mathematics

MDPI

Article

Cloud-Assisted Private Set Intersection via Multi-Key Fully Homomorphic Encryption

Cunqun Fan [1,2], Peiheng Jia [3], Manyun Lin [1,2], Lan Wei [1,2,*], Peng Guo [1,2], Xiangang Zhao [1,2] and Ximeng Liu [4]

[1] Key Laboratory of Radiometric Calibration and Validation for Environmental Satellites, National Satellite Meteorological Center (National Center for Space Weather), China Meteorological Administration, Beijing 100081, China
[2] Innovation Center for FengYun Meteorological Satellite (FYSIC), Beijing 100081, China
[3] School of Mathematics and Computer Science, Shanxi Normal University, Taiyuan 030031, China
[4] College of Computer and Data Science, Fuzhou University, Fuzhou 350108, China
* Correspondence: weilan@cma.cn

Abstract: With the development of cloud computing and big data, secure multi-party computation, which can collaborate with multiple parties to deal with a large number of transactions, plays an important role in protecting privacy. Private set intersection (PSI), a form of multi-party secure computation, is a formidable cryptographic technique that allows the sender and the receiver to calculate their intersection and not reveal any more information. As the data volume increases and more application scenarios emerge, PSI with multiple participants is increasingly needed. Homomorphic encryption is an encryption algorithm designed to perform a mathematical-style operation on encrypted data, where the decryption result of the operation is the same as the result calculated using unencrypted data. In this paper, we present a cloud-assisted multi-key PSI (CMPSI) system that uses fully homomorphic encryption over the torus (TFHE) encryption scheme to encrypt the data of the participants and that uses a cloud server to assist the computation. Specifically, we design some TFHE-based secure computation protocols and build a single cloud server-based private set intersection system that can support multiple users. Moreover, security analysis and performance evaluation show that our system is feasible. The scheme has a smaller communication overhead compared to existing schemes.

Keywords: private set intersection; homomorphic encryption; multi-key TFHE; cloud computing; privacy protection

MSC: 68U99; 68U99; 68T09; 68Q06

Citation: Fan, C.; Jia, P.; Lin, M.; Wei, L.; Guo, P.; Zhao, X.; Liu, X. Cloud-Assisted Private Set Intersection via Multi-Key Fully Homomorphic Encryption. *Mathematics* **2023**, *11*, 1784. https://doi.org/10.3390/math11081784

Academic Editor: Antanas Cenys

Received: 21 March 2023
Revised: 4 April 2023
Accepted: 4 April 2023
Published: 8 April 2023

1. Introduction

With the rapid growth of data in the Internet era, the demand for data storage and computing capacity in various fields far exceeds the capacity of their own devices. To solve this problem, cloud computing has been proposed. Cloud computing is generally defined as an internet-based computing method. In this way, the shared software and hardware information and resources can be provided to various terminals and other devices of the computer as required. Cloud computing technology can transmit various information to the Internet and store and calculate data, and users can view the calculation results and data information. However, current security issues in the context of cloud computing are more prominent [1]. Data security issues in cloud computing mainly include storage data security, computing data security, and transmission data security. When users store data on the cloud server, the cloud server will obtain the users' data first, but the abnormal use of malicious users can also cause a risk of data leakage. In the process of cloud server computing, the cloud server will know the calculation results and additional data. This information that should only be known by users also has a risk of leakage. In addition,

data theft can easily occur during data transmission, and user data can show problems of theft and tampering [2].

Private set intersection (PSI), as an interactive encryption protocol, calculates the intersection of two data owners' data and returns it to one of them. We generally refer to the party receiving the data as the receiver and the party receiving nothing as the sender. It is important and necessary to protect the privacy of the set in computing, especially when the information in the set is important private information such as the customer transaction information of a bank or the address book of a user. With the concerted efforts of many researchers, PSI technology has developed rapidly, and more and more efficient solutions have been proposed [3–15]. After several years of development, PSI technology has been applied to the fields of internet of vehicles [16], profile matching [17], and private contact search [18]. In the current situation where the data volume is large and scattered in the hands of different participants, PSI technology can well balance the relationship between privacy and information sharing. Leveraging the storage and computing power of cloud servers allows PSI protocols to compute larger datasets, but current cloud-assisted PSI schemes suffer from information leakage [19] or large communication overhead [20].

Fully homomorphic encryption (FHE) refers to the computation of data that has been homomorphically encrypted, and the computed decryption result is the same as that obtained by the same computation for unencrypted data. The concept of FHE has been proposed as early as the late 1970s, but it has only started to develop rapidly in the last two decades. The development of fully homomorphic encryption is generally divided into three stages. In 2009, the first generation of fully homomorphic encryption started to develop, and Gentry constructed the first fully homomorphic encryption scheme [21]. The scheme first constructs a somewhat homomorphic encryption (SHE) scheme that can homomorphically compute circuits of a certain depth, then compresses and decrypts the circuits and performs bootstrapping operations in an orderly manner, and finally obtains a scheme that can homomorphically compute arbitrary circuits. The second generation of fully homomorphic encryption schemes arose in 2011 when Brakerski and Vaikuntanathan implemented FHE for the first time under the LWE assumption using linearization and modulo conversion [22] and implemented FHE under the RLWE assumption [23]. These schemes do not require compression and decryption circuits, and the security and efficiency are greatly improved. In 2013, the third generation of fully homomorphic encryption schemes was born, and Gentry et al. for the first time designed a fully homomorphic encryption scheme, Gentry–Sahai–Waters (GSW), that does not require the computation of a key using the approximate eigenvector technique [24].

There are two broad categories of fully homomorphic algorithms, the BGV [25] scheme proposed by Professor Brakerski of Stanford University, Research Fellow Gentry of IBM, and Professor Vaikuntanathan of the University of Toronto, and the GSW [24] scheme proposed by Gentry of IBM, Sahai of the University of California and Waters of the University of Austin. Fully homomorphic encryption over toru (TFHE) [26] is an improvement of the GSW scheme with higher efficiency. TFHE can accomplish fast comparisons, supports arbitrary boolean circuits, and allows fast bootstrapping to reduce the noise due to ciphertext computation. In previous studies, the BGV scheme has been used to focus on the unbalanced privacy aggregation scenario [27–29]. Unlike previous works, this paper uses the TFHE encryption scheme for the first time to implement privacy-seeking protocol based on cloud computing. At a high level, our contributions can be summarized as follows:

- We have designed a series of security sub-protocols for the MKTFHE cryptosystem, including some basic circuit gate operations and security comparison protocols.
- We have built a cloud-assisted multi-key private set intersection (CMPSI) system based on a single cloud server. Our system can prevent collusion attacks between servers and participants.
- We strictly prove the security of the proposed CMPSI system under the semi-honest model.

- We have conducted extensive experimental evaluation on the performance of the scheme, which proves that our scheme has greatly reduced the communication cost of the participants.

The rest of the paper is organized as follows. In Section 2, we describe the related work of private set intersection. In Section 3, we provide the preliminaries. Section 4 details the system model, threat model, and design goals. Section 5 elaborates on the cryptographic protocol for the private set intersection. Section 6 analyzes the security of our proposed protocols. Section 7 conducts a series of experimental comparisons. Finally, Section 8 concludes this paper.

2. Related Work

PSI was first proposed by Freedman et al. [30], who transformed the element comparison problem into the polynomial root problem and realized PSI through multiplicative homomorphic encryption. However, when the polynomial order is large, it will lead to a costly exponential computation of the homomorphic encryption. In recent years, many researchers have intensively studied the PSI problem, and many PSI protocols with high efficiency and low communication overhead have emerged. PSI computing protocols are mainly divided into two categories according to whether there is a third party, namely, the traditional PSI computing protocol based on public key encryption, obfuscation circuit [31–33] and inadvertent transmission [34] technology and the cloud-assisted PSI computing protocol that uses cloud servers to complete computing.

Traditional PSI computing protocols rely on a series of basic cryptography technologies for computing. These basic cryptography technologies are mainly divided into PSI based on public key encryption mechanism, PSI based on obfuscation circuit, and PSI based on inadvertent transmission. The PSI calculation protocol proposed by Freedman et al. [34] is based on the public key encryption mechanism. This scheme represents the elements in the set as the roots of polynomials and uses polynomials to calculate the intersection. However, the cost of calculation will become large with the increase in the order of polynomials. Hazay et al. also improved the article [30] and adopted the bit commitment protocol to prevent the scenario of inconsistent input data on the server [35], so that the PSI protocol can be applied to the protocol of malicious adversaries. In 2012, Huang et al. first proposed PSI computing protocols based on obfuscated circuits [36], which are Bitwise-AND (BWA), Pairwise-Comparisons (PWC), and Sort-Compare-Suffle (SCS) protocols. In 2013, the PSI protocol proposed by Dong et al. [37] used OT technology for the first time. The author used OT technology to ensure the security of the protocol. Pinkas et al. [38] proposed a new PSI protocol based on Hash and random OT protocols and optimized the SCS protocol in [36]. The computational efficiency of the protocol was greatly improved, and the complexity of the algorithm was also reduced. Based on the article [34], Freedman et al. further optimized and improved their scheme in 2014 [39]. Specifically, the scheme uses different hash functions for the client and server when mapping the set elements. In 2018, Pinkas et al. realized PSI based on unintentional pseudorandom function [40] through the circuit. In 2020, Pinkas et al. [12] constructed a PSI protocol with malicious security based on the protocols [41] in the literature. The traditional PSI does not need the assistance of a third party, but in the application, the participants are generally resource-constrained users, who are insufficient in providing sufficient data storage and computing power.

With the development of cloud computing, the PSI protocol based on cloud servers began to develop. The cloud-assisted PSI scheme provides a new optimization method for the existing PSI scheme by the excellent storage and computing capabilities of the cloud server. The cloud-assisted PSI uses the third-party cloud computing framework to complete the calculation and uses the storage and computing resources of the cloud server to enable the protocol to calculate large-scale datasets. Kerschbaum [42] implemented the anti-collusion outsourcing PSI protocol through two single functions, but the method has the risk of brute force cracking. Then, Kerschbaum [43] proposed another kind of cloud-assisted PSI using bloom filter and homomorphic encryption. Liu et al. [19] proposed a

relatively simple PSI protocol, but it can disclose the cardinality of set intersection. Abadi et al. [44] implemented the PSI protocol using homomorphic encryption and polynomial interpolation in 2015. This protocol outsources the collection of clients to a third-party server to perform infinite PSI operations. Based on this work, a verifiable cloud outsourcing PSI protocol [45] is proposed to ensure the privacy and integrity of data. Ali et al. [46] proposed an attribute-based private set intersection scheme. The cloud server can calculate the corresponding access rights of the participants. The PSI protocol based on the cloud server can use the computing and storage capabilities of the cloud server, but it has produced the privacy disclosure problem of data outsourcing, and the excessive cost of users in the operation of the protocol is another problem that needs to be solved. Table 1 shows the comparison between our scheme and the existing scheme.

Table 1. Comparison with existing schemes.

	CMPSI	[46]	[47]	[48]	[19]	[42]	[43]
The year	2023	2020	2019	2014	2014	2012	2012
Private against the CSP	52	52	52	52	56	52	52
PSI computation authorization	52	52	52	52	56	52	52
Supports multiple user queries	52	52	52	52	56	56	56
Participants can go offline after uploading data	52	52	56	56	56	56	56
CSP can collude with participants	52	56	56	56	56	56	56

3. Preliminaries

In this section, we first introduce the concept of private set intersection and have an example to better understand the concept. Then, we introduce the cryptosystem MKTFHE used in our system and present the algorithm as an example of a NAND gate. Table 2 lists some of the symbols used in this paper.

Table 2. Notation used.

Notations	**Definition**
λ	Security parameter
$\mathbb{Z}$	Integer set
$\mathbb{T}$	(R)LWE over the real torus
s_i	Private key of participant i
$(\mathbf{PK}_i, \mathbf{BK}_i, \mathbf{KS}_i)$	Public key set of participant i
$[\![x]\!]_{s_i}$	Encrypted data x under s_i
$\mathbf{MKHE_{NAND}}$	NAND gate in multi-key TFHE
CMPSI	Cloud-assisted multi-party private set intersection

3.1. Private Set Intersection

PSI allows two parties holding sets to compare encrypted versions of these sets to compute the intersection. Let the two parties holding the sets be sender X and receiver Y. The sender and receiver hold datasets of size N_x and N_y respectively, each with a number of bits σ. In a basic PSI protocol, receiver Y encrypts its own dataset and sends it to sender X. For each of Y's data, sender X calculates the homomorphic product of the difference with all of its own terms and sends the result to receiver Y. Y decrypts the result of X's calculation and obtains the final intersection information. The result of the calculation is sent to the receiver Y. Y decrypts the result of X's computation and obtains the final intersection information. The basic PSI protocol is shown in Figure 1.

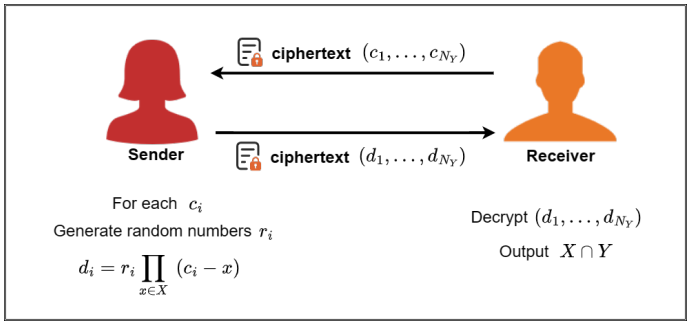

Figure 1. Basic PSI protocol.

In the scheme of this paper, the storage of data and the computation are performed on the cloud server. We construct a new PSI scheme using fully homomorphic encryption. Both the sender and the receiver encrypt the data locally and then send it to the cloud server. Suppose that the sender has encrypted data $a_1, \ldots, a_{N_X}$ and the receiver has encrypted data $b_1, \ldots, b_{N_Y}$. Both parties send their encrypted data to the cloud server. On the cloud server, for each data b_i of the receiver, $c_i = \prod_{0<j\leq N_x}(b_i - a_j)$ is computed. c_i is a Boolean value that represents whether the data b_i of the receiver are in the sender X or not. Figure 2 shows the handshake model of this scheme.

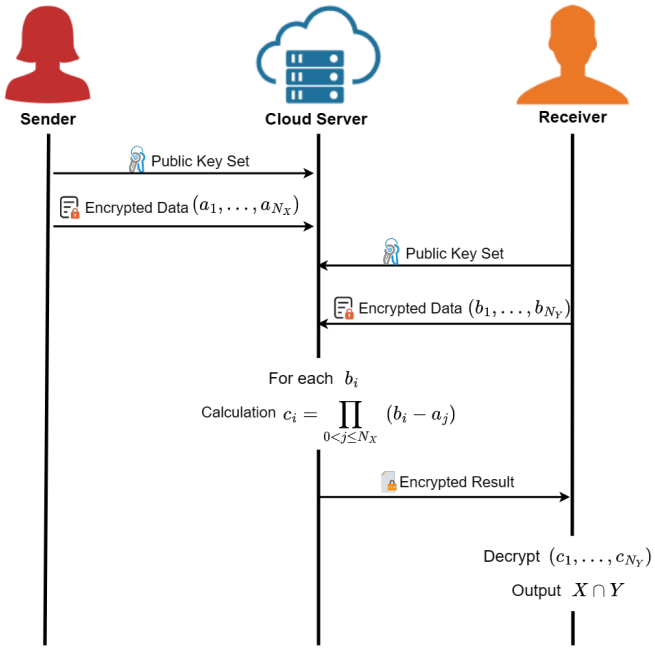

Figure 2. Handshake model.

3.2. MKTFHE Cryptosystem

Homomorphic encryption is the computation of the encrypted data to obtain the encrypted computational result, and the result of the decryption of the obtained encryption result is the same as the result obtained by performing the same operations on the unencrypted plaintext. Fully homomorphic encryption [24,25] is a homomorphic encryption that can satisfy both additive and multiplicative operations. Fully homomorphic encryption over the toru (TFHE) [26] is a type of fully homomorphic encryption that can accomplish fast comparisons and support operations on arbitrary Boolean circuits.TFHE differs from other FHE schemes in that it can be fast bootstrapping to reduce noise during ciphertext operations. In this paper, we use multi-key TFHE [49] to meet the needs of our system. MKTFHE is a multi-key version of TFHE that can compute Boolean circuits on ciphertexts encrypted under different keys, and then performs bootstrapped to refresh the noise as

each binary gate is computed. However, the MKTFHE library only implements multi-key homomorphic NAND gates, which cannot meet the needs of our system. The following describes the five components of MKTFHE and gives an example of the homomorphic computation process with a multi-key homomorphic NAND gate.

1. $\mathbf{Setup}(1^\lambda)$: Takes as input the security parameter λ and returns the public parameter $\mathbf{pp}^{\mathbf{MKTFHE}}$.
   (a) Run $\mathbf{LWE.Setup}(1^\lambda)$ to generate the LWE parameter $\mathbf{pp}^{\mathbf{LWE}} = (n, \chi, \alpha, B', d')$. In the LWE parameters, n is the dimension of the LWE secret, χ is the key distribution of the LWE secret, α is the error rate, B' is the decomposition basis, and d' is the dimension of the key transformation gadget vector. We use the key-switching gadget vector $g' = \left(B'^{-1}, \ldots, B'^{-d'}\right)$.
   (b) Run $\mathbf{RLWE.Setup}(1^\lambda)$ to generate the RLWE parameter $\mathbf{pp}^{\mathbf{RLWE}} = (N, \psi, B, d, \mathbf{a})$. We define N as the dimension of RLWE secret (a power of 2), ψ as the distribution of RLWE secret over R and with error rate α, $B \geq 2$ as an integer base, decomposition dimension d, and gadget vector $g = \left(B^{-1}, \ldots, B^{-d}\right)$. $\mathbf{a}$ is a uniformly distributed sample over distribution T^d.
   (c) Returns the generated public parameter $\mathbf{pp}^{\mathbf{MKTFHE}} = \left(\mathbf{pp}^{\mathbf{LWE}}, \mathbf{pp}^{\mathbf{RLWE}}\right)$.
2. $\mathbf{KeyGen}(\mathbf{pp}^{\mathbf{MKTFHE}})$: Each participant generates its keys independently. Take the public parameter $\mathbf{pp}^{\mathbf{MKTFHE}}$ as input and return the key $\mathbf{s}_i$ and the public key set $(\mathbf{PK}_i, \mathbf{BK}_i, \mathbf{KS}_i)$.
   (a) Generate the LWE secret $\mathbf{s}_i \leftarrow \mathbf{LWE.KeyGen}()$. This step is only for sampling the key from distribution χ.
   (b) Run $(z_i, \mathbf{b}_i) \leftarrow \mathbf{RLWE.KeyGen}()$, and set the public key to $\mathbf{PK}_i = \mathbf{b}_i$. Sample z from distribution ψ, and then, set $\mathbf{z} = (1, z)$. Take an error vector $\mathbf{e}$ from D_α^d and calculate the public key $\mathbf{b} = -z \cdot \mathbf{a} + \mathbf{e} (\mathrm{mod} 1)$. For $z_i = z_{1,0} + z_{i,1}X + \ldots + z_{i,N-1}X^{N-1}$, note $\mathbf{z}_i^* = (z_{i,0}, -z_{i,N-1}, \ldots, z_{i,1}) \in \mathbb{Z}^N$.
   (c) For $j \in [n]$, generate $\left(\mathbf{d}_{i,j}, \mathbf{F}_{i,j}\right) \leftarrow \mathbf{RLWE.UniEnc}\left(s_{i,j}, z_i\right)$, this step is to encrypt the LWE secret using the RLWE secret. In addition, set the bootstrap key to $\mathbf{BK}_i = \left\{\left(\mathbf{d}_{i,j}, \mathbf{F}_{i,j}\right)\right\}_{j \in [n]}$. Taking a random value r from ψ, one can think of $\mathbf{d}$ as the LWE key s under the encryption of the random value r and $\mathbf{F}$ as the random value r under the encryption of the RLWE key z.
   (d) Generate a key conversion key $\mathbf{KS} \leftarrow \mathbf{LWE.KSGen}\left(z_i^*, \mathbf{s}_i\right)$, capable of converting an LWE ciphertext corresponding to $\mathbf{t} \in \mathbb{Z}^N$ into another LWE ciphertext for the same message under $\mathbf{s} \in \mathbb{Z}^N$ encryption.
   (e) Returns key $\mathbf{s}_i$, a triple $(\mathbf{PK}_i, \mathbf{BK}_i, \mathbf{KS}_i)$ of public keys, public key, bootstrap key and key transformation key, respectively.
3. $\mathbf{Enc}(m)$: The data m to be encrypted are taken as input, and return TLWE ciphertext $[\![m]\!] = (b, \mathbf{a}) \in \mathbb{T}^{n+1}$ satisfies $b + \langle \mathbf{a}, \mathbf{s} \rangle \approx \frac{1}{4}m (\mathrm{mod} 1)$.
   (a) Using standard LWE encryption, uniformly sample from $\mathbb{T}^n$ to obtain $\mathbf{a}$ as the mask and sample from D_α to obtain e as the error.
   (b) Output ciphertext $[\![m]\!] = (b, \mathbf{a}) \in \mathbb{T}^{n+1}$, where $b + \langle \mathbf{a}, \mathbf{s} \rangle \approx \frac{1}{4}m (\mathrm{mod} 1)$.
4. $\mathbf{Dec}([\![m]\!], \{\mathbf{s}_i\}_{i \in [k]})$: Takes as input the TLWE ciphertext $[\![m]\!] = (b, \mathbf{a}_1, \ldots, \mathbf{a}_k) \in \mathbb{T}^{kn+1}$ with a set of keys $(\mathbf{s}_1, \ldots, \mathbf{s}_k)$ and returns the decrypted message m which minimizes $\mid b + \sum_{i=1}^k \langle \mathbf{a}_i, \mathbf{s}_i \rangle - \frac{1}{4}m \mid$.
   (a) Input $[\![m]\!] = (b, \mathbf{a}_1, \ldots, \mathbf{a}_k) \in \mathbb{T}^{kn+1}$ with a set of keys $(\mathbf{s}_1, \ldots, \mathbf{s}_k)$.
   (b) Returns the bit $m \in \{0, 1\}$ that minimizes $\mid b + \sum_{i=1}^k \langle \mathbf{a}_i, \mathbf{s}_i \rangle - \frac{1}{4}m \mid$.
5. $\mathbf{NAND}([\![m_1]\!], [\![m_2]\!], \{(\mathbf{PK}_i, \mathbf{BK}_i, \mathbf{KS}_i)\}_{i \in [k]})$: Takes two TLWE ciphertexts and the public key as input. Expand $[\![m_1]\!] \in \mathbb{T}^{k_1 n+1}$ and $[\![m_2]\!] \in \mathbb{T}^{k_1 n+1}$ to $[\![m_1']\!], [\![m_2']\!] \in \mathbb{T}^{kn+1}$ and evaluate the gate homomorphically on encrypted bits. Then the algorithm evalu-

ates the decryption circuit of the TLWE ciphertext and execute the multi-key switching algorithm. Finally, returning the TLWE ciphertext of the same message under joint key encryption.

(a) Given two ciphertexts $[\![m_1]\!] \in \mathbb{T}^{k_1 n+1}$ and $[\![m_2]\!] \in \mathbb{T}^{k_1 n+1}$, let k be the number of participants, associated with either $[\![m_1]\!]$ or $[\![m_2]\!]$. For a public key set, $\mathbf{PK}_i = \mathbf{b}_i$ represents the public key, $\mathbf{BK}_i = \left\{ \left(\mathbf{d}_{i,j}, \mathbf{F}_{i,j} \right) \right\}_{j \in [n]}$ represents the bootstrap key, and $\mathbf{KS}_i$ represents the key transformation key of the j-th participant. Expand ciphertext $[\![m_1]\!]$ and $[\![m_2]\!]$ to $[\![m_1]\!]'$, $[\![m_2]\!]' \in \mathbb{T}^{kn+1}$, i.e., the same message under joint key $\bar{\mathbf{s}} = (\mathbf{s}_1, \ldots, \mathbf{s}_k) \in \mathbb{Z}^{kn}$ encryption. The process of expansion is the process of rearrangement, and 0 is put into the empty slot. Using the expanded ciphertext to perform the calculations. Only the calculation of NAND gate is supported in the document.

(b) Use the Mux gate to implement the main calculation, for $i \in [k]$, let $\tilde{\mathbf{a}}_i = \left(\tilde{a}_{i,j} \right)_{j \in [n]}$. For $i \in [k]$ and $j \in [n]$, recursively compute $[\![\mathbf{c}]\!] \leftarrow [\![\mathbf{c}]\!] + \mathbf{RLWE.Prod}\left([\![\mathbf{c}]\!] \cdot X^{\tilde{a}_{ij}} - [\![\mathbf{c}]\!], (\mathbf{d}_{i,j}, \mathbf{F}_{i,j}), \{\mathbf{b}_l\}_{l \in [k]} \right)$, where $\mathbf{RLWE.Prod}\left([\![\mathbf{c}]\!], (\mathbf{d}_i, \mathbf{F}_i), \{\mathbf{b}_j\}_{j \in [k]} \right)$ is a hybrid product algorithm that multiplies a single encrypted ciphertext $(\mathbf{d}_i, \mathbf{F}_i)$ by a multi-key RLWE ciphertext $[\![c]\!]$.

(c) For $[\![\mathbf{c}]\!] = (c_0, c_1, \ldots, c_k) \in T^{k+1}$, let b^* be a constant term of c_0 and for $i \in [k]$, let $\mathbf{a}_i^*$ be a vector of coefficients of c_i. Compute the LWE ciphertext $[\![m]\!]^* = \left(b^*, \mathbf{a}_1^*, \ldots, \mathbf{a}_k^* \right) \in \mathbb{T}^{kn+1}$. Finally a multi-key key conversion algorithm is executed and returns the ciphertext $[\![m]\!]'' \leftarrow \mathbf{LWE.MKSwitch}\left([\![m]\!]^*, \{\mathbf{KS}_i\}_{i \in [k]} \right)$, where $\mathbf{LWE.MKSwitch}\left([\![m]\!]^*, \{\mathbf{KS}_i\}_{i \in [k]} \right)$ inputs the expanded ciphertext and a series of key conversion keys, returning the ciphertext of the same message under joint key encryption.

4. System Model and Design Goal

4.1. Problem Formulation

Suppose the receiver Y has a dataset T_Y, and Y wants to know their intersection with other data owners but does not want to expose more information. The data owners encrypt their datasets separately and send them to the cloud server. The cloud server can store this encrypted information but cannot decrypt it. Data receiver Y encrypts its data and uploads it to the cloud server, which executes privacy intersection and obtains the intersection information of dataset T_Y with other datasets. The cloud server computes and returns the cryptographic result to receiver Y. Y decrypts the intersection result and obtains the intersection information. Note that each data owner including the data receiver has their separate key to encrypt the data.

4.2. System Model

In Section 3.1, we mention the flow of the basic PSI protocol, in which the sender interacts directly with the receiver for information. Unlike the basic PSI protocol, our system consists of four entities, which are Parameter Generation Center (PGC), Cloud Server (CS), Data Receiver (DR), and Data Owners (DOs). DO owns its own dataset and is able to let other participants obtain information about the intersection of the dataset but does not want to expose more information. DR wants to query the intersection of its own dataset with the dataset of other participants and does not want to expose more information. Specifically, PGC is responsible for generating public parameters in the system and sending them to other entities. CS can store a large amount of data and has excellent computing resources. DR needs to query the intersection. DOs provide their encrypted data to CS. Note that in our system, the data owners can be multiple participants. The general model of our private set intersection system is shown in Figure 3.

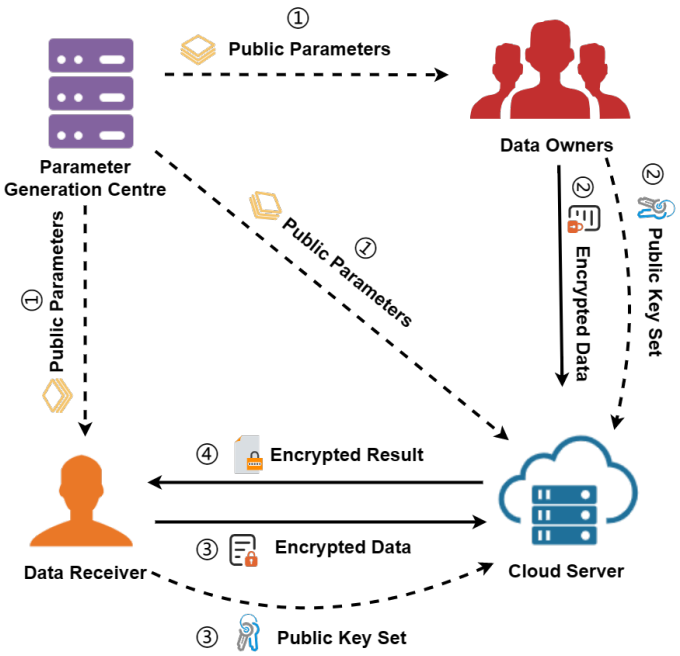

Figure 3. System model.

1. PGC: PGC generates public parameters for our system and sends them to each entity involved in the computation (See ①).
2. CS: CS has huge storage resources to store the encrypted data of the participating parties. At the same time, CS has large enough computing power to satisfy the intersection of the datasets of the participating parties.
3. DR: DR generates its own private key and public key set using public parameters, encrypts its own data using the private key and sends it to CS (See ③), and receives the computation results sent by CS (See ④).
4. DOs: Each DO generates its own private key and public key set using public parameters, encrypts its own dataset using the private key, and sends it to CS (See ②).

Please note that in our system, the participants do not need to be online all the time. Since CS can store the encrypted data, the DOs can go offline after they send their encrypted data to CS. Similarly, DR can be offline after sending data until CS returns the calculation results. In our scheme, DO can be used as DR for frequent item set queries, and the DR can query the intersection information with multiple DOs to achieve multi-user query.

4.3. Threat Model

In our system model, the participating entities are curious but honest individuals. Curious means that the server and the participants try to use existing resources and data to obtain the data of other participants and are curious about the data of other entities; honest means that the server and the participants do not falsify the experimental data and follow the developed protocols to complete the computation. $\mathcal{A}$ is the active adversary we introduce to obtain the real data from other entities. Specifically, $\mathcal{A}$ desires to obtain the real data of DOs and DR. We assume that adversary $\mathcal{A}$ has the following capabilities.

1. $\mathcal{A}$ can obtain all the data that passes through the public channel.
2. $\mathcal{A}$ may collude with CS. Try to obtain the original values of the encrypted data uploaded by DOs and DR.
3. $\mathcal{A}$ may be a DR used to obtain its dataset information, the cryptographic query results returned by the CS, and the encryption and decryption capabilities of the DR.
4. $\mathcal{A}$ may be a DO used to obtain its dataset information and encryption and decryption capabilities.

Note that in our threat model, the attacking adversary $\mathcal{A}$ can be a DR. Since the joint key of multiple participants must be used in decryption to decrypt the computed result of CS, the final decryption result is also not available when $\mathcal{A}$ has only the key of DR. Unlike existing schemes when the attacking adversary $\mathcal{A}$ is a CSP, $\mathcal{A}$ can collude with DR or DO. In our scheme, decryption requires the keys of all participants to perform; thus, CSP colluding with some DR or DO still cannot decrypt the computation results.

4.4. Design Goal

According to the system model and threat model proposed above, the design objectives of this paper are as follows.

1. Data privacy: the original data of DR and the query intersection result as well as the original dataset of DOs cannot be revealed to adversary $\mathcal{A}$.
2. Calculation accuracy: The accuracy of the calculation results of the system cannot be reduced compared with other methods.
3. Low overhead: The time and upload overhead of the calculation cannot be too large compared with other methods.
4. Offline participant: The participant should be able to go offline after encrypting the data and uploading it to ensure the scalability of the system.

5. Cloud-Assisted Multi-Party Private Set Intersection

In this section, we first introduce the initialization of the system. Then, we design the secure computing sub-protocol based on MKTFHE. Finally, we describe our private set intersection scheme.

5.1. System Initialization

Our system can satisfy the DR to query the information of its intersection with multiple participants, and we assume that there is a DR and n DOs. First, PGC generates public parameters for each participant and the cloud server and sends the public parameters to CS, DR, and n DOs. Then, each entity that receives the public parameters generates its own public key set ($\mathbf{PK}$, $\mathbf{BK}$, $\mathbf{KS}$) and private key s based on the public parameters.

5.2. Security Protocol Design

In this paper, four secure computation protocols are proposed to help complete the privacy-seeking intersection, which is a secure AND gate computation protocol ($\mathbf{SC_{AND}}$), secure OR gate computation protocol ($\mathbf{SC_{OR}}$), secure XNOR computation protocol ($\mathbf{SC_{XNOR}}$), and secure comparison protocol ($\mathbf{SCP}$).

5.2.1. Secure AND Gate Computation Protocol

We implement the AND operation between two MKLwe samples. We implement the addition between multi-key Lwe samples (**MKlweAddTo**) to implement this secure computation protocol. Suppose CS has two MKLwe samples *ca* and *cb*: initialize an intermediate sample *temp*, add *ca* and *cb* using **MKlweAddTo** twice, and finally return the result to *res* (Algorithm 1).

Algorithm 1 Secure AND gate computation protocol ($\mathbf{SC_{AND}}$).

Input: MKLwe Sample *ca*, *cb*.
Output: MKLwe Sample *res*.
1: CS initializes *temp* using the public parameter **pp** to hold the intermediate variable LWE sample.
2: $AndConst = modSwitchToTorus32(-1, 8)$
3: $temp \leftarrow \mathbf{MKlweNoiselessTrivial}(AndConst, \mathbf{pp})$
4: $temp \leftarrow \mathbf{MKlweAddTo}(temp + ca)$
5: $res \leftarrow \mathbf{MKlweAddTo}(temp + cb)$

5.2.2. Secure OR Gate Computation Protocol

We implement the OR operation between two MKLwe samples. As with $\mathbf{SC_{AND}}$ above, we use the addition **MKlweAddTo** between multi-key Lwe samples to implement this secure computation protocol. Suppose CS has two MKLwe samples *ca* and *cb*, initialize an intermediate sample *temp*, add *ca* and *cb* using **MKlweAddTo** twice respectively, and finally, return the result to *res* to obtain the result of the OR gate operation between *ca* and *cb* (Algorithm 2).

Algorithm 2 Secure OR gate computation protocol ($\mathbf{SC_{OR}}$).

Input: MKLwe Sample *ca*, *cb*.
Output: MKLwe Sample *res*.
1: CS initializes *temp* using the public parameter **pp** to hold the intermediate variable LWE sample.
2: $ORConst = modSwitchToTorus32(1, 8)$
3: $temp \leftarrow \mathbf{MKlweNoiselessTrivial}(ORConst, \mathbf{pp})$
4: $temp \leftarrow \mathbf{MKlweAddTo}(temp + ca)$
5: $res \leftarrow \mathbf{MKlweAddTo}(temp + cb)$

5.2.3. Secure XNOR Gate Computation Protocol

We implement the XNOR operation between two MKLwe samples. We implement this secure computation protocol using the addition and multiplication of multi-key Lwe samples **MKlweAddMulTo**. Suppose CS has two MKLwe samples *ca* and *cb*: initialize an intermediate sample *temp*, add $2 * ca$ and $2 * cb$ using **MKlweAddMulTo** twice, return the result to *temp* to obtain the XOR gate operation result of *ca* and *cb*, and use the multi-key homomorphic NOT gate $\mathbf{SC_{NOT}}$ once to obtain the XNOR gate operation result. Note that in the cryptographic scheme we use, MKTFHE, the computation of the NOT gate does not require bootstrapping operations; thus, the computation overhead is very small (Algorithm 3).

Algorithm 3 Secure XNOR gate computation protocol ($\mathbf{SC_{XNOR}}$).

Input: MKLwe Sample *ca*, *cb*.
Output: MKLwe Sample *res*.
1: CS initializes *temp* using the public parameter **pp** to hold the intermediate variable LWE sample.
2: $XNORConst = modSwitchToTorus32(1, 8)$
3: $temp \leftarrow \mathbf{MKlweNoiselessTrivial}(XNORConst, \mathbf{pp})$
4: $temp \leftarrow \mathbf{MKlweAddMulTo}(temp + 2 * ca)$
5: $temp \leftarrow \mathbf{MKlweAddMulTo}(temp + 2 * cb)$
6: $res \leftarrow \mathbf{SC_{NOT}}(temp)$

5.2.4. Secure Comparison Protocol

SCP is important in our protocol and is used to determine whether the two input ciphertext vectors are equal or not. Suppose DR has its own encrypted data $[\![\mathbf{x}]\!]_{s_{DR}} = ([\![x_1]\!]_{s_{DR}}, \ldots, [\![x_n]\!]_{s_{DR}})$ sent to CS and DO has its own encrypted data $[\![\mathbf{y}]\!]_{s_{DO}} = ([\![y_1]\!]_{s_{DO}}, \ldots, [\![y_n]\!]_{s_{DO}})$ also sent to CS, where s_{DR} and s_{DO} are the private keys of DR and DO, respectively. For each of $[\![\mathbf{x}]\!]_{s_{DR}} = ([\![x_1]\!]_{s_{DR}}, \ldots, [\![x_n]\!]_{s_{DR}})$ and $[\![\mathbf{y}]\!]_{s_{DO}} = ([\![y_1]\!]_{s_{DO}}, \ldots, [\![y_n]\!]_{s_{DO}})$, the protocol performs $\mathbf{SC_{XNOR}}$ and $\mathbf{SC_{AND}}$ protocols to finally obtain a ciphertext with a Boolean value (Algorithm 4).

Algorithm 4 Secure Comparison Protocol (**SCP**).

Input: Encrypted data vectors $[\![\mathbf{x}]\!]_{s_{DR}} = ([\![x_1]\!]_{s_{DR}}, \ldots, [\![x_n]\!]_{s_{DR}})$,$[\![\mathbf{y}]\!]_{s_{DO}} = ([\![y_1]\!]_{s_{DO}}, \ldots, [\![y_n]\!]_{s_{DO}})$.
Output: Encrypted Boolean values $[\![z]\!]_{s_i}$.
1: CS initializes the intermediate data vector $[\![\mathbf{v}]\!] = ([\![v_1]\!], \ldots, [\![v_n]\!])$ using the public parameter **pp**.
2: **for** $k = 0$ to $n - 1$ **do**
3: $[\![v_k]\!]_{s_i} \leftarrow [\![x_k]\!]_{s_{DR}}$XNOR$[\![y_k]\!]_{s_{DO}}$
4: $[\![z]\!]_{s_i} \leftarrow [\![v_k]\!]_{s_i}AND[\![z]\!]_{s_i}$
5: **end for**

5.3. Private Set Intersection

CMPSI is performed by CS, DR, and DOs working together. Now DR wants to obtain the intersection information of their dataset and DOs dataset. First DOs encrypt their dataset with their own private key s_{DO}, send the encrypted dataset $A_{DO} = \left\{[\![\mathbf{a_1}]\!]_{s_{DO}}, [\![\mathbf{a_2}]\!]_{s_{DO}}, \ldots, [\![\mathbf{a_m}]\!]_{s_{DO}}\right\}$ with the public key set $(\mathbf{PK}_{s_{DO}}, \mathbf{BK}_{s_{DO}}, \mathbf{KS}_{s_{DO}})$ to CS, and then they can go offline. DR encrypts the dataset with its own private key s_{DR} and then sends the encrypted dataset $B_{DR} = \left\{[\![\mathbf{b_1}]\!]_{s_{DR}}, [\![\mathbf{b_2}]\!]_{s_{DR}}, \ldots, [\![\mathbf{a_n}]\!]_{s_{DR}}\right\}$ with its public key set $(\mathbf{PK}_{s_{DR}}, \mathbf{BK}_{s_{DR}}, \mathbf{KS}_{s_{DR}})$ to CS, and then, it can be offline until CS completes the calculation. CS receives the encrypted dataset sent by DOs and DR, saves the data, and performs the secure computation in a secure environment. Finally, DR receives the encryption result calculated by CS and decrypts it using the joint key to obtain the intersection. Let there be m items in the encrypted dataset $A_{DO} = \left\{[\![\mathbf{a_1}]\!]_{s_{DO}}, [\![\mathbf{a_2}]\!]_{s_{DO}}, \ldots, [\![\mathbf{a_m}]\!]_{s_{DO}}\right\}$ of DOs with k Boolean values in each item, and n items in the encrypted dataset $B_{DR} = \left\{[\![\mathbf{b_1}]\!]_{s_{DR}}, [\![\mathbf{b_2}]\!]_{s_{DR}}, \ldots, [\![\mathbf{b_n}]\!]_{s_{DR}}\right\}$ of DR with k Boolean values in each item.

S1(DOs): Each DO encrypts its dataset using its own key s_{DO} generated by the public parameter **pp** issued by PGC and sends it to CS. CS stores the encrypted dataset of all DOs, and for item i of dataset $A_{DO} = \left\{[\![\mathbf{a_1}]\!]_{s_{DO}}, [\![\mathbf{a_2}]\!]_{s_{DO}}, \ldots, [\![\mathbf{a_m}]\!]_{s_{DO}}\right\}$, we have $[\![\mathbf{a_i}]\!]_{s_{DO}} = ([\![a_1]\!]_{s_{DO}}, \ldots, [\![a_k]\!]_{s_{DO}})$.

S2(DR): DR uses the public parameter **pp** to generate its own key s_{DR} to encrypt its dataset and sends it to CS. CS uses DR's encrypted database for secure computation and has $[\![\mathbf{b_j}]\!]_{s_{DR}} = ([\![b_1]\!]_{s_{DR}}, \ldots, [\![b_k]\!]_{s_{DR}})$ for item j of dataset $B_{DR} = \left\{[\![\mathbf{b_1}]\!]_{s_{DR}}, [\![\mathbf{b_2}]\!]_{s_{DR}}, \ldots, [\![\mathbf{b_n}]\!]_{s_{DR}}\right\}$.

S3(CS): CS receives the encrypted data message $B_{DR} = \left\{[\![\mathbf{b_1}]\!]_{s_{DR}}, [\![\mathbf{b_2}]\!]_{s_{DR}}, \ldots, [\![\mathbf{b_n}]\!]_{s_{DR}}\right\}$ from DR and the encrypted data message $A_{DO} = \left\{[\![\mathbf{a_1}]\!]_{s_{DO}}, [\![\mathbf{a_2}]\!]_{s_{DO}}, \ldots, [\![\mathbf{a_m}]\!]_{s_{DO}}\right\}$ from DO. For $j \in \{1, 2, \ldots, n\}$ and $i \in \{1, 2, \ldots, m\}$, each item $[\![\mathbf{b_j}]\!]_{s_{DR}} = ([\![b_1]\!]_{s_{DR}}, \ldots, [\![b_k]\!]_{s_{DR}})$ in $B_{DR} = \left\{[\![\mathbf{b_1}]\!]_{s_{DR}}, [\![\mathbf{b_2}]\!]_{s_{DR}}, \ldots, [\![\mathbf{b_n}]\!]_{s_{DR}}\right\}$ performs **SCP** with each item $[\![\mathbf{a_i}]\!]_{s_{DO}} = ([\![a_1]\!]_{s_{DO}}, \ldots, [\![a_k]\!]_{s_{DO}})$ in $A_{DO} = \left\{[\![\mathbf{a_1}]\!]_{s_{DO}}, [\![\mathbf{a_2}]\!]_{s_{DO}}, \ldots, [\![\mathbf{a_m}]\!]_{s_{DO}}\right\}$, i.e., $\mathbf{SCP}([\![\mathbf{a_i}]\!]_{s_{DO}}, [\![\mathbf{b_j}]\!]_{s_{DR}})$. The result $[\![\mathbf{g_i}]\!]_s = ([\![g_1]\!]_s, \ldots, [\![g_k]\!]_s)$ is obtained as the result of whether the current $[\![\mathbf{b_j}]\!]_{s_{DR}} = ([\![b_1]\!]_{s_{DR}}, \ldots, [\![b_k]\!]_{s_{DR}})$ is the same as each item in $A_{DO} = \left\{[\![\mathbf{a_1}]\!]_{s_{DO}}, [\![\mathbf{a_2}]\!]_{s_{DO}}, \ldots, [\![\mathbf{a_m}]\!]_{s_{DO}}\right\}$.

S4(CS): For each computed $[\![\mathbf{g_i}]\!]_s = ([\![g_1]\!]_s, \ldots, [\![g_k]\!]_s)$, CS runs $\mathbf{SC_{OR}}$ to obtain $[\![\mathbf{c_j}]\!]_s = ([\![c_1]\!]_s, \ldots, [\![c_k]\!]_s)$. $[\![\mathbf{c_j}]\!]_s = ([\![c_1]\!]_s, \ldots, [\![c_k]\!]_s)$ is a cryptographic Boolean value indicating whether each item in $B_{DR} = \left\{[\![\mathbf{b_1}]\!]_{s_{DR}}, [\![\mathbf{b_2}]\!]_{s_{DR}}, \ldots, [\![\mathbf{b_n}]\!]_{s_{DR}}\right\}$ exists in $A_{DO} = \left\{[\![\mathbf{a_1}]\!]_{s_{DO}}, [\![\mathbf{a_2}]\!]_{s_{DO}}, \ldots, [\![\mathbf{a_m}]\!]_{s_{DO}}\right\}$. A value of 1 means it exists and 0 means it does not.

S5 (CS): For $j \in \{1, 2, \ldots, n\}$, execute S4, and send the calculated result $C = \{[\![\mathbf{c_1}]\!]_s, [\![\mathbf{c_2}]\!]_s, \ldots, [\![\mathbf{c_n}]\!]_s\}$ to DR.

S6 (DR): Receive the calculation result from $C = \{[\![\mathbf{c_1}]\!]_s, [\![\mathbf{c_2}]\!]_s, \ldots, [\![\mathbf{c_n}]\!]_s\}$ sent by CS and decrypt it using the joint key to obtain the result.

Please note that in our PSI scheme, the dense state computation is performed by FHE cryptography. All the calculations are performed on the cloud server, and the data on the cloud server are all cryptographic data, so that the privacy of the participants is protected. During the calculation process, the DR does not obtain any information other than its own information and the query result. The DOs do not obtain any information other than their own information and do not expose their information to other participants. The result of the CS calculation is in cryptographic form and cannot be decrypted by the participants except by the DR, which protects the privacy of the calculation result.

6. Security Analysis

In this section, we prove that our scheme is secure under a semi-honest model. We will prove the security of the MKTFHE cryptosystem, $\mathbf{SC_{AND}}$, $\mathbf{SC_{OR}}$, $\mathbf{SC_{XOR}}$, **SCP** and PSI schemes separately. We first present the security of the semi-honest model below.

Definition 1 (Security of the semi-honest model)**.** *According to protocol π, let a_i be the input of participant P_i and b_i be the output of P_i.* $\mathrm{REAL}_i^{\Pi}(\pi)$ *is the viewpoint of P_i when protocol π is actually executed.* $\mathrm{IDEAL}_i^{\Pi}(\pi)$ *is the viewpoint of P_i, simulated by a_i and b_i, executed in the ideal world of protocol π. If* $\mathrm{REAL}_i^{\Pi}(\pi)$ *is computationally indistinguishable from* $\mathrm{IDEAL}_i^{\Pi}(\pi)$*, then protocol π is secure in the semi-fair model [50].*

Note that in our protocols, the execution image usually consists of the exchanged data and the information that can be computed from these data. It follows from Definition (1) that when proving the security of these protocols, the image we simulate should be indistinguishable from the actual execution image when we compute it.

6.1. Security of MKTFHE Cryptosystem

Privacy of **LWE Assumption**: The j-th component $\mathbf{K}_j$ of a key-switching key $KS = \{K_j\}_{j\in[N]}$ from $\mathbf{t} \in \mathbb{Z}^N$ to $\mathbf{s} \in \mathbb{Z}^N$ is generated by adding $t_j \cdot \mathbf{g}'$ to the first column of the $\mathbb{T}^{d'\times(n+1)}$ matrix, the rows of which are instances of LWE under the secret $\mathbf{s}$. Therefore, $\mathrm{KS} \leftarrow \mathbf{LWE.KSGen}(\mathbf{t}, \mathbf{s})$ is computationally indistinguishable from a uniform distribution over $(\mathbb{T}^{d'\times(n+1)})^N$ where LWE assumes a parameter of (n, χ, β) and s is sampled according to χ.

Privacy of **RLWE Assumption**: Under the assumption that the parameter is (N, ψ, α), a uniform distribution over $T^{d\times 5}$ is computationally indistinguishable from the distribution $\mathcal{D}_0 = \{(\mathbf{a}, \mathbf{b}, \mathbf{d}, \mathbf{F}) : \mathbf{pp^{RLWE}} \leftarrow \mathbf{RLWE} \cdot \mathbf{Setup}(1^\lambda), (z, \mathbf{b}) \leftarrow \mathbf{RLWE.KeyGen}(), (\mathbf{d}, \mathbf{F}) \leftarrow \mathbf{RLWE} \cdot \mathbf{UniEnc}(\mu, z)\}$ for any $\mu \in R$. We consider the following distribution: First, we transform $\mathbf{F} = [\mathbf{f}_0 \mid \mathbf{f}_1]$ and $(\mathbf{b}, \mathbf{a})$ into independent uniform distributions of $T^{d\times 2}$ using the RLWE assumption of a secret z. Therefore, $\mathcal{D}_0$ is indistinguishable from $\mathcal{D}_1$ in terms of calculation. $\mathcal{D}_1 = \{(\mathbf{a}, \mathbf{b}, \mathbf{d}, \mathbf{F}) : \mathbf{a}, \mathbf{b} \leftarrow U\left(T^d\right), \mathbf{F} \leftarrow U\left(T^{d\times 2}\right), r \leftarrow \psi, \mathbf{e}_1 \leftarrow D_\alpha^d, \mathbf{d} = r \cdot \mathbf{a} + \mu \cdot \mathbf{g} + \mathbf{e}_1 \pmod 1\}$.Then, $\mathbf{d}$ is made uniformly distributed using the RLWE assumption with a secret of r. Therefore, $\mathcal{D}_1$ is indistinguishable from the distribution $\mathcal{D}_2$. $\mathcal{D}_2 = \left\{(\mathbf{a}, \mathbf{b}, \mathbf{d}, \mathbf{F}) : \mathbf{a}, \mathbf{b}, \mathbf{d} \leftarrow U\left(T^d\right), \mathbf{F} \leftarrow U\left(T^{d\times 2}\right)\right\}$. Since $\mathcal{D}_2$ is independent from μ, our RLWE scheme is semantically private.

In summary, under the (R)LWE assumption, our cryptosystem is semantically private; thus, we can appropriately choose parameters $\mathbf{pp^{LWE}}$ and $\mathbf{pp^{RLWE}}$ to achieve a security level of at least λ-bit.

6.2. Security of Secure Computing Protocols

In this section, we demonstrate the security of our secure computing subprotocols, including $\mathbf{SC_{AND}}$, $\mathbf{SC_{OR}}$, $\mathbf{SC_{XOR}}$ and **SCP**.

Theorem 1. *The* $\mathbf{SC_{AND}}$ *proposed is secure under the semi-honest model.*

Proof of Theorem 1. We use $\mathrm{REAL}_{CS}^{\Pi}(\mathrm{SC_{AND}})$ to denote the execution view in the real world of the of CS, where it is specified as $\mathrm{REAL}_{CS}^{\Pi}(\mathrm{SC_{AND}}) = \{[\![ca]\!], [\![cb]\!], [\![AndConst]\!], [\![temp]\!], [\![res]\!]\}$. $[\![AndConst]\!]$ is obtained from $[\![-1]\!]$ and $[\![8]\!]$ by *modSwitchToTorus*32. $[\![temp]\!]$ is obtained from $[\![AndConst]\!]$ and $[\![ca]\!]$ by **MKlweAddTo** and **MKlweNoiselessTrival**. We assume that $\mathrm{IDEAL}_{CS}^{\Pi}(\mathrm{SC_{AND}}) = \{[\![ca']\!], [\![cb']\!], [\![temp']\!], [\![res']\!], [\![AndConst']\!]\}$ is the execution view of the simulation in the ideal world, where $[\![ca']\!]$, $[\![cb']\!]$, $[\![temp']\!]$, $[\![res']\!]$ and $[\![AndConst']\!]$ are chosen randomly from $\mathbb{T}^{n+1}$. The semantic privacy of our encryption scheme makes $[\![ca]\!]$, $[\![cb]\!]$, $[\![temp]\!]$ and $[\![AndConst]\!]$ computationally indistinguishable from $[\![ca']\!]$, $[\![cb']\!]$, $[\![temp']\!]$ and $[\![AndConst']\!]$ respectively. In addition, $[\![res]\!]$ is computationally indistinguishable from $[\![temp']\!]$ and $[\![AndConst']\!]$ respectively. Thus, it can be concluded that $\mathrm{REAL}_{CS}^{\Pi}(\mathrm{SC_{AND}})$ and $\mathrm{IDEAL}_{CS}^{\Pi}(\mathrm{SC_{AND}})$ are computationally indistinguishable. We can obtain that $\mathbf{SC_{AND}}$ is secure under the semi-honest model. □

Theorem 2. *The* $\mathbf{SC_{OR}}$ *proposed is secure under the semi-honest model.*

Proof of Theorem 2. We use $\mathrm{REAL}_{CS}^{\Pi}(\mathrm{SC_{OR}})$ to denote the execution view in the real world of CS, where it is specified as $\mathrm{REAL}_{CS}^{\Pi}(\mathrm{SC_{OR}}) = \{[\![ca]\!], [\![cb]\!], [\![temp]\!], [\![ORConst]\!], [\![res]\!]\}$. $[\![ORConst]\!]$ is obtained from $[\![1]\!]$ and $[\![8]\!]$ by *modSwitchToTorus*32. $[\![temp]\!]$ is obtained from $[\![ca]\!]$ and $[\![ORConst]\!]$ by **MKlweNoiselessTrivial** and **MKlweAddTo**. $[\![res]\!]$ is obtained from $[\![temp]\!]$ and $[\![cb]\!]$ by **MKlweAddTo**. We assume that $\mathrm{IDEAL}_{CS}^{\Pi}(\mathrm{SC_{OR}}) = \{[\![ca']\!], [\![cb']\!], [\![temp']\!], [\![ORConst']\!], [\![res']\!]\}$ is the execution view of the simulation in the ideal world, where $[\![ca']\!]$, $[\![cb']\!]$, $[\![temp']\!]$, $[\![ORConst']\!]$ and $[\![res']\!]$ are chosen randomly from $\mathbb{T}^{n+1}$. The semantic privacy of our encryption scheme makes $[\![ca]\!]$ and $[\![cb]\!]$ computationally indistinguishable from $[\![ca']\!]$, $[\![cb']\!]$, $[\![temp']\!]$ and $[\![ORConst']\!]$, respectively. In addition, $[\![res]\!]$ is computationally indistinguishable from $[\![ca']\!]$, $[\![cb']\!]$, $[\![temp']\!]$ and $[\![ORConst']\!]$ respectively. Thus, it can be concluded that $\mathrm{REAL}_{CS}^{\Pi}(\mathrm{SC_{OR}})$ and $\mathrm{IDEAL}_{CS}^{\Pi}(\mathrm{SC_{OR}})$ are computationally indistinguishable. We can obtain that $\mathbf{SC_{OR}}$ is secure under the semi-honest model. □

Theorem 3. *The* $\mathbf{SC_{XOR}}$ *proposed is secure under the semi-honest model.*

Proof of Theorem 3. Since the design ideas of $\mathbf{SC_{AND}}$ and $\mathbf{SC_{OR}}$ are similar, we can prove the theorem based on Theorem (1). □

Theorem 4. *The* **SCP** *proposed is secure under the semi-honest model.*

Proof of Theorem 4. We use $\mathrm{REAL}_{CS}^{\Pi}(\mathrm{SCP})$ to denote the execution view in the real world of the CS, where it is specified as $\mathrm{REAL}_{CS}^{\Pi}(\mathrm{SCP}) = \{([\![\mathbf{x}]\!], [\![\mathbf{y}]\!]), [\![z]\!]\}$. $[\![\mathbf{x}]\!]$ and $[\![\mathbf{y}]\!]$ are the encrypted data vectors. $[\![z]\!]$ is the result of determining whether the encrypted data vectors $[\![\mathbf{x}]\!]$ and $[\![\mathbf{y}]\!]$ are equal. $[\![z]\!]$ is a random number between 0 and 1 in the ciphertext. We assume that $\mathrm{IDEAL}_{CS}^{\Pi}(\mathrm{SCP}) = \{([\![\mathbf{x}']\!], [\![\mathbf{y}']\!]), [\![z']\!]\}$ is the execution view of the simulation in the ideal world, where the encrypted data in both $[\![\mathbf{x}']\!]$ and $[\![\mathbf{y}']\!]$ are chosen randomly from $\mathbb{T}^{n+1}$. $[\![z']\!]$ are chosen randomly from $\mathbb{T}^{n+1}$. The semantic privacy of our encryption scheme makes $[\![\mathbf{x}]\!]$ and $[\![\mathbf{y}]\!]$ computationally indistinguishable from $[\![\mathbf{x}']\!]$ and $[\![\mathbf{y}']\!]$, respectively. In addition, $[\![z']\!]$ takes 0 or 1 with equal probability. $[\![z]\!]$ are computationally indistinguishable from $[\![z']\!]$, respectively. Thus, it can be concluded that $\mathrm{REAL}_{CS}^{\Pi}(\mathrm{SCP})$ and $\mathrm{IDEAL}_{CS}^{\Pi}(\mathrm{SCP})$ are computationally indistinguishable. We can obtain that **SCP** is secure under the semi-honest model. □

6.3. Security of CMPSI

Theorem 5. *The* **CMPSI** *proposed is secure under the semi-honest model, and the security of encrypted data, mining results, and query data can be guaranteed.*

Proof of Theorem 5. We can use the above method to prove that our proposed **CMPSI** is secure under the semi-honest model. In S1, CS obtains the encrypted dataset from DOs. In S2, CS obtains the encrypted dataset from DR. From Section 6.1, our cryptosystem is semantically secure, and the semi-honest CS cannot distinguish these messages from the random values of $\mathbb{T}^{n+1}$. In S3, **SCP** is executed to obtain the intersection information of the encryption of individual items in the dataset. Since **SCP** is secure in our system, it can be confirmed that the protocol in S3 is secure. In S4, $\mathbf{SC_{OR}}$ is used to obtain the final encryption result. Since $\mathbf{SC_{OR}}$ is secure in our system, the protocol in S4 is secure. In S5 and S6, the execution of S4 is repeated, the DR receives the message and decrypts it using the joint key, and the protocol is secure from the security of **MKTFHE**. □

Theorem 6. *The* **CMPSI** *proposed is able to resist man-in-the-middle attacks.*

Proof of Theorem 6. As shown in Figure 4, the participants represent the DR and DOs in our scenario. Under normal conditions, the participants can communicate with the CS, and Figure 4a shows the communication under normal conditions. The man-in-the-middle attack changes the original communication channel and can access the communication data between the participant and the cloud server, and Figure 4b shows the impact of the man-in-the-middle attack on the communication. We will prove that our model is resistant to man-in-the-middle attacks in three ways. First, DO encrypts its own dataset T_{DO} into $[\![T_{DO}]\!]_{s_{DO}}$ using its own key s_{DO} and then sends $[\![T_{DO}]\!]_{s_{DO}}$ to CS. Intermediary $\mathcal{A}$ obtains $[\![T_{DO}]\!]_{s_{DO}}$ through the new channel, but $\mathcal{A}$ does not have DO's key, and it is known from the security of **MKTFHE** that $\mathcal{A}$ cannot decrypt $[\![T_{DO}]\!]_{s_{DO}}$. Thus, our model can resist the man-in-the-middle attack during the data transmission from DO to CS. Second, DR wants to obtain the intersection information and sends the encrypted data $[\![T_{DR}]\!]_{s_{DR}}$ to CS. Intermediary $\mathcal{A}$ obtains $[\![T_{DR}]\!]_{s_{DR}}$ through the illegal channel. By the security of **MKTFHE**, $\mathcal{A}$ does not have s_{DR} and cannot obtain T_{DR} from $[\![T_{DR}]\!]_{s_{DR}}$. Thus, our model can resist the man-in-the-middle attack from DR to CS man-in-the-middle attack during the data transfer. Finally, CS needs to return the computed intersection information $[\![T_{DR\cap DO}]\!]_s$ to DR. The middleman $\mathcal{A}$ obtains the information $[\![T_{DR\cap DO}]\!]_s$, and it is known from the security of **MKTFHE** that $\mathcal{A}$ does not have the key to obtain $T_{DR\cap DO}$. Thus, our model can resist the man-in-the-middle attack during the data transmission from CS to DR. □

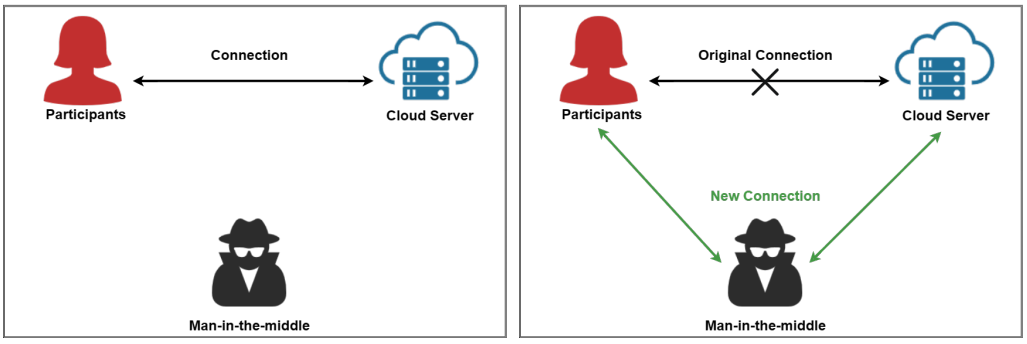

Figure 4. Man-in-the-middle attack; (**a**) normal communication; (**b**) post-attack communication.

6.4. Security Services

According to the above proof of CMPSI security, Table 3 shows the security services provided by the scheme and a demonstration from our model of how the method provides each of these functions.

Table 3. Security services provided.

Security Services	Definition	Proof
Confidentiality	Network information is not disclosed to non-authorized users, entities, or processes.	In our system, DO uses its own key s_{DO} to encrypt its own dataset T_{DO} into $[\![T_{DO}]\!]_{s_{DO}}$ and then sends $[\![T_{DO}]\!]_{s_{DO}}$ to CS. An unauthorized user $\mathcal{A}$ illegally obtains $[\![T_{DO}]\!]_{s_{DO}}$, and according to the security of the MKTFHE cryptosystem in Section 6.1, it is known that without the key, s_{DO} cannot perform decryption. Therefore, unauthorized illegal users $\mathcal{A}$ cannot obtain the information of DO's dataset T_{DO}.
Integrity	Information is transmitted, exchanged, stored, and processed in such a way that it remains uncorrupted or unmodified, that it is not lost, and that it cannot be changed without authorization.	In our system, DR encrypts the dataset T_{DR} as $[\![T_{DR}]\!]_{s_{DR}}$ using the key s_{DR} and sends $[\![T_{DR}]\!]_{s_{DR}}$ to CS. Attacker $\mathcal{A}$ obtains the dataset $[\![T_{DR}]\!]_{s_{DR}}$ through the intermediate channel, and according to the definition of the semi-honest model in Section 4.3, $\mathcal{A}$ does not modify or corrupt the data, and CS can obtain the dataset $[\![T_{DR}]\!]_{s_{DR}}$ intact.
Availability	Assurance that information is available to authorized users, i.e., assurance that legitimate users can use the required information when needed.	In our system, DR is the legal user. When DR wants to obtain the intersection information of its dataset T_{DR}, DR sends $[\![T_{DR}]\!]_{s_{DR}}$ to CS, and CS sends the computed intersection result $[\![T_{DR\cap DO}]\!]_s$ to DR. The legitimate user DR can obtain the required data when needed, which proves the usability of our system.
Non-repudiation	The two parties of information exchange cannot deny that they send or receive information in the exchange process.	In our system, DO sends its encrypted dataset $[\![T_{DO}]\!]_{s_{DO}}$ to CS. According to the definition of the semi-honest model in Section 4.3, DO will not deny that the dataset $[\![T_{DO}]\!]_{s_{DO}}$ is its data, proving the non-repudiation of our system.

7. Performance Analysis

In this section, we evaluate the time overhead and communication overhead of our proposed scheme. The experimental parameters we used [51] are shown in Table 4 below. According to one study [52], the parameters we use reach a privacy level of at least 110 bits, which is a common reference in this field.

Table 4. Parameter sets.

LWE-n	LWE-α	LWE-B$'$	LWE- d$'$	RLWE-N	RLWE-β	RLWE-B	RLWE-d
560	3.05×10^{-5}	2^2	8	1024	3.72×10^{-9}	2^9	3

The test environment used for our experiments was as follows: a 2.30 GHz Intel (R) Core(TM) i5-8300H Dell laptop. The programming language we used was C++, and our system was based on the MKTFHE library. First, we tested the efficiency of the security subprotocols separately. Then, we tested the communication overhead of our scheme and compared it with existing schemes. Finally, we tested our scheme.

7.1. Experiments on Security Computing Protocols

Our secure subprotocol experiments were performed using the MKTFHE library (https://github.com/ilachill/MK-TFHE) (1 February 2023). MKTFHE is a proof-of-concept implementation of a multi-key version of TFHE. The code is written on top of the TFHE library (https://tfhe.github.io/tfhe/) (1 February 2023). The computation of secure NAND gates is given in the MKTFHE library. In the MKTFHE-based implementation, our goal is to implement the MKLwe sample addition and multiplication operations as a way to implement the other circuit gates needed in our scheme in addition to the NADN gate. We first performed experiments on single circuit gates, including experiments on secure AND gate computation protocol, secure OR gate computation protocol, and secure XNOR computation protocol, and the experimental results are shown in Table 5. We compared these with NAND gates and found that the efficiency of individual gate computation is close.

Table 5. Experimental results for single circuit gates.

Gate Circuit	Key Generation Time (s)	FFT Conversion Time (s)	Bootstrapping Time (s)
AND	1.973	0.039	0.226
NAND	1.982	0.038	0.227
OR	1.956	0.040	0.227
XNOR	1.975	0.039	0.220

Then, as shown in Table 6, we tested the experimental time overhead of **SCP** for $k = 8$, 16, and 32, where k is the bits of data. The results show that the time overhead of the **SCP** protocol is linearly related to the number of bits of input.

Table 6. Running time of **SCP**.

k	**8**	**16**	**32**
Running time (s)	3.52	7.17	14.20

7.2. Overhead Evaluation

In our scenario, DOs and DRs are resource-constrained users; thus, it is important to have a smaller communication overhead. In our scheme, each participant uses their key to encrypt the data and uploads it to the cloud server; thus, the total communication overhead is related to the total data size. We tested the communication overhead of our scheme on datasets with aggregate sizes of 2^8, 2^{12}, 2^{16}, and 2^{20}. We compared our scheme with the scheme based on RSA [53] and the scheme based on pseudorandom permutation (PRP) [48]. As shown in Figure 5, our scheme is significantly superior to the privacy intersection scheme based on RSA. For the server-assisted scheme with limited security [48], the communication cost of our scheme is also lower. Our experimental results are the average of ten experiments.

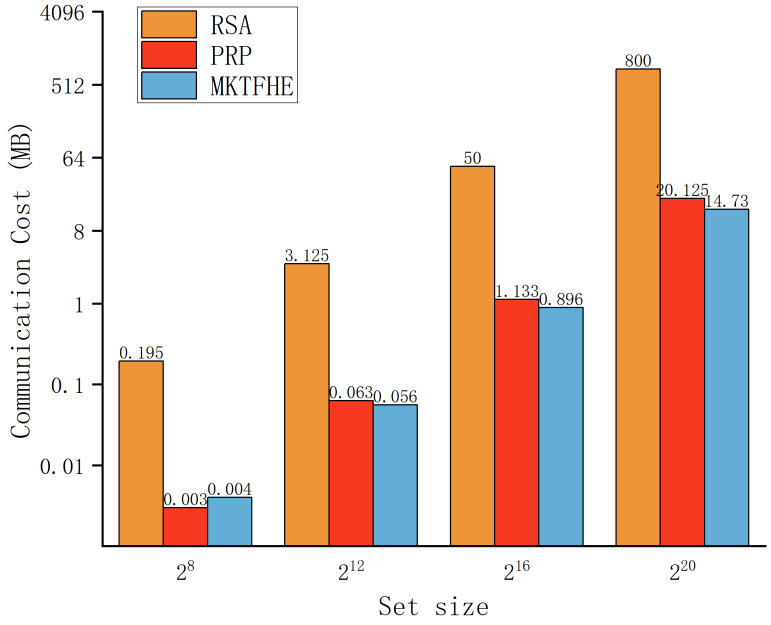

Figure 5. Communication overhead.

Our scheme is based on the underlying PSI protocol, and the computation of the ciphertext is performed directly on the cloud server. To the best of our knowledge, our proposed scheme is the first scheme that uses MKTFHE to achieve the ideal PSI, and the time overhead of the scheme is a very important metric. For users with limited resources,

low overhead in the process of data encryption and decryption is necessary. We tested the time cost of using encryption and decryption and the size of ciphertext on datasets with sizes of 2^8, 2^{12}, 2^{16} and 2^{20}. Table 7 shows that for DOs and DR with limited resources, the cost of our scheme in data encryption and decryption is very small. Finally, we tested the computing cost of the cloud server. In the experiment, we used data from 16, 32, and 64 bit systems to test the performance of our proposed scheme. Table 8 shows our experimental results. The results show that the time cost of the scheme is linearly related to the size of the dataset and the number of bits of data. Please note that the cloud has excellent computing power, so that the efficiency of the solution can be faster in actual use.

Table 7. Cost during encryption.

	2^8	2^{12}	2^{16}	2^{20}
Encryption time (ms)	13.5	208.2	3162.2	47,987.1
Cipher size (kb)	3.5	57.3	917.5	15,083.6

Table 8. Cloud computing time (min).

Data Set Size	16bit	32bit	64bit
2^2	0.51	0.99	1.98
2^4	8.47	16.02	31.70
2^6	137.68	273.83	547.66

8. Conclusions

In this paper, we proposed CMPSI, a cloud-assisted private set intersection via multi-key fully homomorphic encryption, which allows the participants to outsource the encrypted data to cloud servers for storage and computation. We also designed some MKTFHE-based secure computing protocols to complete the design of our system. We analytically demonstrated the security of our scheme under a semi-honest model. Through experiments, we tested the performance of our proposed scheme and proved that our scheme has less communication overhead by comparing it with existing schemes. We also proved the feasibility of the scheme.

As future research work, we plan to apply our proposed MKTFHE to a wider range of areas, such as association rule mining systems in large shopping malls. In addition, we will improve our framework to handle more complex computations and further improve the performance of our system.

Author Contributions: Conceptualization, C.F.; Methodology, X.L.; Software, C.F. and P.J.; Validation, P.J.; Formal analysis, M.L.; Investigation, M.L. and P.G.; Data curation, X.Z.; Writing—original draft, X.Z.; Writing—review & editing, L.W. and X.L.; Visualization, P.G.; Supervision, L.W. All authors have read and agreed to the published version of the manuscript.

Funding: This work was funded by the National Key Technology Research and Development Program of China (grant nos. 2021YFB3901000 and 2021YFB3901005); the Civil Aerospace Technology Advance Research Project of China (D040405); the Application Pilot Plan of Fengyun Satellite (FY-APP-2021.0501).

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

PSI	Private set intersection
CMPSI	Cloud-assisted multi-key private set intersection
TFHE	Fully homomorphic encryption over toru
MKTFHE	Multi-key fully homomorphic encryption over toru

References

1. Abdulsalam, Y.S.; Hedabou, M. Security and privacy in cloud computing: technical review. *Future Internet* **2022**, *14*, 11. [CrossRef]
2. Aburukba, R.; Kaddoura, Y.; Hiba, M. Cloud Computing Infrastructure Security: Challenges and Solutions. In Proceedings of the 2022 International Symposium on Networks, Computers and Communications (ISNCC), Shenzhen, China, 19–22 July 2022; pp. 1–7.
3. Shao, Z.; Bo, Y. Private set intersection via public key encryption with keywords search. *Secur. Commun. Netw.* **2015**, *8*, 396–402. [CrossRef]
4. Shi, R.H.; Mu, Y.; Zhong, H.; Cui, J.; Zhang, S. An efficient quantum scheme for Private Set Intersection. *Quantum Inf. Process.* **2016**, *15*, 363–371. [CrossRef]
5. Yang, X.; Luo, X.; Xu, A.W.; Zhang, S. Improved outsourced private set intersection protocol based on polynomial interpolation. *Concurr. Comput. Pract. Exp.* **2018**, *30*, e4329. [CrossRef]
6. Tajima, A.; Sato, H.; Yamana, H. Outsourced Private Set Intersection Cardinality with Fully Homomorphic Encryption. In Proceedings of the 2018 6th International Conference on Multimedia Computing and Systems (ICMCS), Rabat, Morocco, 10–12 May 2018.
7. Ruan, O.; Huang, X.; Mao, H. An efficient private set intersection protocol for the cloud computing environments. In Proceedings of the 2020 IEEE 6th International Conference on Big Data Security on Cloud (BigDataSecurity), Baltimore, MD, USA, 25–27 May 2020; pp. 254–259.
8. Jiang, Y.; Wei, J.; Pan, J. Publicly Verifiable Private Set Intersection from Homomorphic Encryption. In Proceedings of the Security and Privacy in Social Networks and Big Data: 8th International Symposium, SocialSec 2022, Xi'an, China, 16–18 October 2022; pp. 117–137.
9. Debnath, S.K.; Kundu, N.; Choudhury, T. Efficient post-quantum private set-intersection protocol. *Int. J. Inf. Comput. Secur.* **2022**, *17*, 405–423. [CrossRef]
10. Wang, Q.; Zhou, F.; Xu, J.; Peng, S. Tag-based verifiable delegated set intersection over outsourced private datasets. *IEEE Trans. Cloud Comput.* **2020**, *10*, 1201–1214. [CrossRef]
11. Pinkas, B.; Rosulek, M.; Trieu, N.; Yanai, A. SpOT-light: lightweight private set intersection from sparse OT extension. In Proceedings of the Advances in Cryptology—CRYPTO 2019: 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, 18–22 August 2019; pp. 401–431.
12. Pinkas, B.; Rosulek, M.; Trieu, N.; Yanai, A. PSI from PaXoS: fast, malicious private set intersection. In Proceedings of the Advances in Cryptology—EUROCRYPT 2020: 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, 10–14 May 2020; pp. 739–767.
13. Chase, M.; Miao, P. Private set intersection in the internet setting from lightweight oblivious PRF. In Proceedings of the Advances in Cryptology—CRYPTO 2020: 40th Annual International Cryptology Conference, CRYPTO 2020, Santa Barbara, CA, USA, 17–21 August 2020; pp. 34–63.
14. Rindal, P.; Schoppmann, P. VOLE-PSI: fast OPRF and circuit-psi from vector-ole. In Proceedings of the Advances in Cryptology—EUROCRYPT 2021: 40th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, 17–21 October 2021; pp. 901–930.
15. Shi, R.H.; Li, Y.F. Quantum private set intersection cardinality protocol with application to privacy-preserving condition query. *IEEE Trans. Circuits Syst. Regul. Pap.* **2022**, *69*, 2399–2411. [CrossRef]
16. Zhou, Q.; Zeng, Z.; Wang, K.; Chen, M. Privacy Protection Scheme for the Internet of Vehicles Based on Private Set Intersection. *Cryptography* **2022**, *6*, 64. [CrossRef]
17. Qian, Y.; Xia, X.; Shen, J. A profile matching scheme based on private set intersection for cyber-physical-social systems. In Proceedings of the 2021 IEEE Conference on Dependable and Secure Computing (DSC), Aizuwakamatsu, Japan, 30 January–2 February 2021; pp. 1–5.
18. Demmler, D.; Rindal, P.; Rosulek, M.; Trieu, N. *PIR-PSI: Scaling Private Contact Discovery*; Cryptology ePrint: Archive, CA, USA, 2018.
19. Liu, F.; Ng, W.K.; Zhang, W.; Han, S.; et al. Encrypted set intersection protocol for outsourced datasets. In Proceedings of the 2014 IEEE International Conference on Cloud Engineering, Boston, MA, USA, 11–14 March 2014; pp. 135–140.
20. De Cristofaro, E.; Tsudik, G. Practical private set intersection protocols with linear complexity. In Proceedings of the Financial Cryptography and Data Security: 14th International Conference, FC 2010, Tenerife, Canary Islands, 25–28 January 2010; pp. 143–159.
21. Gentry, C. Fully homomorphic encryption using ideal lattices. In Proceedings of the 41st Annual ACM Symposium on Theory of Computing, Bethesda, MA, USA, 31 May–2 June 2009; pp. 169–178.

22. Brakerski, Z.; Perlman, R. Lattice-based fully dynamic multi-key FHE with short ciphertexts. In Proceedings of the Advances in Cryptology—CRYPTO 2016: 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, 14–18 August 2016; pp. 190–213.
23. López-Alt, A.; Tromer, E.; Vaikuntanathan, V. On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption. In Proceedings of the 44th Annual ACM Symposium on Theory of Computing, New York, NY, USA, 20–22 May 2012; pp. 1219–1234.
24. Gentry, C.; Sahai, A.; Waters, B. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In Proceedings of the Annual Cryptology Conference, Barbara, CA, USA, 18–22 August 2013; pp. 75–92.
25. Brakerski, Z.; Gentry, C.; Vaikuntanathan, V. (Leveled) fully homomorphic encryption without bootstrapping. *ACM Trans. Comput. Theory (TOCT)* **2014**, *6*, 1–36. [CrossRef]
26. Chillotti, I.; Gama, N.; Georgieva, M.; Izabachene, M. Faster fully homomorphic encryption: Bootstrapping in less than 0.1 seconds. In Proceedings of the International Conference on the Theory And Application of Cryptology and Information Security, Taipei, Taiwan, 5–9 December 2016; pp. 3–33.
27. Chen, H.; Laine, K.; Rindal, P. Fast private set intersection from homomorphic encryption. In Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, Dallas, TX, USA, 30 October–3 November 2017; pp. 1243–1255.
28. Chen, H.; Huang, Z.; Laine, K.; Rindal, P. Labeled PSI from fully homomorphic encryption with malicious security. In Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, Toronto, ON, Canada, 15–19 October 2018; pp. 1223–1237.
29. Cong, K.; Moreno, R.C.; da Gama, M.B.; Dai, W.; Iliashenko, I.; Laine, K.; Rosenberg, M. Labeled PSI from homomorphic encryption with reduced computation and communication. In Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security, Copenhagen, Denmark, 15–19 November 2021; pp. 1135–1150.
30. Freedman, M.J.; Nissim, K.; Pinkas, B. Efficient private matching and set intersection. In Proceedings of the Advances in Cryptology-EUROCRYPT 2004: International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, 2–6 May 2004; pp. 1–19.
31. Yao, A.C. Protocols for secure computations. In Proceedings of the 23rd Annual Symposium on Foundations of Computer Science (SFCS 1982), Chicago, IL, USA, 3–5 November 1982; pp. 160–164.
32. Micali, S.; Goldreich, O.; Wigderson, A. How to play any mental game. In Proceedings of the 19th ACM Symposium on Theory of Computing, New York, NY, USA, 1 January 1987; pp. 218–229.
33. Kolesnikov, V. Gate evaluation secret sharing and secure one-round two-party computation. In Proceedings of the Advances in Cryptology-ASIACRYPT 2005: 11th International Conference on the Theory and Application of Cryptology and Information Security, Chennai, India, 4–8 December 2005; pp. 136–155.
34. Even, S.; Goldreich, O.; Lempel, A. A randomized protocol for signing contracts. *Commun. ACM* **1985**, *28*, 637–647. [CrossRef]
35. Hazay, C.; Nissim, K. Efficient Set Operations in the Presence of Malicious Adversaries. In Proceedings of the Public Key Cryptography, Paris, France, 26–28 May 2010; Volume 6056; pp. 312–331.
36. Huang, Y.; Evans, D.; Katz, J. Private set intersection: Are garbled circuits better than custom protocols? In Proceedings of the NDSS, San Diego, CA, USA, 5–8 February 2012.
37. Dong, C.; Chen, L.; Wen, Z. When private set intersection meets big data: An efficient and scalable protocol. In Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security, Berlin, Germany, 4–8 November 2013; pp. 789–800.
38. Pinkas, B.; Schneider, T.; Zohner, M. Faster Private Set Intersection based on OT Extension (Full Version). In Proceedings of the USENIX Security Symposium, San Diego, CA, USA, 20–22 August 2014.
39. Freedman, M.J.; Hazay, C.; Nissim, K.; Pinkas, B. Efficient set intersection with simulation-based security. *J. Cryptol.* **2016**, *29*, 115–155. [CrossRef]
40. Pinkas, B.; Schneider, T.; Zohner, M. Scalable private set intersection based on OT extension. *ACM Trans. Priv. Secur. (TOPS)* **2018**, *21*, 1–35. [CrossRef]
41. Orrù, M.; Orsini, E.; Scholl, P. Actively secure 1-out-of-N OT extension with application to private set intersection. In Proceedings of the Topics in Cryptology–CT-RSA 2017: The Cryptographers' Track at the RSA Conference 2017, San Francisco, CA, USA, 14–17 February 2017; pp. 381–396.
42. Kerschbaum, F. Collusion-resistant outsourcing of private set intersection. In Proceedings of the 27th Annual ACM Symposium on Applied Computing, Trento, Italy, 25–29 March 2012; pp. 1451–1456.
43. Kerschbaum, F. Outsourced private set intersection using homomorphic encryption. In Proceedings of the Proceedings of the 7th ACM Symposium on Information, Computer and Communications Security, Hong Kong, 7–11 June 2012; pp. 85–86.
44. Abadi, A.; Terzis, S.; Dong, C. O-PSI: delegated private set intersection on outsourced datasets. In Proceedings of the ICT Systems Security and Privacy Protection: 30th IFIP TC 11 International Conference, SEC 2015, Hamburg, Germany, 26–28 May 2015; Proceedings 30; pp. 3–17.
45. Abadi, A.; Terzis, S.; Dong, C. VD-PSI: verifiable delegated private set intersection on outsourced private datasets. In Proceedings of the Financial Cryptography and Data Security: 20th International Conference, FC 2016, Christ Church, Barbados, 22–26 February 2016; Revised Selected Papers 20; pp. 149–168.
46. Ali, M.; Mohajeri, J.; Sadeghi, M.R.; Liu, X. Attribute-based fine-grained access control for outscored private set intersection computation. *Inf. Sci.* **2020**, *536*, 222–243. [CrossRef]

47. Abadi, A.; Terzis, S.; Metere, R.; Dong, C. Efficient Delegated Private Set Intersection on Outsourced Private Datasets. *IEEE Trans. Dependable Secur. Comput.* **2019**, *16*, 608–624. [CrossRef]
48. Kamara, S.; Mohassel, P.; Raykova, M.; Sadeghian, S. Scaling private set intersection to billion-element sets. In Proceedings of the Financial Cryptography and Data Security: 18th International Conference, FC 2014, Christ Church, Barbados, 3–7 March 2014; Revised Selected Papers 18; pp. 195–215.
49. Chen, H.; Chillotti, I.; Song, Y. Multi-key homomorphic encryption from TFHE. In Proceedings of the International Conference on the Theory and Application of Cryptology and Information Security, Kobe, Japan, 8–12 December 2019; pp. 446–472.
50. Oded, G. *Foundations of Cryptography: Volume 2, Basic Applications*; Cambridge University Press: Cambridge, MA, USA, 2009.
51. Pradel, G.; Mitchell, C. Privacy-Preserving Biometric Matching Using Homomorphic Encryption. In Proceedings of the 2021 IEEE 20th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom), Shenyang, China, 18–20 August 2021; pp. 494–505.
52. Albrecht, M.R.; Player, R.; Scott, S. On the concrete hardness of learning with errors. *J. Math. Cryptol.* **2015**, *9*, 169–203. [CrossRef]
53. Ciampi, M.; Orlandi, C. Combining private set-intersection with secure two-party computation. In Proceedings of the Security and Cryptography for Networks: 11th International Conference, SCN 2018, Amalfi, Italy, 5–7 September 2018; pp. 464–482.