of fixed-point number
S	the data set $\{s_1, s_2, \cdots, s_m\}$
W	the target set $\{w_1, w_2, \cdots, w_\tau\}$
$\mathbf{D}$	the encrypted data of S
$\mathbf{\Phi}$	the query parameter
$\boldsymbol{\alpha}$	the query response
β	the counting result

A. Linear Secret Sharing

Linear Secret Sharing Schemes (LSSS) are a useful tool in construction of cryptographic security protocols. Generally, the LSSS scheme involves a dealer who owns a secret r, and a set of n participants holding the shares of r. For the (t, n)-threshold case [32], the dealer divides the secret r into n shares and distributes them among n participants, with the properties that any t or more shares can easily reveal r, but any $t-1$ or less shares cannot reveal any information on r. Formally, the LSSS scheme is defined as follows.

Definition 1 (Linear Secret Sharing Schemes, LSSS): Let $\mathbb{F}_p$ be the finite filed of prime order p. Let $x_1, x_2, \cdots, x_n$ be n distinct nonzero elements in $\mathbb{F}_p$. The linear secret sharing scheme includes the following two phases:

- *Secret-sharing phase*: the dealer, who owns a secret $r \in \mathbb{F}_p$, randomly selects a polynomial $f(x) = r + \sum_{i=1}^{t-1} a_i x^i$, where $t-1$ elements $a_1, a_2, \cdots, a_{t-1} \in \mathbb{F}_p$. The i-th share r_i of the secret r is $r_i = f(x_i) \in \mathbb{F}_p$ for $i \in [1, n]$. Then, the share r_i is sent to participant P_i, privately.
- *Secret-reconstruct phase*: k participants $P_1, P_2, \cdots, P_k$ can recover the secret r from their shares $r_1, r_2, \cdots, r_k$ by Lagrange interpolation, i.e., $r = \sum_{i=1}^{k} L(i) \cdot r_i$, where $k \geq t$ and $L(i) = \prod_{j \in [1,k], j \neq i} \frac{x_j}{x_j - x_i} \in \mathbb{F}_p$ is the Lagrange coefficient.

B. Homomorphic Encryption

Homomorphic Encryption (HE) is a key technique of cryptography that allows to perform either addition or multiplication operation on ciphertext without decryption. Generally, the HE scheme has four algorithms, including $\mathbf{KGen}$, $\mathbf{Enc}$, $\mathbf{Dec}$ and $\mathbf{Eval}$, where $\mathbf{Eval}$ is an additional algorithm that realizes homomorphic evaluations on ciphertexts. Next, the definition of HE is described as follows.

Definition 2 (Homomorphic Encryption, HE): A homomorphic public key encryption scheme consists of four algorithms $(KGen, Enc, Dec, Eval)$ such that

TABLE II
THE MAPPING RELATIONSHIP OF ELEMENTS AMONG THREE DOMAINS.

Domain ($l=4,\gamma=2,p=17$)	Data Range																
$\mathbb{P}=\{-7\times 2^{-2}\le \bar{x}\le 7\times 2^{-2}\}$	$-\frac{7}{4}$	$-\frac{3}{2}$	$-\frac{5}{4}$	-1	$-\frac{3}{4}$	$-\frac{1}{2}$	$-\frac{1}{4}$	0	$\frac{1}{4}$	$\frac{1}{2}$	$\frac{3}{4}$	1	$\frac{5}{4}$	$\frac{3}{2}$	$\frac{7}{4}$		
$\mathbb{U}=\{-7\le \bar{x}\le 7\}$	-7	-6	-5	-4	-3	-2	-1	0	1	2	3	4	5	6	7		
$\mathbb{Z}_p=\{-7\le \bar{x}\le 9\}$	-7	-6	-5	-4	-3	-2	-1	0	1	2	3	4	5	6	7	8	9

- $\mathbf{KGen}(1^\kappa)\rightarrow(pk,sk)$, takes as input the security parameter κ, and it outputs a public key pk and a private key sk.
- $\mathbf{Enc}(pk,x)\rightarrow C_x$, takes as inputs the public key pk and a message x, and it outputs a ciphertext C_x.
- $\mathbf{Dec}(sk,C_x)\rightarrow x$, takes as inputs the private key sk and a ciphertext C_x, and it outputs the message x.
- $\mathbf{Eval}(C_x,opt,C_y)\rightarrow C_z$, takes as inputs the addition or multiplication operation $opt\in\{+,*\}$, and two ciphertexts, C_x and C_y, and it outputs a new ciphertext C_z.

According to [37], the HE scheme in Definition 2 is said to be fully homomorphic if **Eval** enables both homomorphic addition and multiplication operations. The definition of fully homomorphic encryption is given as follows.

Definition 3 (Fully Homomorphic Encryption, FHE): A HE scheme $(KGen,Enc,Dec,Eval)$ is said to be fully homomorphic, if it is homomorphic for both addition and multiplication operations *opt*, where $opt\in\{+,*\}$.

IV. DEFINITION AND SYSTEM MODEL

A. Fixed-point Representation

Fixed-point numbers are real numbers with fixed position of decimal point, and divide real numbers into an integer part and a fractional part with fixed length. For a l-bit fixed-point number, e is denoted as the length of integer part (including the sign bit), γ is denoted as the length of fractional part, and $l=e+\gamma$. Formally, the fixed-point representation of a number is defined as follows.

Definition 4 (Fixed-point Representation): Given a set of l-bit integers $\mathbb{U}=\{x\in\mathbb{Z}|-2^{l-1}+1\le x\le 2^{l-1}-1\}$, a l-bit fixed-point representation is the subset $\mathbb{P}$ of real numbers, defined as $\mathbb{P}=\{\tilde{x}=\bar{x}\cdot 2^{-\gamma}|\bar{x}\in\mathbb{U}\}$. Accordingly, the range of $\mathbb{P}$ is the set $\{-2^{e-1}+2^{-\gamma}\le\tilde{x}\le 2^{e-1}-2^{-\gamma}\}$, where γ is the length of the fractional part, and $e=l-\gamma$ is the length of the integer part.

Example 1: Let $l=8$ and $\gamma=2$. From Definition 4, the fixed-point representation of 3.25, 1.5 and 5 are listed as:

1) $\mathbf{3.25}$: $\tilde{x}=3.25=(000011.01)_2=(00001101)_2\times 2^{-2}=13\times 2^{-2}$, where $\tilde{x}=13$.
2) $\mathbf{1.5}$: $\tilde{x}=1.5=(000001.10)_2=(00000110)_2\times 2^{-2}=6\times 2^{-2}$, where $\tilde{x}=6$.
3) $\mathbf{5}$: $\tilde{x}=5=(000101.00)_2=(00010100)_2\times 2^{-2}=20\times 2^{-2}$, where $\tilde{x}=20$.

In the above definition of fixed-point representation, a fixed-point number $\tilde{x}\in\mathbb{P}$ can be encoded in an integer $\bar{x}\in\mathbb{U}$. For this reason, we define the mapping

$$int_\gamma:\mathbb{P}\rightarrow\mathbb{U}, int_\gamma(\tilde{x})=\tilde{x}\cdot 2^\gamma, \tag{1}$$

to map a fixed-point number $\tilde{x}$ in $\mathbb{P}$ to an integer $\bar{x}$ in $\mathbb{U}$.

In addition, all secure computations are limited to the integer field $\mathbb{Z}_p$. So, we use the function

$$fld_p:\mathbb{U}\rightarrow\mathbb{Z}_p, fld_p(\bar{x})=\bar{x}\mod p, \tag{2}$$

to encode an integer $\bar{x}$ in $\mathbb{U}$ as a single field element x in $\mathbb{Z}_p$ by $x=\bar{x}\mod p$, where $p>2^l$.

Actually, in order to achieve secure multiparty computation with fixed-point numbers, the fixed-point representation establishes the mapping among three domains, i.e., $\mathbb{P}\rightarrow\mathbb{U}\rightarrow\mathbb{Z}_p$. Exactly, a fixed-point number $\tilde{x}\in\mathbb{P}$ is firstly mapped to an integer $\bar{x}=\tilde{x}\cdot 2^\gamma\in\mathbb{U}$, and then encoded into a field element $x=\bar{x}\mod p\in\mathbb{Z}_p$.

Example 2: Let $l=4$ and $\gamma=2$. According to Definition 4, the range of 4-bit integer set is $\mathbb{U}=\{-7\le\bar{x}\le 7\}$, and the range of 4-bit fixed-point representation set is $\mathbb{P}=\{-7\times 2^{-2}\le\bar{x}\le 7\times 2^{-2}\}$. Assume that $p=17>2^4$. the range of integer filed is $\mathbb{Z}_p=\{-7\le\bar{x}\le 9\}$. For clarity, Table II lists the mapping relationship of elements among these three domains.

From Table II, it can be seen that the mapping $\mathbb{P}\rightarrow\mathbb{U}$ is one-to-one mapping. Similarly, the mapping $\mathbb{U}\rightarrow\mathbb{Z}_p$ is one-to-one mapping. Moreover, under the condition of $p>2^l$, there exist some additional elements (e.g., 8, 9) in $\mathbb{Z}_p$ that can not be represented by fixed-point representation, but this will not affect the fixed-point computation. So, the fixed-point numbers can be effectively encoded as field elements.

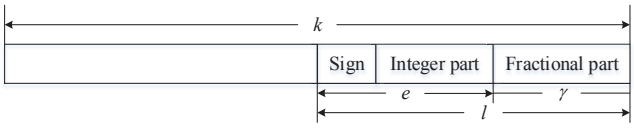


Fig. 2. The length relation between fixed-point numbers and storage space.

In practical algebraic operations, there exists data overflow in multiplication of fixed-point numbers. In order to ensure the accuracy of computation with fixed-point numbers, fixed-point representation needs to sacrifice half of the storage space. For clarity, Fig.2 shows the relationship between the length l of fixed-point numbers and the size k of storage space. Exactly, the length l of fixed-point numbers should be limited to not more than half of size k of storage space, i.e., $l\le k/2$.

B. Private Counting Problem

We firstly define Data Counting (DC) problem. Informally, counting is an action of determining the total number of data objects. Formally, the definition of DC is as follows.

Definition 5 (Data Counting, DC): Given the data set $S = \{s_1, s_2, \cdots, s_m\}$, where all $s_i \in \mathbb{Z}$ for $i \in [1, m]$, data counting is an approach for obtaining the number of one or more certain elements in S.

For clarity, the set W of τ elements to be counted, denoted by $W = \{w_1, w_2, \cdots, w_\tau\}$, is called as the target set, where all $w_i \in \mathbb{Z}$ for $i \in [1, \tau]$. Formally, given S and W, we use the function $Count : \mathbb{Z}^m \times \mathbb{Z}^\tau \to \mathbb{Z}$ to define the DC problem as $Count(S, W) = |\{\forall s_i, s_i \in W, i \in [1, m]\}|$, where $|\cdot|$ denotes the length of any input set.

In cryptography, Private Data Counting (PDC) is a protocol that allows a user for data counting from a server in possession of a data set, without revealing the information of stored data and counting result. Without loss of generality, the definition of PDC is given as follows.

Definition 6 (Private Data Counting): Given the DC problem $Count(S, W)$, a DC scheme is called private data counting between an inquirer and a server if the following properties hold:

- **Correctness**: the counting result obtained by the inquirer is consistent with that of $Count(S, W)$;
- **Query Privacy**: the query information W is kept secret for the server, and no information except for the counting result is available for the inquirer;
- **Data Privacy**: there does not exist the stored data for revealing to the server in the counting process.

C. Multiparty Fixed-point Fully Homomorphic Encryption

We intend to reduce PDC on real numbers (called R-PDC) into addition and multiplication operations with fixed-point numbers. So, we introduce the definition of multiparty Fixed-point Fully Homomorphic Encryption (F-FHE), as follows.

Definition 7 (F-FHE): A multiparty fixed-point fully homomorphic encryption scheme consists of four algorithms $(ParamGen, Enc_{pp}, Dec_{pp}, Eval_{pp})$ such that

- $\mathbf{ParamGen}(1^\kappa, n) \to pp$, takes as inputs the security parameter κ and the number n of participants, and outputs the public parameter pp.
- $\mathbf{Enc_{pp}}(\tilde{x}) \to \mathbf{C}_{\tilde{x}}$, takes as inputs the public parameter pp and a plaintext $\tilde{x} \in \mathbb{P}$, and outputs a ciphertext vector $\mathbf{C}_{\tilde{x}} \in \mathbb{Z}_p^n$, where $\mathbf{C}_{\tilde{x}} = (c_{x,1}, \cdots, c_{x,n})$ and $c_{x,k}$ is the sub-cipher for the k-th participant for $\tilde{x}$.
- $\mathbf{Dec_{pp}}([\mathbf{C}_{\tilde{x}}]) \to \tilde{x}$, takes as inputs the public parameter pp and a ciphertext $[\mathbf{C}_{\tilde{x}}] \in \mathbb{Z}_p^n$, and outputs a plaintext $\tilde{x} \in \mathbb{P}$, where $[\mathbf{C}_{\tilde{x}}]$ represents the aggregation of shared sub-ciphers from the n participants.
- $\mathbf{Eval_{pp}}(\mathbf{C}_{\tilde{x}}, opt, \mathbf{C}_{\tilde{y}}) \to \mathbf{C}_{\tilde{z}}$, takes as inputs the public parameter pp, the operation $opt \in \{+, *\}$, and two ciphertexts vectors, $\mathbf{C}_{\tilde{x}}$ and $\mathbf{C}_{\tilde{y}}$, and outputs a new ciphertext vector $\mathbf{C}_{\tilde{z}}$.

In the definition of F-FHE, threshold cryptosystem is introduced into the multiparty setting, thus the threshold is used to replace the key in traditional cryptosystem. According to the FHE scheme on integers [37] and the secure computation method with fixed-point numbers [12], a candidate F-FHE cryptosystem is given in the supplementary material and applied into the implementation of our R-PDC scheme.

D. System Model over IoT Architecture

IoT architecture, including perception, network and application layers, can be applied to implement R-PDC. In Fig.3, various sensors upload their collected data into multiple servers at network layer. And then, multiple servers adopt the above mentioned F-FHE cryptosystem to ensure the security of data counting process. Finally, they can provide query services for inquirers in application layer. Especially, our model consists of the following three entities.

- **Owners** ($\mathcal{O}$): refers to multiple distinct data entities, e.g., sensors, wearable devices, or IoT devices;
- **Inquirer** ($\mathcal{Q}$): is an individual who requests count queries of designated elements from multiservers, receives query responses from multiservers, and recovers the final counting results;
- **MultiServers** ($\mathcal{MS}$): represents the n servers $P = \{P_1, P_2, \cdots, P_n\}$, that jointly perform secure multiparty computation on encrypted data from multiple sources, faithfully execute the query requests and send the query responses to the inquirer.

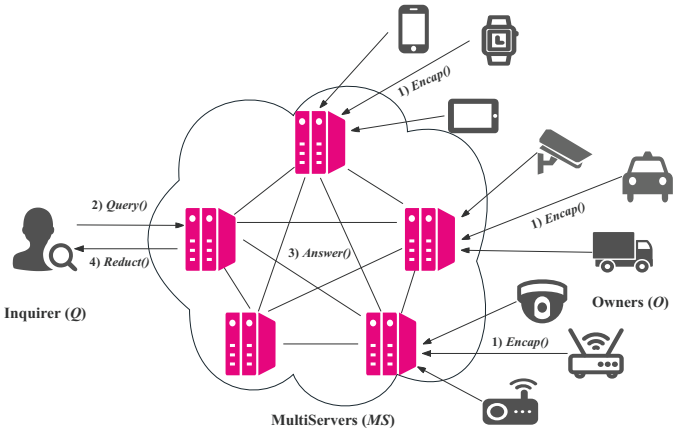


Fig. 3. The R-PDC model over IoT Architecture.

Note that, in application, each of multiservers may be the agent of data entities that distributes their own data to other servers, thereby forming a union data set $S = \{S_i | 1 \le i \le d\}$, where S_i is a set of data from the i-th data entity.

In the above model, the R-PDC is decomposed into four steps. As shown in Fig.3, each of these steps is represented as an algorithm. Accordingly, the R-PDC scheme over F-FHE, that consists of four algorithms, is defined as follows.

Definition 8 (R-PDC): Given the public parameter pp under security parameter κ, a data set S and a target set W, a PDC scheme based on F-FHE consists of four algorithms $(Encap, Query, Answer, Reduct)$ such that

- $\mathbf{Encap}(S)$, denotes an encryption algorithm invoked by the owners $\mathcal{O}$ that takes the data set S as input, and outputs the encrypted data set $\mathbf{D}$. Then, $\mathbf{D}$ will be stored on the multservers $\mathcal{MS}$, i.e., $Encap(S) = \mathbf{D}$;
- $\mathbf{Query}(W)$, denotes a query algorithm executed by the inquirer $\mathcal{Q}$ that takes the target set W as input, and outputs a query parameter $\mathbf{\Phi}$, i.e., $Query(W) = \mathbf{\Phi}$;

- $\mathbf{Answer}(\mathbf{D}, \mathbf{\Phi})$, denotes a R-PDC protocol run by the multiservers $\mathcal{MS}$ that take the data set $\mathbf{D}$ and the query parameter $\mathbf{\Phi}$ as inputs, and outputs the query response $\boldsymbol{\alpha}$, i.e., $Answer(\mathbf{D}, \mathbf{\Phi}) = \boldsymbol{\alpha}$;
- $\mathbf{Reduct}(\boldsymbol{\alpha})$, denotes a reduction algorithm executed by the inquirer $\mathcal{Q}$ that takes the query response $\boldsymbol{\alpha}$ as input, and outputs counting result β, i.e., $Reduct(\boldsymbol{\alpha}) = \beta$.

V. The R-PDC Method

In this section, we present a general R-PDC method based on curve approximation, which can implement data counting on real numbers with an allowable error. In particular, an approximate query function is formally defined and constructed to improve the efficiency of data queries.

A. Private Counting Method

We here present a new R-PDC method, the core of which is to construct an ideal counting curve (also called query function) applied into $\mathbf{Query}$ algorithm. Formally, for a given target set W, an ideal counting curve $g(x)$ is constructed with the properties that $g(x) = 1$ for any $x \in W$ and $g(x) = 0$ for any $x \notin W$. That is, the elements in S are divided into two categories by W: one is the element that belongs to W where the corresponding value of $g(x)$ is set to 1; the other does not belong to W where the value of $g(x)$ is set to 0. So that the correct counting result is $Count(S, W) = \sum_{i=1}^{m} g(s_i)$.

In order to improve efficiency of query, a scaleable algebraic curve $f(x)$, called as *approximate query function*, is generated to approximate the above $g(x)$. This means that there exists an error function $e(x)$ defined as $e(x) = f(x) - g(x)$, but this error must be controlled within the allowable range. This process is called **curve approximation**. Formally, the definition of $f(x)$ is given as follows.

Definition 9 (Approximate Query Function, AQF): Let $f : S \to \mathbb{R}$ be a real-valued function whose domain is a finite set S. Then, given a target set $W \subseteq S$ and a positive real number ϵ, f is said to be ϵ-AQF if for any $x \in W$, the following property holds:

$$f(x) = \begin{cases} 1 + e(x) & x \in W, \\ 0 + e(x) & x \notin W, \end{cases} \tag{3}$$

where $e(x)$ is a allowable error with $|e(x)| < \epsilon$.

According to Definition 9, the constructed $f(x)$ should meet the following properties: for the specific element x' in S, the corresponding value $f(x')$ of the function is equal or approximately equal to the value $g(x')$, i.e., $g(x') \approx f(x')$. In other words, there exists some specific two-dimensional points $\{(x', g(x'))\}$ in coordinates, such that the points $\{(x', f(x'))\}$ on the curve $f(x)$ coincides or approximately coincides with these points $\{(x', g(x'))\}$. Moreover, for the element that belongs to W, the value of $f(x)$ is approximately set to 1; otherwise, the value of $f(x)$ is approximately set to 0.

Finally, we discuss how to use polynomial fitting method to convert the $f(x)$ into a polynomial $p(x)$ of degree λ for performing secure data counting on real numbers with fixed-point representation. To do it, for two given domain S and W, a general and efficient R-PDC method is presented by using curve approximation, and the counting procedure is described in Fig.4.

1) **Curve Generation:** construct an ideal counting curve $g(x)$ by the target set W, and generate an AQF $f(x)$ to approximate $g(x)$ by using curve approximation.
2) **Single Counting:** calculate $f(s_i)$ for any $i \in [1, m]$ to check whether s_i in S belongs to W. Specifically, if $s_i \in W$, then the value of $f(s_i)$ is approximately set to 1, i.e., $f(s_i) \approx 1$; otherwise, $f(s_i) \approx 0$.
3) **Cumulative Counting:** accumulate $f(s_i)$ for all $i \in [1, m]$ to get the sum $\sum_{i=1}^{m} f(s_i)$.
4) **Result Recovery:** obtain the counting result by rounding the above sum $\sum_{i=1}^{m} f(s_i)$, i.e., $Count(S, W) = \lfloor \sum_{i=1}^{m} f(s_i) \rceil$, where the cumulative counting error satisfies $|\sum_{i=1}^{m} e(s_i)| = |\sum_{i=1}^{m} (f(s_i) - g(s_i))| < \epsilon$.

Fig. 4. The R-PDC method based on curve approximation.

In Fig.4, the cumulative counting error $\sum_{i=1}^{m} e(s_i)$ refers to the sum of the counting errors generated by using AQF $f(x)$ to count all m elements s_i in S. To ensure the correctness of $Count(S, W)$ on S and W, it is required that the cumulative counting error satisfies $|\sum_{i=1}^{m} e(s_i)| < \epsilon$, where $0 < \epsilon \leq 0.5$. According to the above settings, we have $Count(S, W) = \lfloor \sum_{i=1}^{m} f(s_i) \rceil = \sum_{i=1}^{m} g(s_i) + \lfloor \sum_{i=1}^{m} e(s_i) \rceil = \sum_{i=1}^{m} g(s_i)$. This means that the $f(x)$ can replace $g(x)$ to achieve efficient counting within an allowable counting error. Moreover, the subsequent Theorem 1 is presented to illustrate that the errors can be limited into an allowable range to ensure the correctness of counting results.

B. Construction of AQF

We now consider the construction of AQF $f(x)$ for counting a target element w. For this case, the constructed $f(x)$ neither depends on the elements in data set S, nor knows whether the target element w is in S.

Given the data set $S = \{s_1, s_2, \cdots, s_m\}$, where $s_i \in \mathbb{Z}$ for all $i \in [1, m]$. Next, we consider the PDC problem over S with the target element $w = 0$. In Fig.5, the red solid line represents an ideal counting curve $g(x)$ that satisfies this counting case. Exactly, the curve $g(x)$ should meet the properties that $g(x) = 1$ when $x = 0$, and $g(x) = 0$, otherwise.

According to the properties of $g(x)$, multiple curves may be constructed to approximate this ideal counting curve. As shown in Fig.5, the blue dotted line represents a candidate AQF $f(x)$. The curve $f(x)$ meets the following properties:

1) The value of $f(x)$ is as close to 1 as possible when $x = 0$, i.e., $f(x) \approx 1$ or $f(x) = 1$ when $x = 0$;
2) The value of $f(x)$ approaches 0 or equals 0 when $x \neq 0$, i.e., $f(x) \approx 0$ or $f(x) = 0$ when $x \neq 0$;
3) The $f(x)$ is an even function since $g(x)$ is even.

According to the above properties, the constructing idea of $f(x)$ based on curve approximation is given as follows: at first, the function $f(x) = x^{-a}$ is chosen as the curve approximation of $g(x)$, where $a \geq 1$ and a is an integer. Besides, the larger the value of a is, the better the approximation effect of $f(x)$ is. Then, the function $f(x) = (x^2)^{-a}$ is established by curve symmetry in property 3). Finally, the function $f(x) = (x^2 +$

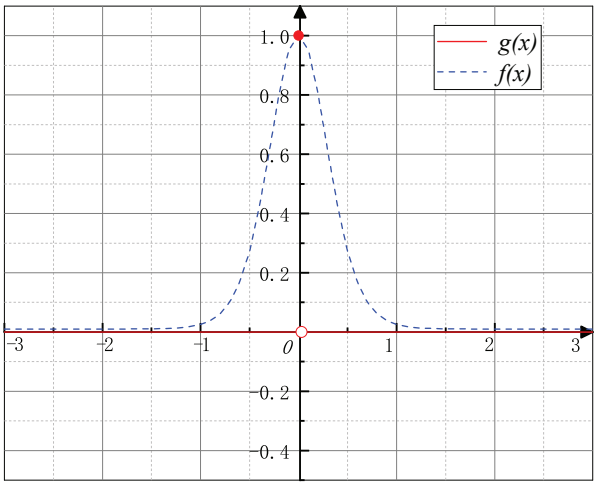


Fig. 5. An example of curve for counting a target element.

$1)^{-a}$ is established in terms of the properties 1) and 2). Thus, the final AQF $f(x)$ is expressed as

$$f(x) = (x^2 + 1)^{-a}. \tag{4}$$

The above curve $f(x)$ can be used to count the element $w = 0$ in any data set S. According to our R-PDC method (see Fig.4), the counting result is $Count(S, w) = \lfloor \sum_{i=1}^{m} f(s_i) \rceil$, where $i \in [1, m]$. Note that, the rounding method is used to ensure the correctness of counting results since the R-PDC method using curve approximation has counting errors.

Moreover, the curve $f(x)$ is able to count any element w by translating along the x-axis. That is, for any data set S and target element w, the AQF $f(x)$ can be uniformly defined as

$$f(x) = ((x - w)^2 + 1)^{-a}, \tag{5}$$

where $a \geq 1$ and a is an integer. Thus, this kind of curve construction can meet query requirements for different target elements by curve translation.

In order to ensure that the constructed $f(x)$ can count any element w in S, the Theorem 1 is given as follows.

Theorem 1: Given the data set $S = \{s_1, s_2, \cdots, s_m\}$ and the target element w, the function $f(x) = ((x-w)^2+1)^{-a}$ is a $(m, \epsilon)-$AQF to count w, provided that $m \cdot 2^{-a} < \epsilon$, where a is a variable integer and $a \geq 1$, m is the number of elements in S, ϵ is the upper bound of allowable error satisfying $\epsilon \in (0, 0.5]$.

Proof 1: For the given data set $S = \{s_1, s_2, \cdots, s_m\}$ and target element w, the counting error on the element $s_i \in \mathbb{Z}$ is

$$\begin{aligned} e(s_i) &= f(s_i) - g(s_i) \\ &= \begin{cases} 0 & s_i = w, \\ ((s_i - w)^2 + 1)^{-a} & s_i \neq w, \end{cases} \end{aligned} \tag{6}$$

where $i \in [1, m]$.

According to Equation (6), when $s_i = w$, the counting error of $f(x)$ at $x = w$ is 0, i.e., $e(s_i) = 0$, and the error function $e(x)$ is expressed as $e(x) = ((x - w)^2 + 1)^{-a}$ when $s_i \neq w$. Obviously, the function $e(x)$ is symmetric about the line $x = w$, and it decreases monotonically when $x > w$. Thus, $e(x)$ gets the maximum value at $x = w + 1$ for all $x \in \mathbb{Z}$, so that the equation $0 < e(s_i) \leq e(w+1) = 2^{-a}$ holds for all $s_i \in \mathbb{Z}$ ($s_i \neq w$). Further, the cumulative counting error, denoted as $\sum_{i=1}^{m} e(s_i)$, for all s_i ($i \in [1, m]$) satisfies

$$\sum_{i=1}^{m} e(s_i) \leq m \cdot e(w + 1) = \frac{m}{2^a} \tag{7}$$

Assume that ϵ is the upper bound of allowable error. To ensure the correctness of data counting, it is required that the equation $0 \leq \sum_{i=1}^{m} e(s_i) \leq m \cdot 2^{-a} < \epsilon$ holds, where $0 < \epsilon \leq 0.5$. So that $a > \frac{\log m - \log \epsilon}{\log 2} \approx 3.3(\log m - \log \epsilon)$ holds, and a is proportional to $\log m$. When m is fixed, the smaller the value of ϵ is, the larger the value of a is, and the smaller the cumulative counting error $\sum_{i=1}^{m} e(s_i)$ is. According to our R-PDC method, the counting result of w is

$$\begin{aligned} Count(S, w) &= \left\lfloor \sum_{i=1}^{m} f(s_i) \right\rceil = \left\lfloor \sum_{i=1}^{m} (g(s_i) + e(s_i)) \right\rceil \\ &= \sum_{i=1}^{m} g(s_i) + \left\lfloor \sum_{i=1}^{m} e(s_i) \right\rceil. \end{aligned} \tag{8}$$

According to Equation (8), if the equation $0 \leq \sum_{i=1}^{m} e(s_i) < \epsilon$ holds and $0 < \epsilon \leq 0.5$, then we have $\lfloor \sum_{i=1}^{m} e(s_i) \rceil = 0$. So that $Count(S, w) = \sum_{i=1}^{m} g(s_i)$. That is, the $f(x)$ can replace $g(x)$ to achieve data counting by using our R-PDC method. Thus, we complete the proof of Theorem 1.

Example 3: Let $S = \{-4, 0, 3, 4, 3, 2, 2, -3, 1, 1, -2, 0, 2, 4 - 1, -1, -4, -1, 0, -3\}$, $w = 0$ and $\epsilon = 0.5$. According to Theorem 1, when the number m of element in S is 20, i.e., $m = 20$, the parameter a satisfies

$$a > \frac{\log m - \log \epsilon}{\log 2} \approx 5.3219. \tag{9}$$

Here, we choose $a = 6$. Then, the function $f(x) = (x^2 + 1)^{-6}$ is a candidate AQF that is used for counting $w = 0$. Exactly, for given S, the counting result of w is $Count(S, 0) = \left\lfloor \sum_{i=1}^{20} f(s_i) \right\rceil = \lfloor 3.0784 \rceil = 3$, and the cumulative counting error is $\sum_{i=1}^{20} e(s_i) = \sum_{i=1}^{20} (f(s_i) - g(s_i)) = 0.0784 < 0.5$. Thus, the counting result of $Count(S, 0)$ is consistent with the number of element $w = 0$ in S.

Moreover, considering the presented R-PDC method only involves addition and multiplication operations, polynomial fitting method is adopted to convert $f(x)$ into a polynomial $p(x)$ of degree λ, i.e., $p(x) = \sum_{k=0}^{\lambda} \tilde{\eta}_k x^k$, thereby implementing private data counting with fixed-point representation.

Example 4: From Example 3, the function $f(x) = (x^2 + 1)^{-6}$ can be approximated by a candidate polynomial $p(x)$ of degree $\lambda = 24$, where the coefficients of $p(x)$ are listed in Table III. For clarity, Fig.6 shows the polynomial fitted curve $p(x)$ of AQF $f(x)$ for target element $w = 0$. Thus, for given S, the counting result of $w = 0$ by using $p(x)$ is $Count(S, 0) = \left\lfloor \sum_{i=1}^{20} p(s_i) \right\rceil = \lfloor 3.0499 \rceil = 3$, and the cumulative counting error is $\sum_{i=1}^{20} e(s_i) = \sum_{i=1}^{20} (p(s_i) - g(s_i)) = 0.0499 < 0.5$.

Remark: From Examples 3 and 4, it is easy to see that, the cumulative counting error of counting $w = 0$ on S with $f(x)$ is different from that with $p(x)$ (0.0784 and 0.0499, respectively). This is because $p(x)$ is the fitted polynomial of

TABLE III
THE COEFFICIENTS OF POLYNOMIAL $p(x)$ WITH DEGREE $\lambda = 24$ FOR AQF $f(x)$.

	$\tilde{\eta}_k$	$\tilde{\eta}_{k+1}$	$\tilde{\eta}_{k+2}$	$\tilde{\eta}_{k+3}$	$\tilde{\eta}_{k+4}$
$k=0$	0.9315	1.4235×10^{-9}	-3.7059	-8.7902×10^{-9}	5.7551
$k=5$	1.5909×10^{-8}	-4.6177	-1.3196×10^{-8}	2.1809	6.0434×10^{-9}
$k=10$	-0.6535	-1.6816×10^{-9}	0.1298	2.9903×10^{-10}	-0.0175
$k=15$	-3.4703×10^{-11}	0.0016	2.6153×10^{-12}	-9.8795×10^{-5}	-1.2332×10^{-13}
$k=20$	3.8993×10^{-6}	3.3037×10^{-15}	-8.9183×10^{-8}	-3.8366×10^{-17}	8.9852×10^{-10}

degree $\lambda = 24$ for given $f(x)$, and the counting error of $p(x)$ is related to the degree λ as well as the elements in S. Exactly, if the degree λ and the elements in S change, the fitted curve $p(x)$ of degree λ will also change. So, the values $p(s_i)$ will be different for the same element s_i, thereby introducing distinct cumulative counting errors.

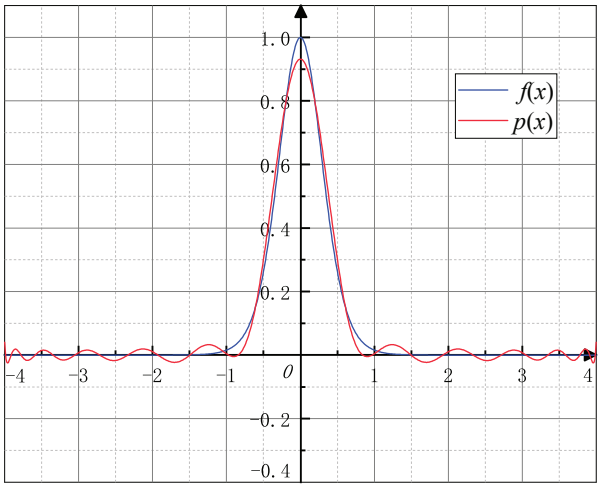


Fig. 6. Polynomial fitted curve $p(x)$ of AQF $f(x)$ for target element $w = 0$.

Finally, according to the coefficients of $p(x)$ in Table III, we choose $\gamma = 21$ and make use of $p(x)$ to perform multiparty counting on fixed-point numbers. As a result, the correctness of using $p(x)$ instead of $f(x)$ for multiparty joint data counting follows the Theorem 1.

VI. THE R-PDC SCHEME

In this section, an efficient R-PDC scheme is presented by integrating F-FHE[2] with the R-PDC method. First of all, the F-FHE cryptosystem is deployed into multi-server system, which contains n servers, $P = \{P_1, P_2, \cdots, P_n\}$. Then, the public parameter pp of F-FHE is published to the owners $\mathcal{O}$, the inquirer $\mathcal{Q}$ and the multiservers $\mathcal{MS}$. On this basis, for given data set S and target set W, the R-PDC scheme consists of four algorithms, including **Encap**, **Query**, **Answer** and **Reduct**, as follows.

• $\mathbf{Encap}(S)$: As described in Algorithm 1, the **Encap** algorithm is executed to generate the encrypted data set $\mathbf{D}$ for a given data set S. According to step 1-3 in this algorithm, the owners $\mathcal{O}$ invokes $\mathbf{Enc_{pp}}$ of F-FHE, i.e.,

$$\mathbf{C}_{s_i} = \mathbf{Enc_{pp}}(s_i) = (c_{s_i,1}, c_{s_i,2}, \cdots, c_{s_i,n})^T \in \mathbb{Z}_p^n,$$

[2]A candidate F-FHE cryptosystem is given in the supplementary material.

to encapsulate each data s_i in S for $i \in [1, m]$, where s_i is either integer or fixed-point number. Then, the share $c_{s_i,k}$ is sent to the server P_k for $k \in [1, n]$. So, an encrypted data set $\mathbf{D} = (\mathbf{C}_{s_1}, \mathbf{C}_{s_2}, \cdots, \mathbf{C}_{s_m}) \in \mathbb{Z}_p^{n \times m}$ is generated and stored on the multiservers $\mathcal{MS}$ by step 4, where the k-th row of $\mathbf{D}$ indicates the shares stored on P_k.

Algorithm 1 $\mathbf{Encap}(S)$

Input: the data set $S = \{s_1, s_2, \cdots, s_m\}$.
Output: the encrypted data set $\mathbf{D} = (\mathbf{C}_{s_1}, \mathbf{C}_{s_2}, \cdots, \mathbf{C}_{s_m})$.
1: **for** $i = 1$ to m **do**
2: $\mathcal{O}$ computes $\mathbf{C}_{s_i} = (c_{s_i,1}, c_{s_i,2}, \cdots, c_{s_i,n})^T \leftarrow \mathbf{Enc_{pp}}(s_i)$;
3: **end for**
4: $\mathcal{O}$ sends $\mathbf{D} = (\mathbf{C}_{s_1}, \mathbf{C}_{s_2}, \cdots, \mathbf{C}_{s_m})$ to $\mathcal{MS}$;
5: **Return** $\mathbf{D} = (\mathbf{C}_{s_1}, \mathbf{C}_{s_2}, \cdots, \mathbf{C}_{s_m})$.

• $\mathbf{Query}(W)$: In the **Query** algorithm (described in Algorithm 2), the inquirer $\mathcal{Q}$ firstly constructs an AQF $f(x)$ for a given target set W by using the R-PDC method (see Fig.4). Then, $\mathcal{Q}$ converts $f(x)$ into a polynomial $p(x) = \sum_{k=0}^{\lambda} \tilde{\eta}_k x^k$ of degree λ by using polynomial fitting method. After that, in step 3-5, $\mathcal{Q}$ invokes $\mathbf{Enc_{pp}}$ to generate

$$\mathbf{C}_{\tilde{\eta}_i} = \mathbf{Enc_{pp}}(\tilde{\eta}_i) = (c_{\eta_i,1}, c_{\eta_i,2}, \cdots, c_{\eta_i,n})^T \in \mathbb{Z}_p^n,$$

for coefficient $\tilde{\eta}_i$, where $i \in [0, \lambda]$. Finally, the data share $c_{\eta_i,k}$ is sent to the server P_k for $k \in [1, n]$. So, a query parameter $\mathbf{\Phi} = (\mathbf{C}_{\tilde{\eta}_0}, \mathbf{C}_{\tilde{\eta}_1}, \cdots, \mathbf{C}_{\tilde{\eta}_\lambda}) \in \mathbb{Z}_p^{n \times (\lambda+1)}$ is constructed and stored on the multiservers $\mathcal{MS}$, where the k-th row of $\mathbf{\Phi}$ indicates the shares stored on P_k.

Algorithm 2 $\mathbf{Query}(W)$

Input: the target set W.
Output: the query parameter $\mathbf{\Phi} = (\mathbf{C}_{\tilde{\eta}_0}, \mathbf{C}_{\tilde{\eta}_1}, \cdots, \mathbf{C}_{\tilde{\eta}_\lambda})$.
1: $\mathcal{Q}$ constructs an AQF $f(x)$ on W;
2: $\mathcal{Q}$ converts $f(x)$ into a polynomial $p(x) = \sum_{k=0}^{\lambda} \tilde{\eta}_k x^k$;
3: **for** $i = 1$ to λ **do**
4: $\mathcal{Q}$ computes $\mathbf{C}_{\tilde{\eta}_i} = (c_{\eta_i,1}, c_{\eta_i,2}, \cdots, c_{\eta_i,n})^T \leftarrow \mathbf{Enc_{pp}}(\tilde{\eta}_i)$;
5: **end for**
6: $\mathcal{Q}$ sends $\mathbf{\Phi} = (\mathbf{C}_{\tilde{\eta}_0}, \mathbf{C}_{\tilde{\eta}_1}, \cdots, \mathbf{C}_{\tilde{\eta}_\lambda})$ to $\mathcal{MS}$;
7: **Return** $\mathbf{\Phi} = (\mathbf{C}_{\tilde{\eta}_0}, \mathbf{C}_{\tilde{\eta}_1}, \cdots, \mathbf{C}_{\tilde{\eta}_\lambda})$.

• $\mathbf{Answer}(\mathbf{D}, \mathbf{\Phi})$: In the **Answer** algorithm, the multi-servers $\mathcal{MS}$, using $\mathbf{Eval_{pp}}$ of F-FHE, jointly perform secure fixed-point computation to generate a query response $\boldsymbol{\alpha}$. The process is described in Algorithm 3. Specifically, according to step 3-6, $\mathcal{MS}$ firstly use multiparty fixed-point fully homomorphism iteratively to compute the k-th power s_i^k for each element s_i in S. Then, the items $\tilde{\eta}_k \cdot s_i^k$ of $p(s_i)$, denoted as $item_{i,k}$, are iteratively computed in step 7-10. After that, the result $\mathbf{C}_{p(s_i)}^{(\lambda+1)}$ of $p(s_i)$ is computed by accumulating all

of $item_{i,k}$ in step 11-14. Finally, for all m elements in S, the results $\mathbf{C}_{p(s_k)}^{(\lambda+1)}$ are accumulated to get the sum $\mathbf{C}_{\alpha}^{(m)}$ for $k \in [1, m]$ by step 15, where $\mathbf{C}_{\alpha}^{(m)}$ is denoted as the query response $\boldsymbol{\alpha} = (c_{\alpha,1}, c_{\alpha,2}, \cdots, c_{\alpha,n})^T \in \mathbb{Z}_p^n$.

Algorithm 3 $\mathbf{Answer}(\mathbf{D}, \mathbf{\Phi})$

Input: the encrypted data set $\mathbf{D}$, and the query parameter $\mathbf{\Phi}$.
Output: the query response $\boldsymbol{\alpha} = (c_{\alpha,1}, c_{\alpha,2}, \cdots, c_{\alpha,n})^T$.
1: Set $\mathbf{C}_{\alpha}^{(0)} = 0$;
2: **for** $i = 1$ to m **do**
3: Set $(\mathbf{C}_{s_i})^1 = \mathbf{C}_{s_i}$ and $(\mathbf{C}_{s_i})^0 = 1$;
4: **for** $k = 1$ to λ **do**
5: $\mathcal{MS}$ computes $(\mathbf{C}_{s_i})^k \leftarrow \mathbf{Eval_{pp}}((\mathbf{C}_{s_i})^{k-1}, *, \mathbf{C}_{s_i})$;
6: **end for**
7: Set $item_{i,0} = \mathbf{C}_{\tilde{\eta}_0}$;
8: **for** $k = 0$ to λ **do**
9: $\mathcal{MS}$ computes $item_{i,k} \leftarrow \mathbf{Eval_{pp}}(\mathbf{C}_{\tilde{\eta}_k}, *, (\mathbf{C}_{s_i})^k)$;
10: **end for**
11: Set $\mathbf{C}_{p(s_i)}^{(0)} = 0$;
12: **for** $k = 0$ to λ **do**
13: $\mathcal{MS}$ computes $\mathbf{C}_{p(s_i)}^{(k+1)} \leftarrow \mathbf{Eval_{pp}}(\mathbf{C}_{p(s_i)}^{(k)}, +, item_{i,k})$;
14: **end for**
15: $\mathcal{MS}$ computes $\mathbf{C}_{\alpha}^{(i)} \leftarrow \mathbf{Eval_{pp}}(\mathbf{C}_{\alpha}^{(i-1)}, +, \mathbf{C}_{p(s_i)}^{(\lambda+1)})$;
16: **end for**
17: Set $\mathbf{C}_{\alpha}^{(m)}$ as $\boldsymbol{\alpha} = (c_{\alpha,1}, c_{\alpha,2}, \cdots, c_{\alpha,n})^T$;
18: $\mathcal{MS}$ sends $\boldsymbol{\alpha} = (c_{\alpha,1}, c_{\alpha,2}, \cdots, c_{\alpha,n})^T$ to $\mathcal{Q}$;
19: **Return** $\boldsymbol{\alpha} = (c_{\alpha,1}, c_{\alpha,2}, \cdots, c_{\alpha,n})^T$.

Algorithm 4 $\mathbf{Reduct}(\boldsymbol{\alpha})$

Input: the query response $\boldsymbol{\alpha}$.
Output: the counting result β.
1: $\mathcal{Q}$ obtains $\boldsymbol{\alpha} = (c_{\alpha,1}, c_{\alpha,2}, \cdots, c_{\alpha,n})^T$ from $\mathcal{MS}$;
2: $\mathcal{Q}$ computes $\beta \leftarrow \mathbf{Dec_{pp}}([\boldsymbol{\alpha}]) \in \mathbb{P}$;
3: **Return** β.

• $\mathbf{Reduct}(\boldsymbol{\alpha})$: After receiving the query response $\boldsymbol{\alpha} = (c_{\alpha,1}, c_{\alpha,2}, \cdots, c_{\alpha,n})^T \in \mathbb{Z}_p^n$ from the multiservers $\mathcal{MS}$, the inquirer $\mathcal{Q}$ invokes the $\mathbf{Dec_{pp}}$ algorithm of F-FHE to retrieve the data counting result $\beta \leftarrow \mathbf{Dec_{pp}}([\boldsymbol{\alpha}]) \in \mathbb{P}$. The algorithm process is described in Algorithm 4.

VII. Security Evaluation

In this section, we define two security properties, data and query privacy, for the R-PDC scheme. Moreover, the presented scheme is proved to be statistically secure against chosen element attack with any counting query for query privacy and against chosen element attack with designation query for data privacy, respectively.

A. Query Privacy

In our security analysis, the R-PDC scheme is built on a multi-server system with n servers, $P = \{P_1, P_2, \cdots, P_n\}$. Assume that the number of corrupted servers be less than t under the control of adversary. Given the public parameter pp under security parameter κ, any two different query elements, s_0 and s_1, the query privacy of R-PDC requires that the difference of two corresponding query parameters, $\mathbf{Query}(s_0)$ and $\mathbf{Query}(s_1)$, are statistically indistinguishable for any computational unbounded adversary $\mathcal{A}$, who controls over some subset $T \subseteq P$ of n servers. That is,

$$\Pr\left[b' = b : \begin{array}{l} \forall S \in \{s_0, s_1\}^*, \mathbf{Encap}(S) = \mathbf{D}, \\ b \in_R \{0,1\}, \mathbf{Query}(s_b) = \mathbf{\Phi}, \\ \mathcal{A}(s_0, s_1, \mathbf{D}, [\mathbf{\Phi}]_T) = b' \end{array}\right] \leq \frac{1}{2} + \mu(\kappa), \tag{10}$$

for a negligible function $\mu(\kappa)$, where the data set S is any sequence over s_0 and s_1, i.e., $S \in \{s_0, s_1\}^*$, $[\mathbf{\Phi}]_T$ denotes partial information of $\mathbf{\Phi}$ held by a set T of corrupted servers.

The query privacy of R-PDC can be modeled by an indistinguishable game against chosen element attack with any counting query (called IND-CEA-CQ) as follows.

- **Init.** The adversary $\mathcal{A}$ chooses two different elements, s_0 and s_1, randomly and sends them to the owners $\mathcal{O}$.
- **Setup.** The owners $\mathcal{O}$ constructs an arbitrary data set S with s_0 and s_1, i.e., the sequence $S \in \{s_0, s_1\}^*$, and generates an encrypted data set $\mathbf{D}$ by invoking $\mathbf{Encap}(S)$. After that, $\mathcal{O}$ sends $\mathbf{D}$ to $\mathcal{A}$.
- **Challenge Query.** The inquirer $\mathcal{Q}$ flips a coin $b \in \{0, 1\}$ randomly and computes $\mathbf{\Phi} = \mathbf{Query}(s_b)$. Then, $\mathcal{Q}$ sends $\mathbf{\Phi}$ to the n servers.
- **Guess.** $\mathcal{A}$ exploits $[\mathbf{\Phi}]_T$, offered by a set T of corrupted servers, outputting a guess $b' \in \{0, 1\}$ of b.

We define by $Adv_{R-PDC}^{\text{IND-CEA-CQ}}(\mathcal{A})$ the advantage of $\mathcal{A}$ guessing b correctly in the above IND-CEA-CQ game as

$$\begin{aligned} &Adv_{R-PDC}^{\text{IND-CEA-CQ}}(\mathcal{A}) \\ &= \Pr\left[b' = b : \begin{array}{l} \forall S \in \{s_0, s_1\}^*, \mathbf{Encap}(S) = \mathbf{D}, \\ b \in_R \{0,1\}, \mathbf{Query}(s_b) = \mathbf{\Phi}, \\ \mathcal{A}(s_0, s_1, \mathbf{D}, [\mathbf{\Phi}]_T) = b' \end{array}\right] - \frac{1}{2}. \end{aligned} \tag{11}$$

Definition 10: A R-PDC scheme is said to be statistically $(t-1)$-secure against chosen element attack with any counting query if the advantage $Adv_{R-PDC}^{\text{IND-CEA-CQ}}(\mathcal{A})$ is negligible for all computational unbounded adversaries $\mathcal{A}$ and any set T ($|T| < t$) of corrupted servers.

Theorem 2: Our R-PDC scheme is statistically $(t-1)$-secure against chosen element attack with any counting query in the presence of a computational unbounded adversary if Shamir's (t, n)-threshold secret sharing scheme holds.

The theorem proof is shown in the supplementary material.

B. Data Privacy

The data privacy of the R-PDC scheme is defined as the zero-knowledge of data counting process, the precondition of which is that any information of stored data will not be revealed to the servers in data counting process. Formally, given the public parameter pp under security parameter κ, any two different elements, s_0 and s_1, and any computational unbounded adversary $\mathcal{A}$, who controls over some subset $T \subseteq P$ of n servers, we have

$$\Pr\left[b' = b : \begin{array}{l} b \in_R \{0,1\}, \mathbf{Encap}(s_b) = \mathbf{D}, \\ \sigma \in_R \{0,1\}, \mathbf{Query}(s_\sigma) = \mathbf{\Phi}, \\ \mathbf{Answer}(\mathbf{D}, \mathbf{\Phi}) = \boldsymbol{\alpha}, \\ \mathcal{A}(s_0, s_1, \sigma, [\mathbf{D}]_T, [\mathbf{\Phi}]_T, [\boldsymbol{\alpha}]_T) = b' \end{array}\right] \leq \frac{1}{2} + \mu(\kappa), \tag{12}$$

TABLE IV
THE COMPUTATIONAL COMPLEXITY OF EACH ALGORITHM IN OUR SCHEME.

Algorithm	Number of Operations		Computational Cost	Time Complexity
	Multiplication	Addition		
Encap	$(2t-3)nm$	$(t-1)nm$	$(3t-4)nm$	$O(nm)$
Query	$(\lambda+1)(2t-3)n$	$(\lambda+1)(t-1)n$	$(\lambda+1)(3t-4)n$	$O(n)$
Answer	$(2\lambda+1)((2t-2)n\gamma$ $+4\gamma+(2t-2)n+2)m$	$(2\lambda+1)((t+1)n\gamma$ $+(t+1)n+1)m+(\lambda+1)m$	$(2\lambda+1)((3t-1)n\gamma+4\gamma+$ $(3t-1)n+3)m+(\lambda+1)m$	$O(nm\gamma)$
Reduct	n	$n-1$	$2n-1$	$O(n)$

for a negligible function $\mu(\kappa)$, where $[\mathbf{D}]_T$, $[\mathbf{\Phi}]_T$ and $[\boldsymbol{\alpha}]_T$ denote partial information of $\mathbf{D}$, $\mathbf{\Phi}$ and $\boldsymbol{\alpha}$ held by a set T of corrupted servers, respectively.

The data privacy of R-PDC can be modeled by an indistinguishable game against chosen element attack with designation query (called IND-CEA-DQ) as follows.

- **Init.** The adversary $\mathcal{A}$ chooses two different elements, s_0 and s_1, randomly and sends them to the owners $\mathcal{O}$.
- **Setup.** The owners $\mathcal{O}$ flips a coin $b \in \{0,1\}$ randomly, and encrypts s_b to generate an encrypted data set $\mathbf{D}$ by invoking $\mathbf{Encap}(s_b)$. Then, $\mathcal{O}$ sends $\mathbf{D}$ to the n servers.
- **Challenge Query.** $\mathcal{A}$ flips a coin $\sigma \in \{0,1\}$ randomly, and requests the inquirer $\mathcal{Q}$ to compute $\mathbf{\Phi} = \mathbf{Query}(s_\sigma)$. Then, the n servers jointly compute $\boldsymbol{\alpha} = \mathbf{Answer}(\mathbf{D}, \mathbf{\Phi})$.
- **Guess.** $\mathcal{A}$ exploits $\{[\mathbf{D}]_T, [\mathbf{\Phi}]_T, [\boldsymbol{\alpha}]_T\}$ offered by a set T of corrupted servers to output a guess $b' \in \{0,1\}$ of b.

We define $Adv_{R-PDC}^{\text{IND-CEA-DQ}}(\mathcal{A})$ as the advantage of $\mathcal{A}$ guessing b correctly in the above IND-CEA-DQ game, i.e.,

$$Adv_{R-PDC}^{\text{IND-CEA-DQ}}(\mathcal{A}) = \Pr\left[b' = b : \begin{array}{l} b \in_R \{0,1\}, \mathbf{Encap}(s_b) = \mathbf{D}, \\ \sigma \in_R \{0,1\}, \mathbf{Query}(s_\sigma) = \mathbf{\Phi}, \\ \mathbf{Answer}(\mathbf{D}, \mathbf{\Phi}) = \boldsymbol{\alpha}, \\ \mathcal{A}(s_0, s_1, \sigma, [\mathbf{D}]_T, [\mathbf{\Phi}]_T, [\boldsymbol{\alpha}]_T) = b' \end{array}\right] - \frac{1}{2}. \tag{13}$$

Definition 11: A R-PDC scheme is said to be statistically $(t-1)$-secure against chosen element attack with designation query if the advantage $Adv_{R-PDC}^{\text{IND-CEA-DQ}}(\mathcal{A})$ is negligible for all computational unbounded adversaries $\mathcal{A}$ and any set T ($|T| < t$) of corrupted servers.

Theorem 3: Our R-PDC scheme is statistically $(t-1)$-secure against chosen element attack with designation query in the presence of a computational unbounded adversary if Shamir's (t,n)-threshold secret sharing scheme holds.

The theorem proof is shown in the supplementary material.

VIII. PERFORMANCE ANALYSIS

In this section, the computation and communication performance of the R-PDC scheme are evaluated theoretically and experimentally. Note that, the presented R-PDC is built on the F-FHE cryptosystem, so the performance of R-PDC is closely related to F-FHE, which is described in the supplementary material. In the experiments, the Iris dataset[3] provided by the UCI Machine Learning Repository is used for performance evaluation. This dataset contains 150 instances and 4 features, i.e., calyx length, calyx width, petal length and petal width, where each instance belongs to one of 3 classes. For simplicity, we extract real values of the feature "petal length" of all 150 instances to construct a set of data S with size $m = 150$ for our experimental evaluation.

The number n of servers, the size m of data set and the fractional part length γ of fixed-point number are three important parameters that affect the performance of our scheme. The other parameter, threshold t of SSS, is regarded as constant since $t \leq n$. In our experiment, we assume that $t = 4$ and $\lambda = 8$ for the lightweight examples. All the programs were executed on a Windows 10 (64-bit) PC with AMD Ryzen 7 4800H @2.90 GHz processor and 16G DDR4-RAM. A parallel development environment was built in Java by using MPJ Express[4] to simulate multiparty computing.

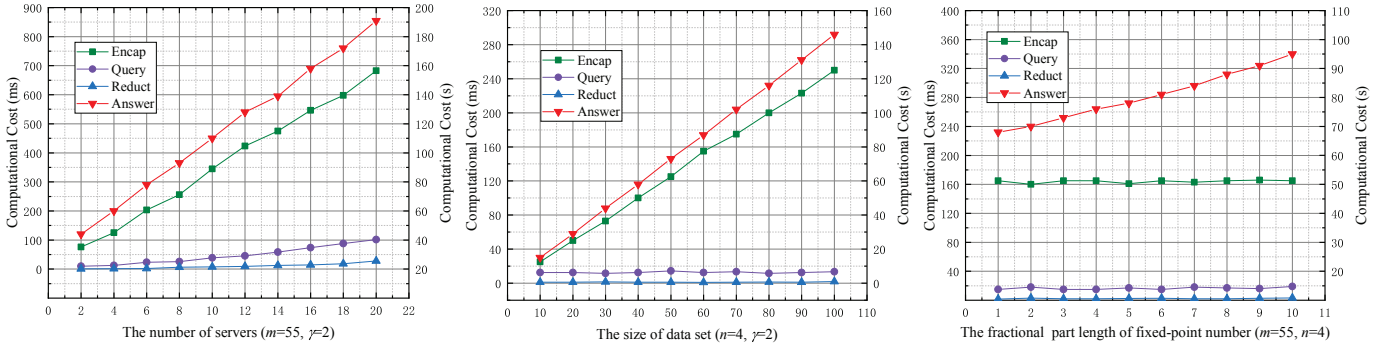
A. Computational Complexity of The Presented R-PDC

We first analyze the computational complexity of our R-PDC scheme. Here, the numbers of multiplications and additions in $\mathbb{Z}_p$ are taken as the measurement standard. Table IV lists the computational costs of main algorithms. It is easy to see that the time complexity $O(nm)$ of $\mathbf{Encap}$ is the quadratic formula on m and n. But the time complexity $O(nm\gamma)$ of $\mathbf{Answer}$ is the cubic formula on m, n and γ. So, the computational cost of $\mathbf{Answer}$ is higher than that of $\mathbf{Encap}$. Moreover, the time complexities of $\mathbf{Query}$ and $\mathbf{Reduct}$ are $O(n)$, which is linear correlation with n. Thus, the query and reduct process of the inquirer $\mathcal{Q}$ is more efficient, compared with joint computing of the multiservers $\mathcal{MS}$.

Next, the correctness of theoretical results was verified by experiments. In Fig.7(a), we show the experimental computation costs under different values of n when $m = 55$ and $\gamma = 2$, where the right y-axis corresponds to the computational cost of $\mathbf{Answer}$. As shown in Fig.7(a), the computational costs of the four algorithms, $\mathbf{Encap}$, $\mathbf{Query}$, $\mathbf{Answer}$ and $\mathbf{Reduct}$, increase linearly with n. Moreover, the computational cost of $\mathbf{Answer}$ is significantly larger than those of the other three algorithms. This is because the process of joint data counting in $\mathbf{Answer}$ involves much more complex calculation steps. For example, when $n = 4$, the computational cost of $\mathbf{Answer}$ is about 60 seconds, but the computational costs of the other three algorithms are less than 0.15 seconds.

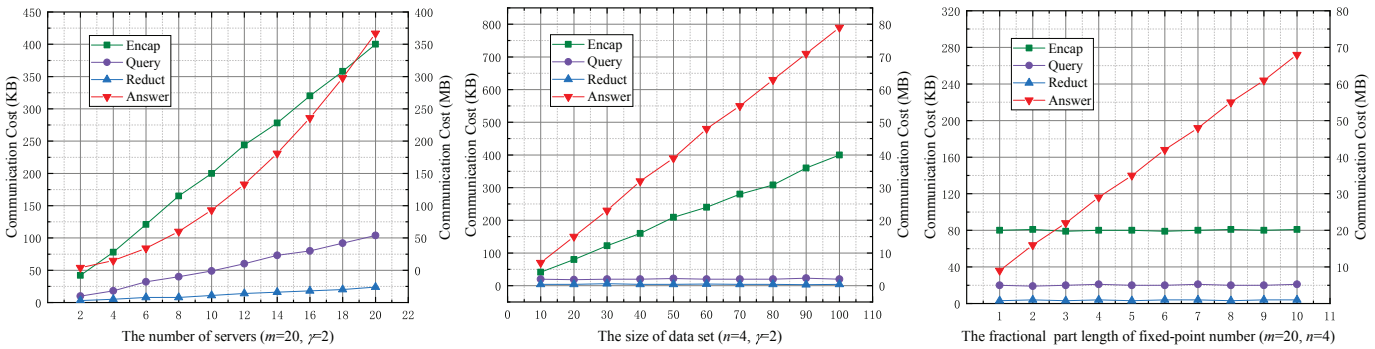
[3]https://archive.ics.uci.edu/dataset/53/iris

[4]MPJ Express is an open source Java message passing library that allows application developers to write and execute parallel applications for multicore processors and compute clusters/clouds.



(a) (b) (c)

Fig. 7. The trend curves of computational costs of algorithms. (a) $m = 55$, $\gamma = 2$, varying with the number n of servers. (b) $n = 4$, $\gamma = 2$, varying with the size m of data set. (c) $m = 55$, $n = 4$, varying with fractional part length γ of fixed-point number.



(a) (b) (c)

Fig. 8. The trend curves of communication costs of algorithms. (a) $m = 20$, $\gamma = 2$, varying with the number n of servers. (b) $n = 4$, $\gamma = 2$, varying with the size m of data set. (c) $m = 20$, $n = 4$, varying with fractional part length γ of fixed-point number.

In Fig.7(b), we show the trend curves of experimental computation costs under different values of m when $n = 4$ and $\gamma = 2$, where the right y-axis corresponds to the computational cost of **Answer**. Obviously, the computational costs of **Query** and **Reduct** are approximately fixed, but for **Encap** and **Answer**, their computational costs increase linearly with m. Moreover, it can be see that the computational costs of **Query** and **Reduct** are smaller than those of **Encap** and **Answer**. For example, when m varies from 10 to 100, the computational costs of **Query** and **Reduct** are fixed at about 0.001 and 0.012 seconds, respectively. For the other two algorithms, the computational cost of **Encap** is 0.16 seconds when m is 60, but that of **Answer** reaches 90 seconds.

Fig.7(c) shows the experimental computation costs under different values of γ when $m = 55$ and $n = 4$, where the right y-axis corresponds to computational cost of **Answer**. As shown in Fig.7(c), the computational costs of the three algorithms, **Encap**, **Query** and **Reduct**, are approximately fixed, but the computational cost of **Answer** increase linearly with γ. Moreover, the computational cost of **Answer** is larger than those of the other three algorithms. For example, when γ varies from 1 to 10, the computational costs of **Encap**, **Query** and **Reduct** are approximately fixed at 0.001, 0.015 and 0.165 seconds, respectively. When $\gamma = 4$, the computational cost of **Answer** reaches 75 seconds, but the computational costs of the other three algorithms are less than 0.165 seconds. So, the computational cost of our scheme is mainly concentrated in **Answer**, which is consistent with the theoretical analysis result in Table IV.

B. Communication Complexity of The Presented R-PDC

We next pay attention to communication complexity. Let $|b|$ denote the transmitted data length in one time of communication. Table V lists the communication costs of main algorithms. It is easy to see that the communication complexity of **Encap** is $O(nm)$, which is the quadratic formula on m and n. However, the communication complexity $O(n^2m\gamma)$ of **Answer** is the quartic formula on m, n and γ. It means that the communication cost of **Answer** is higher than that of **Encap**. Moreover, the communication complexities of both **Query** and **Reduct** are $O(n)$, which is linear correlation with n. Thus, compared with **Query** and **Reduct**, both **Encap** and **Answer** require higher communications.

For clarity, Fig.8(a) shows the experimental communication costs under different values of n when $m = 20$ and $\gamma = 2$, where the right y-axis corresponds to communication cost of **Answer**. As shown in Fig.8(a), the communication costs of the three algorithms, **Encap**, **Query** and **Reduct**, increase linearly with n, but the communication cost of **Answer** increases quadratically with n. So, the communication cost of **Answer** is higher than those of the other three algorithms. For example, when $n = 8$, the communication cost of **Answer** is

TABLE V
THE COMMUNICATION COMPLEXITY OF OUR SCHEME.

Algorithm	Entities	Communication Cost (bits)	Communication Complexity
Encap	$\mathcal{O} \rightarrow \mathcal{MS}$	$nm \cdot \|b\|$	$O(nm)$
Query	$\mathcal{Q} \rightarrow \mathcal{MS}$	$(\lambda+1)n \cdot \|b\|$	$O(n)$
Answer	$\mathcal{MS} \leftrightarrow \mathcal{MS}$	$((2n^2+n)\gamma + n^2)(2\lambda+1)m \cdot \|b\|$	$O(n^2 m\gamma)$
Reduct	$\mathcal{MS} \rightarrow \mathcal{Q}$	$n \cdot \|b\|$	$O(n)$

about 60MB, but those of the other three algorithms are less than 160KB (Exactly, 160KB, 40KB and 8KB, respectively).

In Fig.8(b), we shows the experimental communications costs under different values of m when $n = 4$ and $\gamma = 2$, where the right y-axis corresponds to communication cost of **Answer**. As shown in Fig.8(b), the communication costs of the two algorithms, **Query** and **Reduct**, are approximately fixed, but for **Encap** and **Answer**, their communication costs increase linearly with m. Moreover, the communication cost of **Answer** is larger than those of the other three algorithms. For example, when $m = 30$, the communication cost of **Answer** is about 24MB, but those of the other three algorithms are about 120KB, 20KB and 4KB, respectively.

Fig.8(c) shows the experimental costs of communications under different values of γ when $m = 20$ and $n = 4$, where the right y-axis corresponds to communication cost of **Answer**. Obviously, the communication costs of **Encap**, **Query** and **Reduct** are approximately fixed, but the communication cost of **Answer** increase linearly with γ. Moreover, the communication cost of **Answer** is larger than those of the other three algorithms. For example, when $\gamma = 6$, the communication cost of **Answer** is about 40MB, but those of **Encap** , **Query** and **Reduct** are only 80KB, 20KB and 4KB, respectively. So, the communication costs of our scheme is mainly concentrated in multiparty joint computing of **Answer**.

In summary, the experimental results of R-PDC is consistent with its theoretical analysis. Moreover, the whole performance test indicates that our R-PDC scheme is enough efficient for privacy-preserving queries.

IX. APPLICATIONS

The R-PDC method can be applied into various privacy-preserving queries based on data mining algorithms. We, taking the famous Naive Bayes Classifier and Apriori algorithm as examples, illustrate how to integrate R-PDC with these algorithms. Moreover, we show how to extend the R-PDC method, called *n-dimensional R-PDC*, to develop more practical applications.

A. Privacy-preserving Naive Bayes Classifier

Naive Bayes Classifier is a simple but efficient probability classifier based on Bayes' theorem. Exactly, given the attribute values $\{a_1, a_2, \cdots, a_m\}$ that describe a new instance, the Bayesian approach to classifying the instance is to assign the most probable class value $v_{map} \in V$, where V is the set of possible class values. Moreover, the Naive Bayes Classifier simplifies assumption that the attribute values are conditionally independent among them for given class values. Therefore, the class output v_{map} of Naive Bayes Classifier can be reduced as

$$v_{map} = argmax\left(P(v_j)\prod_{i=1}^{m} P(a_i|v_j)\right), v_j \in V. \quad (14)$$

According to this equation, the key of Naive Bayes Classifier is to calculate the probabilities $P(v_j)$ and $P(a_i|v_j)$, as follows.

First of all, the prior probabilities $P(v_j)$ can be estimated by the formula $P(v_j) = \frac{n_j}{n}$, where n_j is the number of instances with class value v_j and n is the total number of instances in the dataset. Undoubtedly, the number n_j can be calculated by our R-PDC method, i.e., $n_j = Count(S_v, v_j)$, where S_v is the set of class values of total instances in the private dataset. Therefore, $P(v_j)$ can be obtained for the public n.

Next, let $P(a_i|v_j)$ denote the conditional probability of an instance with the attribute value a_i belonging to a certain class v_j. We can calculate it by $P(a_i|v_j) = \frac{P(a_i v_j)}{P(v_j)} = \frac{n_{i,j}}{n_j}$, where $n_{i,j}$ is the number of instances with the attribute a_i and class v_j. Similarly, the 2-dimensional R-PDC method for counting $n_{i,j}$ is as follows: At first, two AQFs, $f_{v_j}(x)$ and $f_{a_i}(y)$, are constructed to count v_j and a_i, respectively. Next, a bivariate AQF $f_{v_j,a_i}(x,y) = f_{v_j}(x) \cdot f_{a_i}(y)$ can be generated, where $f_{v_j,a_i}(x,y) \approx 1$ for the instance with attribute value a_i and class value v_j simultaneously, and $f_{v_j,a_i}(x,y) \approx 0$, otherwise. Finally, the corresponding values of $f_{v_j,a_i}(x,y)$ are calculated for all instances and accumulated to get the value

$$n_{i,j} = Count(S_v \cup S_a, \{v_j, a_i\}) \approx \sum_{k=1}^{n} f_{v_j,a_i}(s_k^{(v)}, s_k^{(a)}),$$

where $S_v \cup S_a$ denotes a two-dimensional dataset that combines two attributes (v and a) together, $s_k^{(v)}$ and $s_k^{(a)}$ denote the attribute values of the k-th instance in $S_v \cup S_a$ for v and a. Therefore, the conditional probabilities $P(a_i|v_j)$ is obtained since n_j is calculated by the R-PDC method.

Finally, the class with the highest probability is chosen as the class output by Equation (14). Note that, the 2-dimensional R-PDC method can be further extended to n-dimensional R-PDC for a private dataset with multiple attributes.

B. Privacy-preserving Apriori Algorithm

Let's consider a more complex application scenario. The R-PDC method can be undoubtedly applied into Apriori[5] algorithm for the induction of association rules. An association rule is an implication in the form of $X \Rightarrow Y$, where X and Y are sets of items, called itemsets, and $X \cap Y = \emptyset$. An itemset X with k items is called k-itemset. The support of X, denoted by $Support(X)$, is the percentage of transactions that contain X to the total number n of transactions in the dataset. That is, $Support(X) = \frac{n_X}{n}$, where n_X is the number of transactions that contain X in the dataset.

An itemset X is called s-frequent if $Support(X) \geq sn$, where s indicates a required support threshold and $0 < s \leq 1$.

[5]Apriori is one of the best-known basic algorithm for mining frequent item sets in a set of transactions.

Actually, the main goal of Apriori algorithm is to find the set of all s-frequent itemsets for a given support threshold s. Thus, the key of Apriori is to calculate $Support(X)$ of all k-itemsets, i.e., to count n_X in the dataset for the public n.

The R-PDC method can be directly used to count n_X for any 1-itemset X ($k = 1$) since only one item need to be counted. If X is a k-itemset ($k > 1$), then it consists of k items, written as $X = \{a_1, a_2, \cdots, a_k\}$. Similarly, the k-dimensional R-PDC method can be used to count n_X as follows: At first, an AQF $f_{a_j}(x_j)$ is constructed to count each item a_j in X for $j \in [1, k]$. Next, a k-variable AQF $f_{a_1,\cdots,a_k}(x_1, \cdots, x_k) = f_{a_1}(x_1) \cdots f_{a_k}(x_k)$ can be generated, where $f_{a_1,\cdots,a_k}(x_1, \cdots, x_k) \approx 1$ for any transaction contains k-itemset X, and $f_{a_1,\cdots,a_k}(x_1, \cdots, x_k) \approx 0$, otherwise. Finally, the corresponding values of $f_{a_1,\cdots,a_k}(x_1, \cdots, x_k)$ for all transactions are calculated and accumulated to get the value

$$n_X = Count(S_{a_1} \cup S_{a_2} \cup \cdots \cup S_{a_k}, \{a_1, a_2, \cdots, a_k\}),$$

where $S_{a_1} \cup S_{a_2} \cup \cdots \cup S_{a_k}$ denotes a k-dimensional dataset that combines k items (i.e., $\{a_1, a_2, \cdots, a_k\}$) together. Therefore, the value of $Support(X) = \frac{n_X}{n}$ is obtained for the public n. According to the condition $Support(X) \geq sn$, all s-frequent itemsets X can be found from the given dataset.

X. Conclusion

In this paper, we present a new R-PDC method for privacy-preserving queries on multisource data in real-number fields. Meanwhile, a candidate F-FHE cryptosystem is deployed into multi-server system over IoT to implement ciphertext-state computation on fixed-point numbers. On this basis, we present an efficient R-PDC scheme and provide the full security proofs. The results of performance evaluations indicate the efficiency of our scheme for privacy-preserving queries.

An intriguing avenue for further research is the extension of our method from integer-based privacy computation to the realm of decimal numbers. Additionally, considerations could be extended to applying R-PDC into broader application scenarios, such as financial data analysis and healthcare data privacy protection, to further validate its practicality and generalizability. These exploring work will provide substantial support and insights for the advancement of privacy computation and establish a solid foundation for privacy-preserving techniques on real-number data.

Acknowledgments

This work was supported by the National Natural Science Foundation of China under Grant 61972032.

References

[1] X. Zhang, X. Zhu, W. Bao, L. T. Yang, J. Wang, H. Yan, and H. Chen, "Distributed learning on mobile devices: a new approach to data mining in the internet of things," *IEEE Internet of Things Journal*, vol. 8, no. 13, pp. 10 264–10 279, 2020.

[2] X. Yang, R. Lu, J. Shao, X. Tang, and H. Yang, "An efficient and privacy-preserving disease risk prediction scheme for e-healthcare," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 3284–3297, 2018.

[3] Z. Guan, X. Zhou, P. Liu, L. Wu, and W. Yang, "A blockchain-based dual-side privacy-preserving multiparty computation scheme for edge-enabled smart grid," *IEEE Internet of Things Journal*, vol. 9, no. 16, pp. 14 287–14 299, 2021.

[4] Y. Li and W. Xu, "Privpy: General and scalable privacy-preserving data mining," in *25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 2019, pp. 1299–1307.

[5] Y. Zhu, R. Yu, D. Ma, and W. C. Chu, "Cryptographic attribute-based access control (ABAC) for secure decision making of dynamic policy with multiauthority attribute tokens," *IEEE Transactions on Reliability*, vol. 68, no. 4, pp. 1330–1346, 2019.

[6] Z. Cai, X. Zheng, J. Wang, and Z. He, "Private data trading towards range counting queries in internet of things," *IEEE Transactions on Mobile Computing*, vol. 22, no. 8, pp. 4881–4897, 2023.

[7] C. Zhang, L. Zhu, J. Ni, C. Huang, and X. Shen, "Verifiable and privacy-preserving traffic flow statistics for advanced traffic management systems," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 9, pp. 10 336–10 347, 2020.

[8] H. Al-Hamadi and R. Chen, "Trust-based decision making for health iot systems," *IEEE Internet of Things Journal*, vol. 4, no. 5, pp. 1408–1419, 2017.

[9] Y. Miao, Y. Yang, X. Li, Z. Liu, H. Li, K.-K. R. Choo, and R. H. Deng, "Efficient privacy-preserving spatial range query over outsourced encrypted data," *IEEE Transactions on Information Forensics and Security*, vol. 18, pp. 3921–3933, 2023.

[10] Z. Yang, S. Zhong, and R. N. Wright, "Privacy-preserving classification of customer data without loss of accuracy," in *2005 SIAM International Conference on Data Mining*. SIAM, 2005, pp. 92–102.

[11] T. D. Vo-Huu, E.-O. Blass, and G. Noubir, "Epic: efficient privacy-preserving counting for mapreduce," *Computing*, vol. 101, no. 9, pp. 1265–1286, 2019.

[12] O. Catrina and S. De Hoogh, "Secure multiparty linear programming using fixed-point arithmetic," in *15th European Symposium on Research in Computer Security*. Springer, 2010, pp. 134–150.

[13] O. Catrina and A. Saxena, "Secure computation with fixed-point numbers," in *14th International Conference on Financial Cryptography and Data Security*. Springer, 2010, pp. 35–50.

[14] O. Catrina, "Round-efficient protocols for secure multiparty fixed-point arithmetic," in *2018 International Conference on Communications*. IEEE, 2018, pp. 431–436.

[15] M. Liedel, "Secure distributed computation of the square root and applications," in *8th International Conference on Information Security Practice and Experience*. Springer, 2012, pp. 277–288.

[16] C. Ugwuoke, Z. Erkin, and R. L. Lagendijk, "Secure fixed-point division for homomorphically encrypted operands," in *13th International Conference on Availability, Reliability and Security*. ACM, 2018, pp. 1–10.

[17] E. Boyle, N. Chandran, N. Gilboa, D. Gupta, Y. Ishai, N. Kumar, and M. Rathee, "Function secret sharing for mixed-mode and fixed-point secure computation," in *40th Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2021, pp. 871–900.

[18] F. Kerschbaum, A. Schröpfer, A. Zilli, R. Pibernik, O. Catrina, S. de Hoogh, B. Schoenmakers, S. Cimato, and E. Damiani, "Secure collaborative supply-chain management," *Computer*, vol. 44, no. 9, pp. 38–43, 2011.

[19] M. d. Cock, R. Dowsley, A. C. Nascimento, and S. C. Newman, "Fast, privacy preserving linear regression over distributed datasets based on pre-distributed data," in *8th ACM Workshop on Artificial Intelligence and Security*. ACM, 2015, pp. 3–14.

[20] A. Gascón, P. Schoppmann, B. Balle, M. Raykova, J. Doerner, S. Zahur, and D. Evans, "Privacy-preserving distributed linear regression on high-dimensional data," *Proceedings on Privacy Enhancing Technologies*, vol. 4, pp. 248–267, 2017.

[21] P. Mohassel and Y. Zhang, "Secureml: A system for scalable privacy-preserving machine learning," in *2017 IEEE symposium on security and privacy*. IEEE, 2017, pp. 19–38.

[22] P. Mohassel and P. Rindal, "ABY^3: A mixed protocol framework for machine learning," in *2018 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2018, pp. 35–52.

[23] T. Araki, J. Furukawa, Y. Lindell, A. Nof, and K. Ohara, "High-throughput semi-honest secure three-party computation with an honest majority," in *2016 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2016, pp. 805–817.

[24] P.-A. Fouque, J. Stern, and G.-J. Wackers, "Cryptocomputing with rationals," in *International Conference on Financial Cryptography*. Springer, 2002, pp. 136–146.

[25] L. Jiang, C. Xu, X. Wang, and C. Lin, "Statistical learning based fully homomorphic encryption on encrypted data," *Soft Computing*, vol. 21, pp. 7473–7483, 2017.

[26] N. Dowlin, R. Gilad-Bachrach, K. Laine, K. Lauter, M. Naehrig, and J. Wernsing, "Manual for using homomorphic encryption for bioinformatics," *Proceedings of the IEEE*, vol. 105, no. 3, pp. 552–567, 2017.
[27] M. Hirt and K. Sako, "Efficient receipt-free voting based on homomorphic encryption," in *2000 International Conference on the Theory and Application of Cryptographic Techniques*, vol. 1807. Springer, 2000, pp. 539–556.
[28] M. Roughan and Y. Zhang, "Secure distributed data-mining and its application to large-scale network measurements," *ACM SIGCOMM Computer Communication Review*, vol. 36, no. 1, pp. 7–14, 2006.
[29] C. Clifton, M. Kantarcioglu, J. Vaidya, X. Lin, and M. Y. Zhu, "Tools for privacy preserving distributed data mining," *ACM Sigkdd Explorations Newsletter*, vol. 4, no. 2, pp. 28–34, 2002.
[30] M. Naehrig, K. Lauter, and V. Vaikuntanathan, "Can homomorphic encryption be practical?" in *3rd ACM workshop on Cloud computing security workshop*. ACM, 2011, pp. 113–124.
[31] F. Emekçi, O. D. Sahin, D. Agrawal, and A. El Abbadi, "Privacy preserving decision tree learning over multiple parties," *Data & Knowledge Engineering*, vol. 63, no. 2, pp. 348–361, 2007.
[32] A. Shamir, "How to share a secret," *Communications of the ACM*, vol. 22, no. 11, pp. 612–613, 1979.
[33] J.-M. Bohli, W. Li, and J. Seedorf, "Assisting server for secure multiparty computation," in *6th IFIP International Workshop on Information Security Theory and Practice*. Springer, 2012, pp. 144–159.
[34] T. Nishide and K. Ohta, "Multiparty computation for interval, equality, and comparison without bit-decomposition protocol," in *10th International Conference on Practice and Theory in Public-Key Cryptography*. Springer, 2007, pp. 343–360.
[35] S. Dolev, P. Gupta, Y. Li, S. Mehrotra, and S. Sharma, "Privacy-preserving secret shared computations using mapreduce," *IEEE Transactions on Dependable and Secure Computing*, vol. 18, no. 4, pp. 1645–1666, 2019.
[36] S. Dolev, N. Gilboa, and X. Li, "Accumulating automata and cascaded equations automata for communicationless information theoretically secure multi-party computation: Extended abstract," in *3rd International Workshop on Security in Cloud Computing*. ACM, 2015, pp. 21–29.
[37] G. Guo, Y. Zhu, E. Chen, R. Yu, L. Zhang, K. Lv, and R. Feng, "Efficient multiparty fully homomorphic encryption with computation fairness and error detection in privacy preserving multisource data mining," *IEEE Transactions on Reliability*, pp. 1–19, 2023.