Computers, Materials & Continua

Tech Science Press

DOI: 10.32604/cmc.2023.037783
Article

An Effective Security Comparison Protocol in Cloud Computing

Yuling Chen[1,2], Junhong Tao[1], Tao Li[1,*], Jiangyuan Cai[3] and Xiaojun Ren[4]

[1]State Key Laboratory of Public Big Data, College of Computer Science and Technology, Guizhou University, Guiyang, 550000, China
[2]Guangxi Key Laboratory of Cryptography and Information Security, Guilin University of Electronic Technology, Guilin, 550000, China
[3]POWERCHINA Guizhou Electric Power Engineering Co., Ltd., Guiyang, 550000, China
[4]Blockchain Laboratory of Agricultural Vegetables, Weifang University of Science and Technology, Weifang, 261000, China

*Corresponding Author: Tao Li. Email: litao_2019@qfnu.edu.cn
Received: 06 November 2022; Accepted: 22 February 2023

Abstract: With the development of cloud computing technology, more and more data owners upload their local data to the public cloud server for storage and calculation. While this can save customers' operating costs, it also poses privacy and security challenges. Such challenges can be solved using secure multi-party computation (SMPC), but this still exposes more security issues. In cloud computing using SMPC, clients need to process their data and submit the processed data to the cloud server, which then performs the calculation and returns the results to each client. Each client and server must be honest. If there is cooperation or dishonest behavior between clients, some clients may profit from it or even disclose the private data of other clients. This paper proposes the SMPC based on a Partially-Homomorphic Encryption (PHE) scheme in which an addition homomorphic encryption algorithm with a lower computational cost is used to ensure data comparability and Zero-Knowledge Proof (ZKP) is used to limit the client's malicious behavior. In addition, the introduction of Oblivious Transfer (OT) technology also ensures that the semi-honest cloud server knows nothing about private data, so that the cloud server of this scheme can calculate the correct data in the case of malicious participant models and safely return the calculation results to each client. Finally, the security analysis shows that the scheme not only ensures the privacy of participants, but also ensures the fairness of the comparison protocol data.

Keywords: Secure comparison protocols; zero-knowledge proof; homomorphic encryption; cloud computing

1 Introduction

Nowadays, due to the continuous progress of network technology, more and more sensitive data needs to be processed over the network [1]. Such a large amount of sensitive data requires

collaborative computing in various security fields to solve the privacy and security problems, including edge computing, secure multi-party computing, deep learning, cloud computing, federated learning, the Internet of Things, etc., [2,3]. In collaborative computing, participants need to provide their own sensitive data. This not only exposes the sensitive data of the participants but also makes them vulnerable to malicious adversaries during transmission [4,5]. For example, Cloud computing is a technology that can process and store data on remote cloud servers. It can be combined with secure multi-party computing technology to form a Secure Multi-Party in Cloud Computing (SMCC) scenario, where initially participants need to actually submit their private data to the cloud server due to unknown network environments, malicious adversaries, etc. Therefore, private data is vulnerable to malicious attacks or interception in the transmission process, and the cloud server cannot fully guarantee the data's privacy security after obtaining the data. This paper mainly improves the security comparison protocol proposed by Damard et al. (DGK) [6]. The main idea is to use the additive homomorphism of the homomorphic encryption algorithm to calculate each bit and compare the size of the input from both sides without compromising privacy. That is, each client uses the PHE algorithm to encrypt its secret input and then sends the resulting ciphertext to the cloud server. At the same time, ZKP [7] is used to ensure the protocol's correctness. Finally, the cloud server performs operations on the ciphertext and sends the comparison results to each client. Compared to the Full Homomorphic Encryption (FHE) scheme, the PHE in this scheme requires lower computational overhead, and the protocol applies to malicious participant models due to the introduction of the ZKP. More importantly, due to the introduction of OT [8] technology, the cloud server knows nothing about data information. The main contributions of this research work are:

(1) This paper proposes a security comparison protocol using ZKP technology, which can not only securely compare sensitive data but also be applied in the malicious party model.

(2) The scheme uses PHE's addition homomorphism to split the XOR operation of the original protocol so that all computing steps are handed over to the cloud server, which not only ensures the illegal operation of the participants but also ensures the security of the protocol.

(3) The cloud server in this scheme obtains the key bits of clients' data by using OT technology, which not only ensures the sensitive data of clients is not leaked but also ensures the fairness of the protocol.

The organizational structure of this paper is as follows: The second section introduces the related work, and the third section mainly introduces the basic technology used in this paper. The fourth section mainly introduces the details of this scheme. The fourth section is a detailed description of the ZKP algorithm, and the fifth section is an analysis of security and fairness. The sixth section is the conclusion of this paper.

2 Related Work

With the development of 5G technology [9] and the popularity of intelligent mobile devices, although it provides convenience for people's lives, the massive amount of data also brings severe challenges to various fields [10,11]. Terminal devices cannot process such massive and sensitive data, such as Artificial Intelligence (AI), Internet of Things (IoT), Wireless Sensors (WS), Machine Learning (ML), and so on [12–15]. The emergence of cloud computing effectively solves this problem. Cloud computing is a technology that can process and store data on a remote cloud server and at the same time, allows any device connected to the internet to access data [16]. Cloud computing is not only widely used in computer terminals but can also be applied to mobile terminals [17].

In a cloud computing environment, users' data and computing are migrated to an external, virtualized "cloud server," [18], and the cloud server performs data operations and storage. While this method can reduce the maintenance of computer hardware and software [19], it poses many privacy security challenges [20–22]. For example, the unloading of big data causes security problems such as network load and sensitive data loss [23]. There are many studies using blockchain to protect data privacy [24]. However, no matter the public or private chain, there is not only the risk of data leakage but also the easy possibility of malicious attacks [25,26]. Caprolu et al. [27] mainly introduced the advantages and differences of edge computing compared with cloud computing and analyzed data security issues. Qi et al. [28] proposed a category recommendation model for Internet of Things privacy awareness points of interest (POI) based on group preferences. In practical application, this model can not only protect users' privacy information but also make full use of the information contained in the data set to mine users' interests. In terms of data security and privacy protection, SMPC based on homomorphic encryption has become an increasingly mainstream data privacy protection method [29]. Derfouf et al. [30] proposed a security protocol based on partial homomorphic encryption, compared the protocol with the complete homomorphic encryption protocol, and analyzed the advantages and disadvantages of the two schemes. An effective response to the challenge of cloud computing is the foundation for building the Industrial Internet of Things (IIoT) [31].

SMPC can effectively address security and privacy issues in cloud computing [32]. Yao [33] first proposed the millionaire problem, which officially launched the SMPC study, in which several participants with secret inputs work together to compute a function and get their output. Still, none of the participants get any information about the inputs [34]. SMPC can be achieved by garbled circuits, secret sharing, and homomorphic encryption [35].

2.1 Secure Multi-party Computation Based on Full-Homomorphic Encryption

Gentry [36] first proposed a FHE scheme that could perform arbitrary operations in ciphertext. However, key storage and generation of fully homomorphic schemes are too heavy on the network, making them impractical. The dynamic unloading algorithm proposed by Chen et al. [37] can effectively improve the user utility function and realize the optimal unloading of dynamic data. Han et al. [38] introduced the multi-key full-homomorphic encryption scheme based on the Learning with Errors (LWE) encryption scheme and the attribute-based multi-key full-homomorphic encryption scheme, which can perform SMPC in an uncertain cloud environment. Mittal et al. [39] discussed the advantages and disadvantages of different FHE schemes in detail and finally gave suggestions on the prospect of FHE application in cloud computing.

Although FHE is suitable for cloud computing environments, the high computational overhead cannot be applied well in reality. And the FHE scheme needs to ensure that participants are semi-honest. The reason is that it is unrealistic to introduce the ZKP to limit malicious behavior among participants when the cost is already high. Therefore, this paper conducts research in the direction of the PHE, and the ZKP can make the scheme in this paper applicable to malicious participant models.

2.2 Secure Multi-party Computation Based on Partially Homomorphic Encryption

Mahmood et al. [40] constructed a hybrid homomorphic encryption scheme using the additive homomorphic Goldwasser Micali (GM) algorithm and the multiplicative homomorphic Rivest-Shamir-Adleman (RSA) algorithm, which not only improved security but also optimized the algorithm's computational overhead, but still needed to trust the interaction between clients. Tan et al. [41] use Wireless Body Area Network (WBAN) to authenticate nodes, which improves data reliability.

This authentication method can reduce storage costs and resource loss during data transmission. Fan et al. [42] propose a secure approach for data sharing between different domains based on edge computing models. The problem of authentication between different domains is solved, effectively ensuring node trust. Ghanem et al. [43] extended the Homomorphic Encryption Library (HElib) to support SMPC is detailed. The performance of the scheme is also analyzed. Becher et al. [44] proposed a SMPC based on a novel approach to random index distribution security. Luo et al. [45] proposed a multi-round SMPC, was built using information entropy, which ensures security between participants and is more suitable for cloud computing.

3 Preliminaries

3.1 AH-ElGamal Encryption Algorithm

Chen et al. [46] is a variant design of the Elgamal [47] encryption algorithm. It has the property of addition homomorphism. That is, given the ciphertext $E(m_1, r_1)$ and $E(m_2, r_2)$ under the same public key, the private key holder can obtain the ciphertext sum of m_1 and m_2 through homomorphism calculation, that is, $E(m_1, r_1)\, E(m_2, r_2) = E(m_1 + m_2, r_1 + r_2)$. The algorithm consists of key generation, encryption and decryption.

Key Generation: Select two large prime numbers a and b randomly, let $n = ab$, select two generators g and h in Z_n^*, select number x in Z_n randomly, set the public key to (n, g, h, y), calculate $y = h^x mod n$ and set the private key to x.

Encryption: Encrypts the message m, selects the number r randomly, and obtains the ciphertext $C = E(m) = (c_1, c_2) = (g^m y^r, h^r)$.

Decryption: $g^m = c_1(c_2^*)^{-1}$ is first computed and then searched according to the pre-computed index table concerning g to obtain plaintext m.

3.2 Zero-Knowledge Proof

ZKP was first proposed by Golfwasser et al. [48] in 1985. It means that the prover proves to the verifier that an event is actual without giving away any relevant information. ZKP system has three characteristics: completeness, reliability, and zero-knowledge. Completeness means that the verifier always accepts a proposition when it is true. Reliability refers to the probability that the proposition will be accepted by any fraudster when the proposition is not standing. Zero-knowledge means that the prover learns nothing new about the proposition from the evidence.

ZKP is divided into interactive and non-interactive. Interactive ZKP means that the prover needs to constantly challenge the prover to verify the "knowledge" of the prover without disclosing the information of the "knowledge." Non-interactive ZKP does not require repeated challenges but instead uses a random oracle to challenge. The random oracle returns a random output for any input, but for the same input, the random oracle uses one output method each time. In other words, a random oracle is a function that maps all inputs and outputs randomly.

ZKP in this paper replaces all challenges with hash values through a Fiat-Shamir [49] heuristic algorithm, which can be used to transform interactive ZKP into non-interactive ZKP.

3.3 Oblivious Transfer

OT is a cryptographic protocol first proposed by Rabin [50] in 1981. In this protocol, the sender of the message sends a message from the database to the receiver via the Oblivious Transfer technology. However, throughout the transmission process, the sender does not know which message is sent to the

receiver, which ensures anonymity in the information query process. Naor et al. [8] proposed a reusable 1-out-of-N OT protocol based on the Diffie-Hellman hypothesis. The following uses 1-out-of-N OT protocol as an example, The Alice inputs strings $(X_0, X_1, X_2, \ldots, X_{N-1})$, and the Bob gets the output X_r by inputting a bit r. From the perspective of Algorithm 1, the algorithm has two subjects, namely Alice and Bob. Alice has data that Bob wants and the algorithm requires Bob not only to get the data he wants, but also not to disclose any information to Alice. The most essential function of this algorithm for the scheme proposed in this paper is to provide data privacy protection. For details, see the third phase in the next chapter.

Algorithm 1: 1-out-of-N OT protocol

Preliminaries: protocol operates over a group Z_p of prime order. More specifically, G_p can be a subgroup of order q of Z_p^*, where p is prime and $q|p-1$. Let g be a generator group. The protocol uses a function H which is assumed to be a random oracle.

Input: The Bob's input is $r \in (0, N-1)$, and the Alice's input is N strings $(X_0, X_1, ..., X_{N-1})$.

Output: The Bob's output is X_r.

1: Alice: Select random number $a \in Z_q$ and random constants $(C_1, C_2, ..., C_{N-1}) \in Z_q$. To calculate the $g^a, C_i^a (i \in (1, N-1))$. The values $C_1, ..., C_{N-1}$ and g^a send to the Bob.

2: Bob: Select random number $k \in (1, q)$, and set $PK_r = g^k$, if $r \neq 0$ bob computes $PK_0 = \frac{C_r}{g^k}$, send PK_0 to the Alice.

3: Alice: Calculate the PK_0^a and $PK_i^a = \frac{C_i^a}{PK_0^a}, i \in (1, N-1)$. The Alice chooses a random string R(The string should be long enough so that there are no two invocations of the protocol in which R obtains the same value.) and then encrypts each X_i.
$E_i = (H((PK_i^a), R, i) \bigoplus X_i)$
Send E_i and R to the Bob.

4: Bob: The Bob uses $H((g^a)^k, R, r)$ to decrypt E_i.

5: **End**

3.4 DGK Comparison Protocol

DGK comparison protocol is an efficient and secure comparison protocol under the semi-honest model proposed by Damgard et al. [6]. This protocol uses PHE additive homomorphism to solve the millionaire problem efficiently. DGK protocol starts XOR calculation from a high level until the XOR value is 1. The comparison result of this crucial bit can determine the size of two binary numbers. Suppose two binary numbers are represented by m_a and m_b, respectively, and $1 \leq i \leq l$, where l is the length of the binary number, and i is the ordinal number of the binary number. Under the plaintext, the following formula is calculated to obtain the comparison result. If $c_i = 0$, then prove that $m_a < m_b$.

$$c_i = 1 + m_{a,i} - m_{b,i} + 3 \sum_{j=i+1}^{l} (m_{a,j} \oplus m_{a,j}) \tag{1}$$

If the comparison is made in ciphertext, the final result can be obtained by calculating the following formula. For example, $[m_{a,i}] = g^{m_{a,i}} h^{r_{a,i}} \ mod \ n$ indicates the ciphertext that uses the public key of the encryption algorithm to encrypt $m_{a,i}$.

$$\begin{aligned}[c_i] &= \left[1 + m_{a,i} - m_{b,i} + 3 \sum_{j=i+1}^{l} \left(m_{a,j} \oplus m_{a,j}\right)\right] \\ &= [1] \cdot [m_{a,i}] \cdot [m_{b,i}]^{-1} \cdot \left(\prod_{j=i+1}^{l} [m_{a,j} \oplus m_{a,j}]\right)^3 \end{aligned} \tag{2}$$

After the server gets the value of $[c_i]$, to prevent the client from extrapolating the encryption result through repeated learning, the original protocol adds a blind factor to hide the result (However, it

does not affect the final decryption result, because the result of any power of 1 is 1, in other words, any power of g^0 is still g^0), so the server selects the random blind factor s' and blinds $[c_i]$ by the following formula.

$$[c_i]' = [c_i]^{s'} \ mod \ n \tag{3}$$

Finally, it is decrypted by the client. If 0 exists in the obtained result, it is shown that $m_1 < m_2$.

4 Improved DGK Comparison Protocol Based on Zero-Knowledge Proof (ZKP-DGK)

The scheme in this article assumes that under the SMCC scenario, for example, as shown in Fig. 1, the clients submit their own encrypted data to the cloud server for calculation and then return the ciphertext data to each client. The cloud server restricts the illegal behavior of clients by using ZKP technology and transferring comparative computing tasks in the original DGK protocol to the cloud service provider, so as long as the cloud server is semi-honest, it will not leak any privacy.

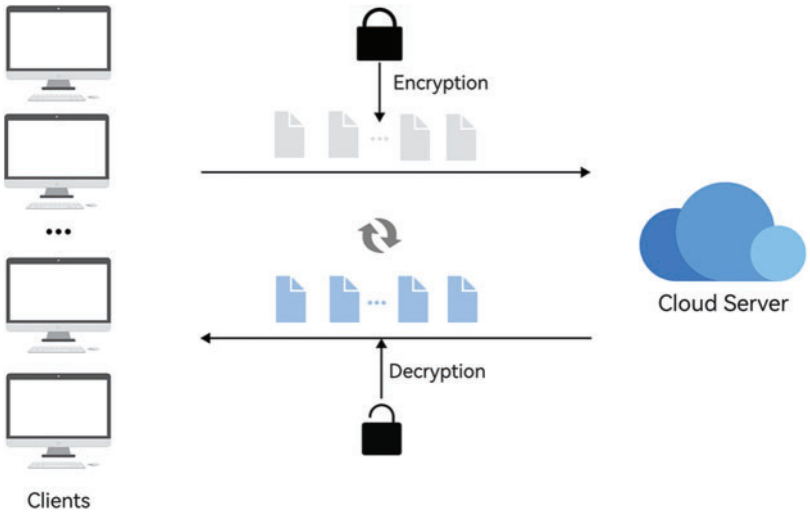

Figure 1: Secure multi-party in cloud computing (SMCC)

4.1 Initialization Phase

The public-private key pair generation mechanism of the participants uses the key generation algorithm in the AH-ElGamal encryption algorithm. After the initialization setting to generate data, suppose there are clients A, B, and server C. The server randomly selects two large prime numbers, a and b, let $n = ab$, and chooses two generating elements, g and h, from Z_n^*. Each client randomly selects the number x in Z_n as its private key and satisfies y as the public key, and the public-private key pairs of each subject are denoted as (y_1, x_1), (y_2, x_2), and (y_3, x_3). The encryption form is $C = E(m) = (c_1, c_2) = (g^m y^r, h^r)$, the decryption form is $g^m = c_1 (c_2{}^x)^{-1}$, and m can be decrypted according to the exponential table. The protocol scheme is broadly implemented in four phases, and each client needs to submit the corresponding ZKP in each phase.

4.2 Exponential Multiplication Phase

The public-private key pair generation mechanism of the participants uses the key generation algorithm in the AH-ElGamal encryption algorithm. After the initialization setting to generate data,

suppose there are clients A, B, and server C. The server randomly selects two large prime numbers, a and b, let $n = ab$, and chooses two generating elements, g and h, from Z_n^*. Each client randomly selects the number x in Z_n as its private key and satisfies y as the public key, and the public private key pairs of each subject are denoted as (y_1, x_1), (y_2, x_2) and (y_3, x_3), the encryption form is $C = E(m) = (c_1, c_2) = (g^m y^r, h^r)$, the decryption form is $g^m = c_1 (c_2{}^x)^{-1}$, and m can be decrypted according to the exponential table. The protocol scheme is broadly carried out in four phases, and each client needs to submit the corresponding zero-knowledge proofs in each phase.

Clients A and B use server C's public key to encrypt their plaintext information $m_{a,i}$ and $m_{b,i}$ respectively, and the encryption format is AH-ElGamal encryption algorithm mentioned above. Similarly, the encrypted ciphertext of client A is $C_{a,i} = E(m_{a,i}) = (c_{1,i}, c_{2,i}) = (g^{m_{a,i}} y^{r_{a,i}}, h^{r_{a,i}})$, and the encrypted ciphertext of client B is $C_{b,i} = E(m_{b,i}) = (c'_{1,i}, c'_{2,i}) = (g^{m_{b,i}} y^{r_{b,i}}, h^{r_{b,i}})$. $r_{a,i}$ and $r_{b,i}$ are random numbers selected by clients A and B during the encryption, which are visible only to clients. $m_{a,i}$ and $m_{b,i}$ are the comparative plaintext of A and B. $m_{a,i}$ consists of the binary number l bits, where i represents the number of bits in binary. For example, $m_{a,i}$ represents the ith bit of m_a , where $0 < i \leq l$. The process at this phase is shown in Fig. 2.

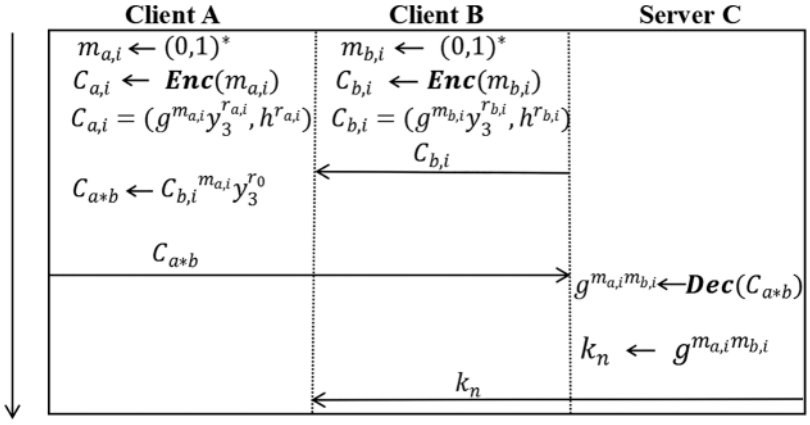

Figure 2: Exponential multiplication phase

First, the client A and B both encrypt their own each bit after getting $\{C_{a,l}, C_{a,l-1} \ldots C_{a,1}\}$ and $\{C_{b,l}, C_{b,l-1} \ldots C_{b,1}\}$, respectively. Then client B sends all the ciphertext to client A. After client A receives all the ciphertext from client B, client A uses its own plaintext value $m_{a,i}$ to perform an exponential operation on the ciphertext of client B and adds a random number r_0 to hide $m_{a,i}$. To ensure the decryption of the encryption algorithm, the second part of the ciphertext is processed in the same way, and finally the ciphertext $C_{a*b} = (c^*_{1,i}, c^*_{2,i}) = (g^{m_{a,i}m_{b,i}} y^{m_{a,i}r_{b,i}+r_0}, h^{m_{a,i}r_{b,i}+r_0})$ is obtained. The resulting ciphertext is sent to server C, and the discrete logarithm encryption proof is submitted. Then C_{a*b} is sent to server C. At last, server C can obtain the value of $m_{a,i}m_{b,i}$ through decryption. Find the value $m_{a,i}m_{b,i} = 0$ in the result and mark it as k_n, where $0 < n \leq l$.

4.3 Additive Homomorphism Phase

Client A multiplies B's ciphertext k_n bit with his ciphertext's to get $C_{a,k_n} C_{b,k_n} = (c_{1,k_n} * c'_{1,k_n}, c_{2,k_n} * c'_{2,k_n}) = (g^{m_{a,k_n}+m_{b,k_n}} y^{r_{a,k_n}+r_{b,k_n}}, h^{r_{a,k_n}+r_{b,k_n}})$, and client A sends all the results to server C. Server C then decrypts each ciphertext. Obtain $m_{a,k_n}m_{b,k_n}$ from $g^{m_{a,k_n}m_{b,k_n}} = c_{1,k_n} * c'_{1,k_n} (c_{2,k_n} * c'_{2,k_n})^{-x_3} = g^{m_{a,k_n}+m_{b,k_n}} y_3{}^{r_{a,k_n}+r_{b,k_n}} (h^{r_{a,k_n}+r_{b,k_n}})^{-x_3}$. Server C then calculates the XOR value $c_\oplus$ through formula (4). Finally, server C finds the

first sequence number with a value of 1 in the result for $c_{\oplus}$ and records it as the key comparison bit t. The process at this phase is shown in Fig. 3.

$$c_{\oplus} = m_{a,k_n} + m_{b,k_n} - 2m_{a,k_n}m_{b,k_n} \tag{4}$$

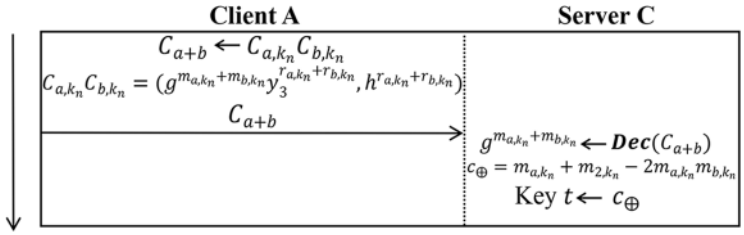

Figure 3: Additive homomorphism phase

4.4 The OT Phase

Before this phase, clients A and B must combine their public keys to generate the common public key $y' = y_1 y_2$. The common public key here adopts the idea of threshold decryption for the final result to be decrypted and verified in the OT phase jointly by A and B, to avoid the evil of one party. Frist, clients A and B use the joint public key y' to generate ciphertext $C'_{a,i} = E'(m_{a,i}) = (c_{a,i}, c'_{a,i}) = (g^{m_{a,i}} y'^{r_{a,i}}, h^{r_{a,i}})$ and $C'_{b,i} = E'(m_{b,i}) = (c_{b,i}, c'_{b,i}) = (g^{m_{b,i}} y'^{r_{b,i}}, h^{r_{b,i}})$ respectively, and send the ciphertext to the OT. Then server C enters t into the OT. After OT performs calculation, server C can obtain $C'_{a,t}$ and $C'_{b,t}$ ciphertext. Finally, server C calculates C^* to get the final comparison result: Finally, server C can get C^* by formulas (5) and (6) and send C^* to clients A and B. The process at this phase is shown in Fig. 4.

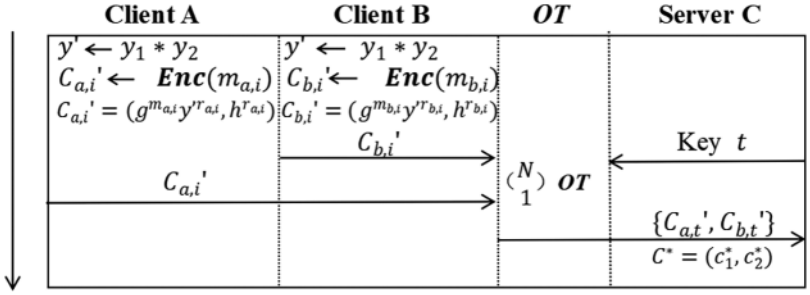

Figure 4: The OT phase

$$c_1^* = g * c_{a,t} * c_{b,t}^{-1} = g^{1+m_{a,t}-m_{b,t}} y'^{r_{a,t}+r_{b,t}} \tag{5}$$

$$c_2^* = c'_{a,t} * c'^{-1}_{b,t} = h^{r_{a,t}-r_{b,t}} \tag{6}$$

4.5 Verification Phase

After obtaining the ciphertext comparison result C^*, clients A and B jointly decrypt the result to obtain $g^{1+m_{a,t}-m_{b,t}}$. If $g^{1+m_{a,t}-m_{b,t}}$ is 1 then $m_a < m_b$ is proved. The process at this phase is shown in Fig. 5.

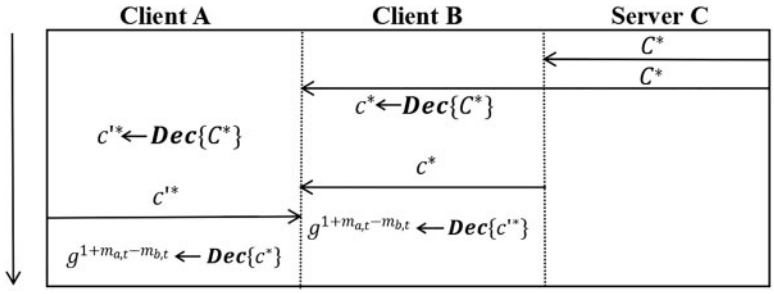

Figure 5: Verification phase

The scheme process described above can not only securely compare confidential data but also ensure the security and fairness of each participant. Compared with the original DGK protocol, this paper proposes that this approach can be used to modify any malicious operation under ZKP restrictions and apply it to malicious participant models.

5 Zero-Knowledge Proof

5.1 Binary Format Proof

Because interactive ZKP is highly inconvenient in the use process (data interaction between sender and receiver is required), the ZKP in this paper adopts Fiat-Shamir [49] heuristic algorithm to replace all interactive steps with the random oracle. And this method can be used to transform interactive ZKP into non-interactive ZKP. This algorithm can be simply understood as using a hash function to replace the random challenge value of the data interaction in the interactive ZKP.

Algorithm 2 is described as follows: There is cyphertext $C_{a,i} = E\left(m_{a,i}\right) = (c_1, c_2) = (g^{m_{a,i}} y^{r_{a,i}}, h^{r_{a,i}})$. The prover proves to the verifier that $m_{a,i}$ in c_1 is the correct binary format [7].

Algorithm 2: Binary Format Proof

Input: $Proof_{1,0}^{C_{a,i}}(c_1, m_{a,i}, r_{r,i})$, random numbers $(\lambda_1,\lambda_2,\lambda_3) \leftarrow Z_q$, Challenge C , total number l.

Output: $Verify_{1,0}^{C_{a,i}}(Proof_{1,0}^{C_{a,i}}, c_1)$

Prover:
for $i = 1, i \leqslant l$ **do**
  $t_1 = g^{\lambda_1} y^{\lambda_2}$
  $t_2 = g^{\lambda_1 m_{a,i}} y^{\lambda_3}$
$H : \{0,1\}^* \leftarrow Z_p^*$ is a secure one-way hash function.The Prover computes Challenge C through a random oracle: $C = H(g, y, t_1, t_2)$
  $s_1 = m_{a,i} C + \lambda_1$
  $s_2 = r_{a,i} C + \lambda_2$
  $s_3 = r_{a,i}(C - s_1) + \lambda_3$
Endfor
Send $(t_1, t_2, C, s_1, s_2, s_3)$ to the Veifier.
Verifier:
if $c_1^C t_1 = g^{s_1} y^{s_2}, c_1^{C-s_1} t_2 = g^0 y^{s_3}$ **then**
  $Verify_{1,0}^{C_{a,i}} = 1$
else
  $Verify_{1,0}^{C_{a,i}} = 0$

5.2 Proof of Ciphertext Format Correctness

Algorithms 3 and 4 are described as follows: Participants must prove that in $C_{a,i} = E\left(m_{a,i}\right) = (c_1, c_2) = (g^{m_{a,i}} y^{r_{a,i}}, h^{r_{a,i}})$ are encrypted following the AH-ElGamal encryption form. The proof is divided into two parts: representation proof and discrete logarithm proof.

(1) Part c_1: The ciphertext is $C_{a,i} = E\left(m_{a,i}\right) = (c_1, c_2) = (g^{m_{a,i}} y^{r_{a,i}}, h^{r_{a,i}}), 1 < i \leq l$. Prover knows that witness ($m_{a,i}$, $r_{a,i}$) and proves that $c_1 = g^{m_{a,i}} y^{r_{a,i}}$ holds.

Algorithm 3: Proof of Part c_1

Input: $Proof^{c_1}(c_1, m_{a,i}, r_{a,i})$, random numbers $(\beta_1,\beta_2) \leftarrow Z_q$, Challenge C , total number l.
Output: $Verify^{c_1}(Proof^{c_1}, c_1)$

Prover:
for $i = 1, i \leqslant l$ **do**
  $t = g^{\beta_1} y^{\beta_2}$
  $H : \{0,1\}^* \leftarrow Z_p^*$ is a secure one-way hash function.The Prover computes Challenge C through a random oracle: $C = H(g, y, c_1, t)$.
  $s_1 = m_{a,i} C + \beta_1$
  $s_2 = r_{a,i} C + \beta_2$
Endfor
Send (t, C, s_1, s_2) to the Veifier.
Verifier:
if $t \times c_1^C = g^{s_1} \times y^{s_2}$ **then**
  $Verify^{c_1} = 1$
else
  $Verify^{c_1} = 0$

(2) Part c_2: The ciphertext is $C_{a,i} = E\left(m_{a,i}\right) = (c_1, c_2) = (g^{m_{a,i}} y^{r_{a,i}}, h^{r_{a,i}}), 1 < i \leq l$. Prover knows that witness ($r_{a,i}$) and proves that $c_2 = h^{r_{a,i}}$ holds.

Algorithm 4: Proof of Part c_2

Input: $Proof^{c_2}(c_2, r_{a,i})$, random numbers $\beta_3 \leftarrow Z_q$, Challenge C , total number l.
Output: $Verify^{c_2}(Proof^{c_2}, c_2)$

Prover:
for $i = 1, i \leqslant l$ **do**
  $t = h^{\beta_3}$
  $H : \{0,1\}^* \leftarrow Z_p^*$ is a secure one-way hash function.The Prover computes Challenge C through a random oracle: $C = H(h, c_2, t)$.
  $s = \beta_3 - C r_{a,i}$
Endfor
Send (t, C, s) to the Veifier.
Verifier:
if $c_2^C \times h^s = t$ **then**
  $Verify^{c_2} = 1$
else
  $Verify^{c_2} = 0$

5.3 Random Number Consistency Proof

The encryption format used in this scheme is $(c_1, c_2) = (g^{m_{a,i}} y^{r_{a,i}}, h^{r_{a,i}})$. Since there are two ciphertexts c_1 and c_2, c_2 is used for decryption. If c_1 and c_2 are inconsistent during the encryption, the decryption operation cannot be performed usually. Therefore, it is necessary to prove that $r_{a,i}$ of c_1 and c_2 are consistent.

Algorithm 5 is described as follows: There is ciphertext $C_{a,i} = (C_{a,l}, C_{a,l-1}, C_{a,l-2} \ldots C_{a,1})$, where $C_{a,i} = E\left(m_{a,i}\right) = (c_1, c_2) = (g^{m_{a,i}} y^{r_{a,i}}, h^{r_{a,i}}), 1 < i \leq l$. The Prover knows witness ($m_{a,i}$, $r_{a,i}$) and shows that $r_{a,i} = r_{a,i}$ in $(g^{m_{a,i}} y^{r_{a,i}}, h^{r_{a,i}})$.

Algorithm 5: Random Number Consistency Proof

Input: $Proof_r^{C_{a,i}}(c_1, m_{a,i}, r_{r,i})$, random numbers $(\gamma_1,\gamma_2,\gamma_3) \leftarrow Z_q$, Challenge C , total number l.
Output: $Verify_r^{C_{a,i}}(Proof_r^{C_{a,i}})$.

Prover:
for $i = 1, i \leqslant l$ **do**
  $t = g^{\gamma_1} y^{\gamma_2} h^{\gamma_3}$
  $H : \{0,1\}^* \leftarrow Z_p^*$ is a secure one-way hash function.The Prover computes Challenge C through a random oracle: $C = H(g, y, t_1, t_2)$
  $s_1 = \gamma_1 - m_{a,i} C$
  $s_2 = \gamma_2 - r_{a,i} C$
  $s_3 = \gamma_3 - r_{b,i} C$
Endfor
Send (t, C, s_1, s_2, s_3) to the Veifier.
Verifier:
  $z = c_1 c_2 = g^{m_{a,i}} y^{r_{a,i}} h^{r_{b,i}}$
if $g^{s_1} y^{s_2} h^{s_3} z^C = t$ **then**
  $Verify_r^{C_{a,i}} = 1$
else
  $Verify_r^{C_{a,i}} = 0$

5.4 Ciphertext Consistency Proof

Since this scheme uses multiple rounds of comparison, it is necessary to ensure that the plaintext information contained in the ciphertext submitted by participants in each round is consistent.

Algorithm 6 is described as follows: The encryption format is $C_{a,i} = (C_{a,l}, C_{a,l-1}, C_{a,l-2} \ldots C_{a,1})$, where$C_{a,i} = E\left(m_{a,i}\right) = (c_1, c_2) = (g^{m_{a,i}} y^{r_{a,i}}, h^{r_{a,i}}), 1 < i \leq l$. The ciphertext $c_1 = g^{m_{a,i}} y^{r_{a,i}}$ in the first round and $c_1' = g^{m'_{a,i}} y^{r_{a,i}}$ in the second round are set. The participants need to prove that $m_{a,i}$ is consistent with $m'_{a,i}$.

6 Security and Fairness Analysis

6.1 Security Analysis

Since the objective of this scheme is to apply under the malicious participant model, a security analysis of the zero-knowledge proofs submitted in the scheme is required.

Algorithm 6: Ciphertext Consistency Proof

Input: $Proof^{m_{a,i}}(c_v)$, random number δ.
Output: $Verify^{m_{a,i}}(Proof^{m_{a,i}})$

Prover:
$t = y^{\delta}$
$H : \{0,1\}^* \leftarrow Z_p^*$ is a secure one-way hash function.The Prover computes Challenge C through a random oracle: $C = H(g, y, t)$
$s = \delta + C \times (r_{a,i} - r'_{a,i})$
Send (t, C, s) to Vierfier
End

Vierfier:
if $y^s = t \times y^{c_v^C}$ **then**
$Verify_r^{m_{a,i}} = 1$
else
$Verify_r^{m_{a,i}} = 0$

6.1.1 Binary Format Proof

Suppose there exists a malicious adversary A' that is able to construct a new challenge C' by mimicking the challenge C of the random oracle and constructs a response s'_1, s'_2, s'_3, while submitting a false proof $Proof_{1,0}^{C_{a,i}}(C_{a,i}, m_{a,i}, r_{a,i})$. Adversary A' can get $c_1^{C-C'} = g^{s_1-s'_1} y^{s_2-s'_2}$ and $c_1^{C-s_1+C'-s'_1} = g^0 y^{s_3-s'_3}$ by combining equations, and then the $m_{a,i}$ and $r_{a,i}$ values of the ciphertext can be obtained by equations $(s_1 - s'_1)(C_1 - C'_1)^{-1}$ and $(s_2 - s'_2)(C_1 - C'_1)^{-1}$, so the plaintext is completely compromised. However, in the case where the security parameter is large enough, the malicious adversary A' must guess the same C' from the random prediction machine under the satisfied condition, which is considered infeasible, and after guessing C' and to obtain the value of t_1 at the same time, the value of t_1 needs to be found by discrete logarithm values of λ_1 and λ_2. The difficulty is to solve the discrete logarithm problem on G_p in polynomial time, which is also considered infeasible, so the hypothesis is not valid.

6.1.2 Proof of Ciphertext Format Correctness

Since the ciphertext encryption format of the participants takes the AH-ElGamal encryption algorithm, the protocol participants need to ensure that the ciphertext is encrypted in the form of $C_{a,i} = E(m_{a,i}) = (c_1, c_2) = (g^{m_{a,i}} y^{r_{a,i}}, h^{r_{a,i}})$. Therefore, this scheme requires the participants to submit two proofs of and respectively. The following is an example of Part c_1.

(1) Part c_1: Suppose that the malicious client A' can construct a new challenge C' by mimicking the challenge C of the random oracle. Then the value of the ciphertext $m_{a,i}$ can be obtained by combining the equation $(s_1 - s'_1)(C_1 - C'_1)^{-1}$ and the value of the random number $r_{a,i}$ can be obtained by $(s_2 - s'_2)(C_1 - C'_1)^{-1}$. Thus, ciphertext c_1 is breached, resulting in plaintext leakage. In the same way as binary format proof analysis, the cheating client A' must guess the same C' from the random oracle under the satisfied condition, which is considered infeasible with sufficiently significant security parameters.

(2) Part c_2: It is assumed that cheating participant A' can forge an identical response to s', and then the r value of the ciphertext can be obtained by combining the equation $\frac{s - s'}{C - C'}$. Like the above analysis, cheating participant A' would have to guess the same C' from the random oracle if the

conditions were satisfied, which would be considered infeasible if the security parameters were large enough.

6.1.3 Exponential Multiplication Phase

In the exponential multiplication phase of the ZKP-DGK scheme, client A needs to exponentiate the ciphertext of client B with its plaintext, and then add a random number r_0 to construct the ciphertext form of $C_{b,i}^{m_{a,i}} y_3^{r_0} = g^{m_{a,i}m_{b,i}} y_3^{m_{a,i}r_{b,i}+r_0}$. This proof can be analyzed by the discrete logarithm proof, and by the same token, it is infeasible for the client A to attempt to forge false proof. And in the additive homomorphism phase, since the ciphertext is bound using the Pedersen commitment, it is also sufficiently secure for A and B.

6.1.4 Verification Phase

In the third stage, clients A and B need to encrypt with the standard public key y' and submit proof of correctness, similar to the first stage. Since server C in the ZKP-DGK scheme is a semi-honest model, and neither client A nor client B under the malicious model is involved in the comparison calculation, as long as server C follows the protocol correctly, then even if clients A and B are malicious adversaries, the security of the scheme will not be affected, and the security of the remaining phases can be demonstrated by referring to the security of the DGK protocol.

6.2 Fairness Analysis

Since this scheme assumes that the clients are malicious models and a semi-honest server C is introduced, it is necessary to consider whether there is information leakage in each stage of data interaction and ciphertext computation in this scheme.

6.2.1 Exponential Multiplication Phase

For malicious clients A and B, want to maximize their gains, so this subsection first analyzes whether clients A and B can access each other's personal input. In the first phase, client A indexes the secret message received from B with its plaintext to obtain $C_{b,i}^{m_{a,i}}$. To avoid the leakage of A's plaintext and add a random number r_0 to hide $m_{a,i}$ one step deeper to obtain $g^{m_{a,i}m_{b,i}} y_3^{m_{a,i}r_{b,i}+r_0}$ and submit proof of correctness of ciphertext format. In this process, client A will not know anything about $m_{b,i}$, because A only performs the exponential operation on $C_{b,i}$. For client B, the value of $m_{a,i}$ cannot be solved from the index table due to the random number r_0. For server C, server C can only decrypt the value of $m_{a,i}$ and $m_{b,i}$ multiplied by $g^{m_{a,i}m_{b,i}} y_3^{m_{a,i}r_{b,i}+r_0}$, but does not obtain the specific values of $m_{a,i}$ and $m_{b,i}$.

6.2.2 Additive Homomorphism Phase

At this stage, client A needs to multiply its ciphertext $C_{a,i}$ with client B's ciphertext $C_{b,i}$ to obtain $C_{a,i}\ C_{b,i} = \left(g^{m_{a,i}+m_{b,i}} y^{r^{a,i}+r^{b,i}}, h^{r_{a,i}+r_{b,i}}\right)$. This stage is the same as the previous proof stage, where clients A and B cannot obtain explicit information about $m_{a,i}$ and $m_{b,i}$. For server C, which has information about $m_{a,i}m_{b,i}$ and $m_{a,i} + m_{b,i}$ at this point, it is fair for all parties involved since server C cannot to determine the specific, exact value of $m_{a,i}$ or $m_{b,i}$ based on $m_{a,i}m_{b,i}$ and $m_{a,i} + m_{b,i}$.

6.2.3 The OT Phase

This phase of oblivious transfer (OT) refers to the 1-out-of-N model in a black box environment. The server C inputs the sequence number t of the critical comparison bit, and the corresponding

ciphertexts $C'_{a,t}$ and $C'_{b,t}$ are obtained by the OT protocol. In this technique, clients A and B cannot know which ciphertext data server C has obtained, and at the same time server C can obtain ciphertexts $C'_{a,t}$ and $C'_{b,t}$.

6.2.4 Verification Phase

At this stage, clients A and B need to decrypt jointly to get the final comparison result, thus ensuring that A and B cannot decrypt privately. Since clients A and B did not participate in the calculation of the comparison results, and server C needed to verify the submitted zero-knowledge proof, clients could be anything wrong. As for semi-honest server C, as long as the server correctly executes the agreement, the agreement's fairness will not be affected.

Analysis shows that, throughout the scheme, neither client A nor client B nor server C have access to private input data. Therefore, this scheme is fair to the clients. A comparison between this paper and [6] and [51] is shown in Table 1. The DGK comparison protocol can only be compared between participants, which relies heavily on honesty between participants. Yu et al. [51] proposed a scheme constructed with ZKP. Although the scheme can be applied to malicious participant models to some extent, the security and fairness of the scheme still depend on the Trusted Third Party (TTP). In this paper, ZKP-DGK not only protects the user's private data, but also limits malicious behavior between data parties, which provides the basis for some real-world applications, such as electronic voting, privacy auction, big data fusion, and so on [52]. With the implementation of practical applications, the data will become larger and larger, placing a severe burden on the network [53]. A large number of scholars have researched this aspect so that network overhead will be the next step for this solution.

Table 1: Security and fairness comparison

Scheme	TTP	Malicious participant models	Security	Fairness
[6]	×	×	Rely on semi-honest participants	Rely on semi-honest participants
[51]	√	√	Rely on the honesty of TTP	Rely on the fairness of TTP
ZKP-DGK	√	√	Yes	Relatively fair

7 Conclusion

HE-based security comparison protocol provides positive implications for the development of cloud computing. However, distrust between data owners and the expensive computing overhead of a complete homomorphism still pose serious challenges to cloud computing. This paper combines the ZKP and PHE, improves DGK comparison protocol, uses the cloud server in cloud computing to replace the third party, and proposes a security comparison protocol suitable for malicious parties. In this scheme, zero-knowledge proofs are used to curb malicious client behavior. Compared to FHE, PHE not only guarantees the privacy of ciphertext but also has a more minor computational cost. During the entire process, private information will not be leaked to any party, including trusted servers. The final analysis shows that the scheme can achieve the security and fairness of ciphertext comparison. However, all scenarios proposed in this paper are completed under the condition that the cloud server is running normally. That is, although the cloud server cannot obtain any valuable data, its

behavior must be guaranteed to be correct. Otherwise, the protocol will not be downright correctly. In future work, we will continue to study ZKP, HE, edge computing, etc., and finally construct a security comparison protocol under a completely malicious model.

Acknowledgement: We are thankful to State Key Laboratory of Public Big Data of Guizhou University for providing an environment for editing manuscripts and experiments.

Funding Statement: This work is financially supported by the National Natural Science Foundation of China under Grant No. (62202118.61962009). And in part by Natural Science Foundation of Shandong Province (ZR2021MF086). And in part by Top Technology Talent Project from Guizhou Education Department (Qian jiao ji [2022]073). And in part by Foundation of Guangxi Key Laboratory of Cryptography and Information Security (GCIS202118).

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] J. Y. Dong, W. J. Wu, Y. Gao, X. X. Wan and P. B. Si, "Deep reinforcement learning based worker selection for distributed machine learning enhanced edge intelligence in internet of vehicles," *Intelligent and Converged Networks*, vol. 1, no. 3, pp. 234–242, 2020.

[2] K. El Makkaoui, A. Beni-Hssane and A. Ezzati, "Cloud-elgamal: An efficient homomorphic encryption scheme," in *2016 Int. Conf. on Wireless Networks and Mobile Communications (WINCOM)*, Fez, Morocc, pp. 63–66, 2016. https://doi.org/10.1109/WINCOM.2016.7777192

[3] L. Y. Qi, Y. H. Yang, X. K. Zhou, W. Rafique and J. Ma, "Fast anomaly identification based on multiaspect data streams for intelligent intrusion detection toward secure industry 4.0," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 9, pp. 6503–6511, 2022.

[4] T. Li, Y. L. Chen, Y. L. Wang, Y. L. Wang, M. H. Zha *et al.,* "Rational protocols and attacks in blockchain system," *Security and Communication Networks*, vol. 1-11, no. 2020, pp. 1–11, 2020. https://doi.org/10.1155/2020/8839047

[5] L. Z. Kong, G. S. Li, W. Rafique, S. G. Shen, Q. He *et al.,* "Time-aware missing healthcare data prediction based on ARIMA model," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, pp. 1–10, 2022. https://doi.org/10.1109/TCBB.2022.3205064

[6] I. Damgard, M. Geisler and M. Kroigaard, "Homomorphic encryption and secure comparison," *International Journal of Applied Cryptography*, vol. 1, no. 1, pp. 22–31, 2008.

[7] J. Groth and M. Kohlweiss, "One-out-of-many proofs: Or how to leak a secret and spend a coin," In: E. Oswald, M. Fischlin (Eds.), *Advances in Cryptology-EUROCRYPT 2015*, Berlin Heidelberg: Springer, pp. 253–280, 2015. https://doi.org/10.1007/978-3-662-46803-6_9

[8] M. Naor and B. Pinkas, "Efficient oblivious transfer protocols," in *Proc. of the Twelfth Annual ACM-SIAM Symp. on Discrete Algorithms, ser. SODA '01.*, Washington, D.C, USA, pp. 448–457, 2001. https://doi.org/10.1145/365411.365502

[9] Y. Zhang, H. Q. Zhang, J. Cosmas, N. Jawad, K. Ali *et al.,* "Internet of radio and light: 5G building network radio and edge architecture," *Intelligent and Converged Networks*, vol. 1, no. 1, pp. 37–57, 2020.

[10] F. Wang, G. S. Li, Y. L. Wang, W. Rafique, M. R. Khosravi *et al.,* "Privacy-aware traffic flow prediction based on multi-party sensor data with zero trust in smart city," *ACM Transactions on Internet Technology*, pp. 1232, 2022. https://doi.org/10.1145/3511904

[11] Y. L. Chen, X. Yang, T. Li, Y. Ren and Y. Y. Long, "A blockchain-empowered authentication scheme for worm detection in wireless sensor network," *Digital Communications and Networks*, vol. 8, pp. 111, 2022. https://doi.org/10.1016/j.dcan.2022.04.007

[12] H. Benfenatki, C. F. Da Silva, A. N. Benharkat, P. Ghodous and F. Biennier, "Methodology for semi-automatic development of cloud-based business applications," in *2014 IEEE 7th Int. Conf. on Cloud Computing*, Anchorage, AK, USA, pp. 954–955, 2014. https://doi.org/10.1109/CLOUD.2014.139
[13] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari and M. Ayyash, "Internet of things: A survey on enabling technologies, protocols, and applications," *IEEE Communications Surveys & Tutorials*, vol. 17, no. 4, pp. 2347–2376, 2015.
[14] M. Ayad, M. Taher and A. Salem, "Mobile gpu cloud computing with real time application," in *5th Int. Conf. on Energy Aware Computing Systems & Applications*, Cairo, Egypt, pp. 1–4, 2015. https://doi.org/10.1109/ICEAC.2015.7352209
[15] L. Y. Qi, W. M. Lin, X. Y. Zhang, W. C. Dou, X. L. Xu *et al.,* "A correlation graph based approach for personalized and compatible web apis recommendation in mobile app development," *IEEE Transactions on Knowledge and Data Engineering*, pp. 1, 2022. https://doi.org/10.1109/TKDE.2022.3168611
[16] W. Zhang, X. Chen and J. H. Jiang, "A multi-objective optimization method of initial virtual machine fault-tolerant placement for star topological data centers of cloud systems," *Tsinghua Science and Technology*, vol. 26, no. 1, pp. 95–111, 2021.
[17] Y. L. Wang, Tao Li, M. Liu, C. M. Li and H. Wang, "STSIIML: Study on token shuffling under incomplete information based on machine learning," *International Journal of Intelligent Systems*, vol. 37, no. 12, pp. 11078–11100, 2022.
[18] Z. Salman and W. M. Elmedany, "A trustworthy cloud environment using homomorphic encryption: A review," in *4th Smart Cities Symp. (SCS 2021), Online Conf.*, Bahrain, pp. 31–36, 2021. https://doi.org/10.1049/icp.2022.0308
[19] S. M. Ghanem and I. A. Moursy, "Secure multiparty computation via homomorphic encryption library," in *2019 Ninth Int. Conf. on Intelligent Computing and Information Systems (ICICIS)*, Cairo, Egypt, pp. 227–232, 2019. https;//doi.org/10.1109/ICICIS46948.2019.9014698
[20] A. K. Sandhu, "Big data with cloud computing: Discussions and challenges," *Big Data Mining and Analytics*, vol. 5, no. 1, pp. 32–40, 2022.
[21] F. Wang, H. Zhu, G. Srivastava, S. Li, M. R. Khosravi *et al.,* "Robust collaborative filtering recommendation with user-item-trust records," *IEEE Transactions on Computational Social Systems*, 2021. https://doi.org/10.1109/TCSS.2021.3064213
[22] L. Z. Kong, L. N. Wang, W. W. Gong, C. Yan, Y. C. Duan *et al.,* "Lsh-aware multitype health data prediction with privacy preservation in edge environment," *World Wide Web Journal*, vol. 25, no. 5, pp. 1793–1808, 2021. https://doi.org/10.1007/s11280-021-00941-z
[23] Y. L. Chen, J. Sun, Y. Yang, T. Li, X. Niu *et al.,* "Psspr: A source location privacy protection scheme based on sector phantom routing in wsns," *International Journal of Intelligent Systems*, vol. 37, no. 2, pp. 1204–1221, 2022.
[24] K. Lu and C. Y. Zhang, "Blockchain-based multiparty computation system," in *2020 IEEE 11th Int. Conf. on Software Engineering and Service Science (ICSESS)*, Beijing, China, pp. 28–31, 2020. https://doi.org/10.1109/ICSESS49938.2020.9237698
[25] T. Li, Z. J. Wang, G. Y. Yang, Y. Cui, Y. L. Chen *et al.,* "Semi-selfish mining based on hidden markov decision process," *International Journal of Intelligent Systems*, vol. 36, no. 7, pp. 3596–3612, 2021.
[26] T. Li, Z. J. Wang, Y. L. Chen, C. M. Li, Y. L. Jia *et al.,* "Is semi-selfish mining available without being detected?," *International Journal of Intelligent Systems*, vol. 37, no. 12, pp. 10576–10597, 2022.
[27] M. Caprolu, R. Di Pietro, F. Lombardi and S. Raponi, "Edge computing perspectives: Architectures, technologies, and open security issues," in *2019 IEEE Int. Conf. on Edge Computing (EDGE)*, Milan, Italy, pp. 116–123, 2019. https://doi.org/10.1109/EDGE.2019.00035
[28] L. Y. Qi, Y. W. Liu, Y. L. Zhang, X. L. Xu, M. Bilal *et al.,* "Privacy-aware point-of-interest category recommendation in internet of things," *IEEE Internet of Things Journal*, vol. 9, no. 21, pp. 21398–21408, 2022. https://doi.org/10.1109/JIOT.2022.3181136

[29] A. Chouhan, A. Kumari and M. Saiyad, "Secure multiparty computation and privacy preserving scheme using homomorphic elliptic curve cryptography," in *2019 Int. Conf. on Intelligent Computing and Control Systems (ICCS)*, Madurai, India, pp. 776–780, 2019. https://doi.org/10.1109/ICCS45141.2019.9065645
[30] M. Derfouf and M. Eleuldj, "Cloud secured protocol based on partial homomorphic encryptions," in *2018 4th Int. Conf. on Cloud Computing Technologies and Applications (Cloudtech)*, Brussels, Belgium, pp. 1–6, 2018. https://doi.org/10.1109/CloudTech.2018.8713353
[31] Y. H. Yang, X. Yang, M. Heidari, G. Srivastava, M. R. Khosravi *et al.,* "Astream: Data-stream-driven scalable anomaly detection with accuracy guarantee in IIoT environment," *IEEE Transactions on Network Science and Engineering*, pp. 1, 2022. https://doi.org/10.1109/TNSE.2022.3157730
[32] T. Oladunni and S. Sharma, "Homomorphic encryption and data security in the cloud," in *Proc. of 28th Int. Conf. on Software Engineering and Data Engineering*, San Diego, California, USA, vol. 64, pp. 129–138, 2019. https://doi.org/10.29007/drnc
[33] A. C. Yao, "Protocols for secure computations," in *23rd Annual Symp. on Foundations of Computer Science (sfcs 1982)*, Chicago, IL, USA, pp. 160–164, 1982. https://doi.org/10.1109/SFCS.1982.38
[34] Y. L. Chen, S. Dong, T. Li, Y. L. Wang and H. Y. Zhou, "Dynamic multi-key fhe in asymmetric key setting from lwe," *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 5239–5249, 2021. https://doi.org/10.1109/TIFS.2021.3127023
[35] K. Patel, "Secure multiparty computation using secret sharing," in *2016 Int. Conf. on Signal Processing, Communication, Power and Embedded System (SCOPES)*, Paralakhemundi, India, pp. 863–866, 2016. https://doi.org/10.1109/SCOPES.2016.7955564
[36] C. Gentry, "Fully homomorphic encryption using ideal lattices," in *Proc. of the Forty-First Annual ACM Symp. on Theory of Computing (STOC '09)*, New York, NY, USA, pp. 169–178, 2009. https://doi.org/10.1145/1536414.1536440
[37] Y. Chen, F. Zhao, Y. Lu and X. Chen, "Dynamic task offloading for mobile edge computing with hybrid energy supply," *Tsinghua Science and Technology*, 2021. https://doi.org/10.26599/TST.2021.9010050
[38] J. L. Han, Z. L. Wang, Y. Q. Shi, M. J. Wang and H. Dong, "Secure multiparty computation via fully homomorphic encryption scheme," in *2018 Eighth Int. Conf. on Instrumentation & Measurement, Computer, Communication and Control (IMCCC)*, Cairo, Egypt, pp. 250–253, 2018. https://doi.org/10.1109/IMCCC.2018.00060
[39] S. Mittal, K. Ramkumar and A. Kaur, "Preserving privacy in clouds using fully homomorphic encryption," in *2021 Int. Conf. on Smart Generation Computing, Communication and Networking (SMART GENCON)*, Pune, India, pp. 1–7, 2021. https://doi.org/10.1109/SMARTGENCON51891.2021.9645822
[40] Z. H. Mahmood and M. K. Ibrahem, "New fully homomorphic encryption scheme based on multistage partial homomorphic encryption applied in cloud computing," in *2018 1st Annual Int. Conf. on Information and Sciences (AiCIS)*, Fallujah, Iraq, pp. 182–186, 2018. https://doi.org/10.1109/AiCIS.2018.00043
[41] X. Tan, J. L. Zhang, Y. J. Zhang, Z. Qin, Y. Ding *et al.,* "A puf-based and cloud-assisted lightweight authentication for multi-hop body area network," *Tsinghua Science and Technology*, vol. 26, no. 1, pp. 36–47, 2021.
[42] K. Fan, Q. Pan, J. X. Wang, T. T. Liu, H. Li *et al.,* "Cross-domain based data sharing scheme in cooperative edge computing," in *2018 IEEE Int. Conf. on Edge Computing (EDGE)*, San Francisco, CA, USA, pp. 87–92, 2018. https://doi.org/10.1109/EDGE.2018.00019
[43] S. M. Ghanem and I. A. Moursy, "Secure multiparty computation via homomorphic encryption library," in *2019 Ninth Int. Conf. on Intelligent Computing and Information Systems (ICICIS)*, Cairo, Egypt, pp. 227–232, 2019. https://doi.org/10.1109/ICICIS46948.2019.9014698
[44] K. Becher and T. Strufe, "Efficient cloud-based secret shuffling via homomorphic encryption," in *2020 IEEE Symp. on Computers and Communications (ISCC)*, Rennes, France, pp. 1–7, 2020. https://doi.org/10.1109/ISCC50000.2020.9219588
[45] Y. Luo, Y. L. Chen, T. Li, Y. L. Wang, Y. X. Yang *et al.,* "An entropy-view secure multiparty computation protocol based on semi-honest model," *Journal of Organizational and End User Computing (JOEUC)*, vol. 34, no. 10, pp. 1–17, 2022.

[46] Z. W. Chen, R. Q. Zhang, Y. T. Yang and Z. C. Li, "A homomorphic elgamal variant based on bgn's method," in *2013 Int. Conf. on Cyber-Enabled Distributed Computing and Knowledge Discovery*, Beijing, China, pp. 1–5, 2013. https;//doi.org/10.1109/CyberC.2013.10

[47] T. Elgamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," *IEEE Transactions on Information Theory*, vol. 31, no. 4, pp. 469–472, 1985.

[48] S. Goldwasser, S. Micali and C. Rackoff, "The knowledge complexity of interactive proof-systems," in *Proc. of the Seventeenth Annual ACM Symp. on Theory of Computing (STOC '85)*, New York, NY, USA, pp. 291–304, 1985. https://doi.org/10.1145/22145.22178

[49] M. Bellare and P. Rogaway, "Random oracles are practical: A paradigm for designing efficient protocols," in *Proc. of the 1st ACM Conf. on Computer and Communications Security, ser. CCS '93*, New York, NY, USA, Association for Computing Machinery, pp. 62–73, 1993. https://doi.org/10.1145/168588.168596

[50] M. O. Rabin, "How to exchange secrets by oblivious transfer," *Technical report*, Aiken Computation Laboratory. Harvard University, 1981.

[51] R. C. Yu, "Research on the sealed-bid auction scheme for blockchain based on secure comparison protocols," M.S. Dissertation, College of Information Engineering, Northwest A&F University, China, 2019.

[52] J. Z. Wang, Y. Shen and B. C. Wang, "Sealed-bid auction scheme based on blockchain and secure multi-party computation," in *2021 IEEE 5th Information Technology, Networking, Electronic and Automation Control Conf. (ITNEC)*, Xi'an, China, vol. 5, pp. 407–412, 2021. https://doi.org/10.1109/ITNEC52019.2021.9587239

[53] H. P. Dai, J. Yu, M. Li, W. Wang, A. Liu *et al.,* "Bloom filter with noisy coding framework for multi-set membership testing," *IEEE Transactions on Knowledge and Data Engineering*, pp. 1–14, 2022. https://doi.org/10.1109/TKDE.2022.3199646