Hindawi
Security and Communication Networks
Volume 2021, Article ID 3963037, 13 pages
https://doi.org/10.1155/2021/3963037

Research Article

Efficient Cloud-Based Private Set Intersection Protocol with Hidden Access Attribute and Integrity Verification

Jianhong Zhang [1,2] **and Menglong Wu** [1]

[1]*School of Information Sciences and Technology, North China University of Technology, 100144 Beijing, China*
[2]*Guizhou University, Guizhou Provincial Key Laboratory of Public Big Data, Guiyang 550025, Guizhou, China*

Correspondence should be addressed to Jianhong Zhang; zjhncut@163.com

Received 14 May 2021; Revised 13 August 2021; Accepted 17 August 2021; Published 2 September 2021

Academic Editor: Marimuthu Karuppiah

1. Introduction

With the rapid progress of the technology of wireless communication and sensor, the Internet of Things (IoT) is changing our lives. From medical devices and air quality monitoring to intelligent street lights, energy-efficient buildings, smart home, and more, the IoT is in more places than ever. As the number of the connected IoT devices increases, the amount of data generated by these devices will also exponentially increase. According to the newer forecast from International Data Corporation (IDC) [1], there will be 41.6 billion connected IoT devices in 2025, and they will generate 79.4 ZB of data. For the IoT devices with limited computation and storage resources, it is an important challenge to how to properly use and store such vast amounts of data. Cloud computing makes it possible to process and store massive amounts of data, and it also makes those data users possessing limited resources to easily access the stored data in cloud at any time and from anywhere.

As a cryptographic tool that is realizing secure multiparty computation, private set intersection (PSI) can make two parties holding sets compute their intersection without revealing any information other than the intersection. Since PSI was proposed by Freedman in [2], a mass of PSI schemes [3–7] have been put forward. The powerful privacy protection of PSI makes it to be important applications in real life, such as private contact tracing [8], DNA testing and pattern matching [9], remote medical diagnostics [10], and the effectiveness assessment of online advertising [11]. Over the last few years, PSI has been further developed such that it becomes very practical with extremely fast implementations that can conduct millions of items in seconds. However, most PSI schemes require two parties possessing datasets to jointly calculate the intersection of the available datasets locally. As the commercial value of cloud computing services drives, the user might delegate cloud service provider to execute the PSI computation for the outsourced datasets in cloud.

To our knowledge, in most of the existing PSI protocols, both participants jointly compute the intersection of their sets in an interactive manner, which makes that each participant must have a local copy of its dataset. It brings a heavy burden to resource-limited users. The advent of cloud computing makes the delegation of PSI computation promising since cloud servers can provide flexible and cost-effective storage space and on-demand computing power service. Recently, several cloud-based PSI protocols [6, 12, 13] are proposed. In these schemes, to achieve PSI computation, the user needs to outsource its dataset to a cloud server. However, the cloud server is not fully trusted, and it might reveal or tamper with the items in the outsourced dataset. To ensure the privacy of the outsourced dataset, they should be processed by applying cryptographic algorithms before outsourcing. However, the complicated cryptographic operations do not only incur the heavy computation burden to the resource-constrained data users but also impede access over the dataset of the data owner. To

realize access control, data owners must be real-time online to execute PSI computations with the authorized data user.

Recently, to avoid data owner real-time online and achieve fine-grained access control, Ali et al. proposed an attribute-based private set intersection computation protocol [14]. However, their scheme cannot ensure the integrity of the returned blinded dataset from the cloud, since when the cloud is not fully trusted, it may return the partial blinded dataset or delete some items of the outsourced dataset. Additionally, in their scheme, the access policy needs to be embedded in the blinded dataset. This kind of direct exposure of access policy can result in privacy revelation of data users since the access policy often contains some sensitive information.

To solve the above issue and provide fine-grained access control, in the paper, we proposed an efficient cloud-based private set intersection computation (PSI) protocol. The main contributions in this works are summarized as follows:

(i) Fine-grained access control of PSI computation: it provides fine-grained access control for PSI computation in the cloud environment and makes access control over the outsourced dataset of data owners realized by applying attribute-based encryption.

(ii) Offline data owner: in the PSI-computation phase, data owner does not need to be online in real-time, which reduces the burdens of communication and computation of data owner.

(iii) Resisting colluding attack: the collusion between cloud server and unauthorized data users cannot obtain any information about the outsourced datasets.

(iv) Data secrecy: for data owner, an authorized data user with dataset Y only learns the information of the intersection $X \cap Y$; none of the other information about dataset X except $X \cap Y$ is obtained by data user.

(v) Integrity: it can ensure the integrity of the returned blinded dataset in order to resist the malicious behaviors of the cloud server.

(vi) Hidden access policy: to satisfy more practical privacy requirements, the proposed protocol enables that cloud server cannot derive any sensitive information about attribute from the blinded dataset.

1.1. Paper Organization. The remainder of the paper is organized as follows. Section 2 reviews related work, and Section 3 describes some preliminaries. In Section 4, we give problem formulation of the proposed protocol. In Section 5, the proposed PSI protocol is given. In Section 6, we analyze the security of the proposed protocol. In Section 7, we evaluate the performance of the proposed protocol. Finally, the paper is summarized in Section 8.

2. Related Work

Meadows proposed the first secure PSI protocol [15] based on multiplicative homomorphic techniques. Due to being based on public-key cryptography, the scheme running time is unacceptable, in particular, when the size of the dataset becomes large. In [2], Freedman et al. proposed a private set intersection protocol by means of partial homomorphic encryption and point-value polynomial representation of sets. Later, Hazay and Nizim extended it to the malicious setting in [16]. In [17], Kissner and Song proposed privacy-preserving set operations. Their scheme can compute not only the intersection of the sets but also the union of the sets in a privacy-preserving way.

In [18], Jarecki and Liu proposed a novel PSI protocol based on the composite residual problem. In their scheme, the user and the server obtain the intersection of the two datasets by using parallel oblivious pseudorandom function. However, it relied on a common reference model. In [19], De Cristofaro et al. proposed two PSI protocols in malicious model. However, their schemes are unable to hide the cardinality of the user's dataset. To overcome this problem, Ateniese et al. proposed a PSI protocol [20] under the RSA assumption. But their scheme is only proven to be secure in the random oracle model. Based on the scheme in [18], De Cristofaro and Tsudik presented an efficient PSI protocol [21] by using OPRF techniques. Over the past few years, many efficient PSI protocols [3–13, 16, 17] have been successively proposed.

According to cryptographic techniques used to construct PSI protocol, the existing PSI protocols are mainly classified into three different groups:

(i) Public-key-based PSI protocol: homomorphic encryption is a common cryptographic technique to design PIS protocol. In the early days, most PSI protocols were constructed based on homomorphic encryption, where the protocols in [2, 8, 10, 15, 22, 23] are the classic instances. In the type of protocols, data owner first encrypts dataset X to obtain the corresponding ciphertext C and sends C to data user, and then using homomorphic properties of homomorphic encryption, data user conducts some specific operations on the ciphertext C and its dataset Y. Finally, data owner obtains the corresponding intersection by using its private key. This type of protocol is suitable for the scenario in which both participants possess strong computing capability. In general, such protocols require a higher computation cost since public-key cryptography is included. However, it is suitable for designing some PSI protocols with a custom function.

(ii) Circuit-based PSI protocol: circuit-based generic technique of secure computation is another method to design PSI protocol. Fairplay proposed the first PSI protocol by using Yao's garbled-circuit approach in [24]. In the subsequent works, Huang et al. presented three PSI protocols based on Yao's

generic garbled-circuit method [25]. Their schemes are competitive with the fastest public-key-based protocols. Afterwards, Pinkas et al. gave some new optimizations for circuit-based PSI in [26]. By using secure multiparty computation idea, this type of protocol can transform the specific function into garbled Boolean circuit to realize secure computation, and its key technique is symmetric cryptography. For this general circuit protocol, its advantage is that it makes the design and implementation of the protocol easier. However, due to its generalization, the garbled circuit makes the scalability of the protocol poor.

(iii) Oblivious transfer- (OT-) based PSI protocol: oblivious transfer protocol is a foundation of secure computation. To realize large-scale data processing, Dong et al. presented two efficient PSI protocols [27] based on bloom filters and OT extension protocol. Their protocols are rather efficient and highly scalable compared with some PSI protocols. To reduce the runtime, Pinkas et al. gave an optimization of PSI protocol [27] using random OT extension in [26]. The core idea of this type of protocols is to have both parties collaboratively engage in many OT protocols. In general, this type of PSI protocols had lower computation costs and communication consumption, but extra keys-related computations are demanded such as secret key agreement.

From the above analysis, public-key-based PSI protocols exist as higher computation complexity, and circuit-based PSI protocol and OT-based PSI protocols have higher efficiency due to using symmetric encryption, but key agreement or secure transferring of secret keys also require additional computation costs and communication overhead.

3. Preliminaries

3.1. Composite Order Bilinear Group. Throughout the paper, we only consider composite order bilinear groups since our scheme is based on such construction. In the following, we review some concepts of such bilinear pair:

(1) $\mathbb{G}_1$ and $\mathbb{G}_T$ are two cyclic groups with the same composite order $N = pq$ where p, q are two distinct primes, and it is deemed to be hard for solving the discrete logarithm problem in group $\mathbb{G}_i, i \in \{1, T\}$.

(2) Let $\hat{e}$: $\mathbb{G}_1 \times \mathbb{G}_1 \longrightarrow \mathbb{G}_T$ denote a computable bilinear map which satisfies the following criteria:

(i) Bilinearity: for arbitrary $a, b \in Z_N$ and all $Q, F \in \mathbb{G}_1$, we have $\hat{e}(Q^a, F^b) = \hat{e}(Q, F)^{ab}$.

(ii) Nondegeneracy: $\exists g, h \in \mathbb{G}_1$ such that $e(g, h)$ has the order N in $\mathbb{G}_T$.

(iii) Orthogonal property: let $\mathbb{G}_p$ and $\mathbb{G}_q$ denote two subgroups of $\mathbb{G}_1$ with the order p and q, respectively. For $\forall h_p \in \mathbb{G}_p$ and $\forall h_q \in \mathbb{G}_q$, then $e(h_p, h_q) = 1$.

3.1.1. The Decisional Bilinear Diffie–Hellman Assumption in $\mathbb{G}$. Let $(g, g^a, g^b, g^c, T) \in \mathbb{G}_1^4$ and $\mathbb{G}_T$ be a random 5-tuple where $a, b, c \in Z_p$, there does not exist an efficient PPT algorithm $\mathcal{A}$ which can distinguish $T \stackrel{?}{=} e(g, g)^{abc}$. A's advantage of breaking the decisional Diffie–Hellman problem in $\mathbb{G}_1$ is defined as

$$\varepsilon = \Pr\left[e(g, g)^{abc} = T \leftarrow A\left(g, g^a, g^c, g^b\right)\right]. \quad (1)$$

We think that the DBDH problem is against $\mathcal{A}$ if the algorithm $\mathcal{A}$ is capable of distinguishing $e(g, g)^{abc}$ and T in a nonnegligible probability ε.

3.2. Access Tree. Access trees can make the representation of access control policies easier to understand. In what follows, we explain the access trees used in our constructions.

Access tree $\mathcal{T}$ is a tree-like access structure, and each leaf node is associated with an attribute value, and an inner-node l is represented with a threshold gate (num_l, k_l), where num_l is the children number of inner-node l and k_l is its threshold value satisfying $0 < k_l \leq \text{num}_l$. Specifically, when $k_l = 1$ and $k_l = \text{num}_l$, it means that the corresponding threshold gate is the OR-gate and the AND-gate, respectively. For each leaf node l, its threshold value is $k_l = 1$.

For the sake of presentation, the children of each node l are ordering from 1 to num_l. At the same time, we define that function parent(l) is the parent of node l, index(l) is the number associated with node l, and then function att(l) is an attribute associated with the leaf node l.

Assume that R is the root of an access tree $\mathcal{T}$, then we use $\mathcal{T}_R$ to represent this tree, and $\mathcal{T}_x$ is denoted by a subtree of $\mathcal{T}$ rooted at node x if node x is an inner node of $\mathcal{T}$. For an attribute set S, if it satisfies the subtree $\mathcal{T}_x$, then it is represented as $\mathcal{T}_x(S) = 1$. The satisfied conditions are divided into the following two cases:

(1) When x is a leaf node, $T_x(S) = 1$ is returned if and only if att$(x) \in S$;

(2) When x is a nonleaf node, $T_x(S)$ can be computed recursively. For all children z of node x, if at least k_x children satisfy $T_z(S) = 1$, then $T_x(S) = 1$ is returned.

4. Problem Formulation

In this section, to better understand the motivation of the proposed scheme, we will give the system model and threat model that correspond to our protocol.

4.1. System Model. For a cloud-based PSI computation protocol with fine-grained access control and integrity verification, its system model is shown in Figure 1. The system model consists of four entities: key generation center (KGC), cloud service provider (CSP), data owners, and data users. The roles of these entities are described as follows.

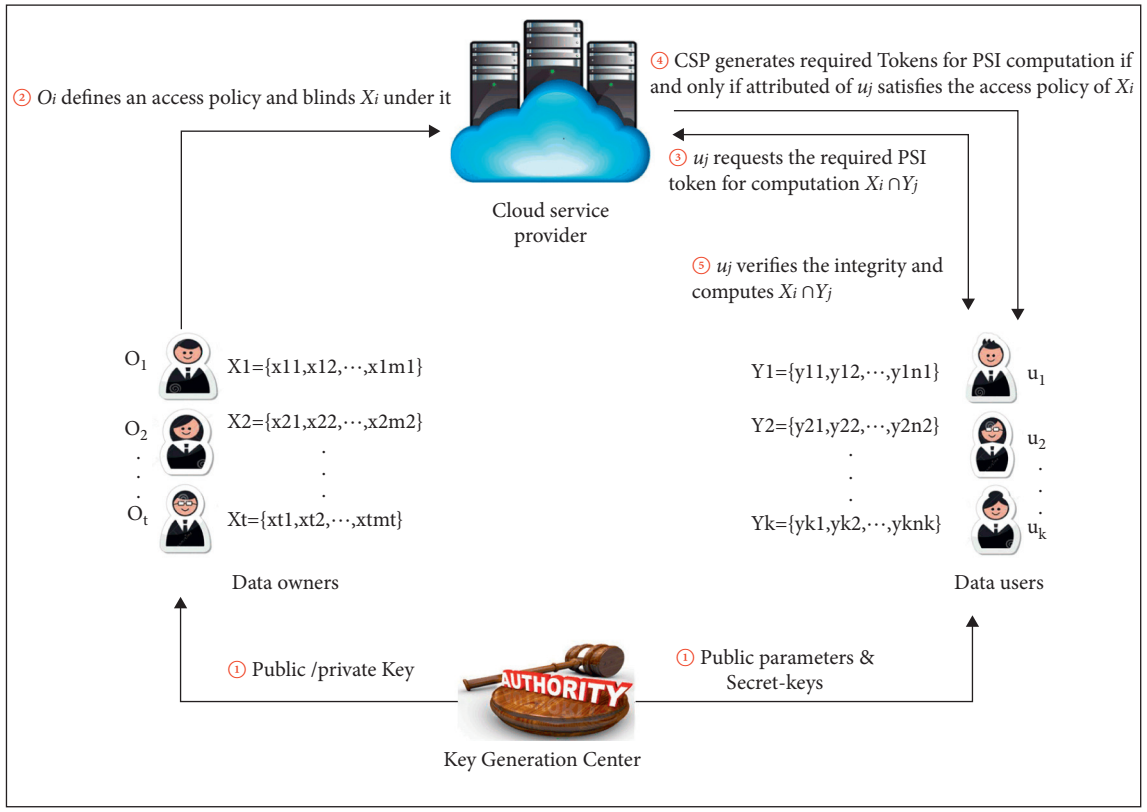

Figure 1: The system model of the proposed scheme.

4.1.1. Key Generation Center. It is responsible for establishing system parameters and generating the secret keys of the attributes for data users. In addition, it also generates a public-private key pair for the data owners' signature algorithm.

4.1.2. Cloud Service Provider. It has abundant storage space and powerful computing capability, and it can provide storage services of the outsourced dataset for data owners and PSI computation services for data users.

4.1.3. Data Owner. It is the owner of a dataset X. To achieve fine-grained access control, data owner needs to define an access control policy before outsourcing the dataset; and then it blinds the dataset based the defined access policy.

4.1.4. Data User. It also possesses a dataset Y and can request the CSP to generate a token in order to compute private set intersection $X \cap Y$. It is worth noting that only data user whose attributes satisfy access policy defined by data owner can obtain the returned PSI token by the CSP.

4.2. Threat Model. In our proposed protocol, the CSP is not a fully trusted entity, like [6, 11, 14, 17], it may attempt to tamper or delete the items in the outsourced dataset, and it also might try to extract sensitive information from the outsourced dataset by colluding some unauthorized data users. For data user, it is identified as a malicious entity. It may collude the CSP to obtain more information beyond the intersections. Additionally, an unauthorized data user also may attempt to obtain the qualification of the PSI computation. The KGC is assumed to be a trusted entity. It is responsible for generating secret keys for data user's attribute set and public/private key pairs for data owners. Data owner is assumed to be a trusted entity. It honestly encrypts its dataset and outsources them to the CSP. They never reveal the elements of their private datasets.

4.3. Security Goals. For the proposed protocol, its goals are given as follows:

(1) It achieves fine-grained access control and makes that only data users satisfying the defined access policy by data owner can conduct PSI computation.

(2) The authorized data users learn nothing, except the intersection of datasets, this is to say, it achieves data secrecy.

(3) The CSP does not learn any information about the protocol results and the outsourced dataset, this is to say, it is adaptively secure against chosen dataset attack.

(4) It must ensure the integrity of the blinded dataset returned by the cloud server to the data user.

5. Our Concrete Construction

In this section, we present a concrete construction of the cloud-based PSI protocol with fine-grained access and integrity verification. The protocol consists of the following three stages, the detailed descriptions are given as follows.

5.1. System Initiation. In this stage, key generation center (KGC) is responsible for initializing system parameters and producing secret key of the data user with attribute set Att. To do it, it needs to execute the following two algorithms: Setup and KeyGen.

5.1.1. Setup $(\lambda, \mathcal{U})$. Taking a security parameter λ and the universal attribute set $\mathcal{U}$ as inputs, it outputs $(\mathbb{G}_1, \mathbb{G}_T, N = p \cdot q, e) \leftarrow \mathrm{Gen}(1^\lambda)$, where $\mathbb{G}_1$ and $\mathbb{G}_T$ are cyclic groups of order N and p, q are two distinct 512-bit primes. Let $\mathbb{G}_p$ and $\mathbb{G}_q$ denote two subgroups of group $\mathbb{G}_1$ with the order p and q, respectively, and g_p and g_q are the generators of $\mathbb{G}_p$ and $\mathbb{G}_q$, respectively. e: $\mathbb{G}_1 \times \mathbb{G}_1 \longrightarrow \mathbb{G}_T$ is a bilinear pairing map.

For the universal attribute set $\mathcal{U} = (w_1, \ldots, w_n)$, where n is the maximum number of attributes, randomly choose $\alpha, t_1, t_2, \ldots, t_n \in Z_p$ and $R_0, R_1, \ldots, R_n \in \mathbb{G}_q$ to compute public keys

$$\begin{aligned} P_x &= g_p \cdot R_0, \\ P_y &= e\left(g_p, g_p\right)^\alpha, \\ T_i &= g_p^{t_i} \cdot R_i, \quad 1 \le i \le n. \end{aligned} \tag{2}$$

Additionally, KGC chooses two collision-resistant hash functions H and H_0 satisfying H: $\mathbb{G}_T \longrightarrow \{0,1\}^l$ and H_0: $\{0,1\}^* \longrightarrow \{0,1\}^l$. Finally, public parameters Param are published as follows:

$$\mathrm{Param} = \left(\mathbb{G}_1, \mathbb{G}_T, N, e, H, H_0, e, g_p, P_x, P_y, T_i (1 \le i \le n)\right), \tag{3}$$

and master secret keys $(\alpha, t_i, (1 \le i \le n))$ are securely stored. Note that each T_i corresponds to each attribute w_i.

5.1.2. KeyGen (Att, α, a, β). For a data user with an attribute set $\mathrm{Att} = (w_1, \ldots, w_t)$, if it wants to register to the system, it sends its attribute set Att to KGC, and then KGC makes use of its secret key to generate the secret key for the data owner with attribute set Att.

(1) First of all, KGC randomly chooses a number $r \in Z_p$ to compute

$$\mathrm{sk}_0 = g_p^{\alpha - r}, \tag{4}$$

(2) and then for each attribute w_j in Att, it computes $\mathrm{sk}_j = g_p^{r \cdot t_j^{-1}}$.

(3) The resultant secret key $\mathrm{sk}_{\mathrm{Att}} = (\mathrm{sk}_0, \{\mathrm{sk}_i\}_{w_i \in \mathrm{Att}})$.

In addition, for data owner, it randomly chooses $\mathrm{sk}_o \in Z_p$ to compute its public key $\mathrm{pk}_o = g_p^{\mathrm{sk}_o}$. This public-private key pair $(\mathrm{pk}_o, \mathrm{sk}_o)$ is used to generate digital signature.

5.2. Blinding of Dataset. For a data owner, if it wants to outsource the dataset $X = \{x_i\}_{i \in I}$ to cloud server, to ensure the security of the dataset, it needs to blind the dataset before outsourcing. To achieve the confidentiality and fine-grained access control of data, it needs to execute the following Blind algorithms.

5.2.1. Blind (Param, $\mathcal{T}$, X). For a dataset X and an access structure $\mathcal{T}$, the algorithm takes X and $\mathcal{T}$ as inputs and outputs the blinded data. The detailed process is given as follows.

(1) First of all, it picks a number $t \in Z_p$ and $R_0' \in \mathbb{G}_q$ at random to compute

$$\begin{aligned} C_0 &= P_x^t \cdot R_0', \\ C &= P_y^t, \\ \{C_i^x &= H_0\left(H\left(C \| \mathrm{pk}_0\right) \oplus H_0\left(x_i\right)\right)\}_{i \in I}, \end{aligned} \tag{5}$$

where $I = |X|$ is the size of the dataset X and $\|$ is a concatenate operator.

(2) Then, it generates a signature $\delta_s = (\delta_1, \delta_2)$ on $C_1^x \| \cdots \| C_{|I|}^x$ by randomly choosing $r_0 \in Z_N$, where $\delta_1 = g_p^{r_0}$ and $\delta_2 = r_0 - \mathrm{sk}_o \cdot H_1(C_1^x \| \cdots \| C_{|I|}^x, \delta_1) \bmod N$.

(3) To facilitate the expression of access structure, we adopt access tree $\mathcal{T}$. For each node v in T, the algorithm randomly allocates a $k_v - 1$ degree polynomial f_v in the top-down manner, where k_v is the children number of node v. Especially for root $R_{\mathcal{T}}$ of access tree T, the allocated polynomial $f_R(x)$ should satisfy $f_R(0) = t$; for the other node v, it should satisfy $f_v(0) = f_{\mathrm{parent}(v)}(\mathrm{index}(v))$, where $\mathrm{index}(v)$ denotes an index value of node v in the children of its parent node, and $\mathrm{parent}(v)$ is the parent node of node v. Finally, for each leaf node i of access tree T corresponding to an attribute w_i, the blinded data is computed

$$C_i = \left(T_i^{f_i(0)}\right) R_i', \tag{6}$$

where R_i' is a random element in group $\mathbb{G}_q$. Note that T_i corresponds to the attribute w_i.

(4) At last, the blinded dataset of X is $BT = (\mathcal{T}, \delta_s, C_0, \{C_i^x\}_{i \in I}, \{C_j\}_{j \in \mathcal{T}})$, and data owner uploads them to cloud server.

5.3. PSI Computation. To obtain private set intersection of a blinded dataset BT of dataset X uploaded by a special data owner, the stage is divided into three parts: Token1 generation, Token2 generation, and set intersection computation. Firstly, a data user with dataset Y needs to run TokenGen1 algorithm to produce a PSI-token 1 for cloud

server, and then cloud server runs TokenGen2 algorithm to produce a PSI-token 2 by using PSI-token 1. Finally, data user computes the intersection $X \cap Y$ of the two datasets X and Y by using the PSI-token 2 produced by cloud server. The detailed processes are given as follows.

5.3.1. TokenGen1 $(Param, sk_{Att})$. Taking system parameters Param and secret key of data user $\mathrm{sk}_{\mathrm{Att}}$, this algorithm randomly selects a number $\mu \in Z_p$ to compute a PSI-token 1:

$$\mathrm{ToK}_1 = \left(\mathrm{sk}_1^{\mu}, \ldots, \mathrm{sk}_2^{\mu}, \ldots, \mathrm{sk}_{|\mathrm{Att}|}^{\mu}, \mathrm{sk}_0^{\mu}\right). \tag{7}$$

5.3.2. TokenGen2 $(Param, ToK_1, BT)$. Taking the PSI-token 1 ToK_1 and the blinded dataset BT, this algorithm firstly verifies whether Att satisfies the access tree $\mathcal{T}$ correlating to the blinded dataset BT. If it does not, then it outputs $\perp$ and aborts it.

For the sake of illustration, we define a recursive algorithm $\mathrm{Tok}_2(\cdot)$ that takes as input an access tree $\mathcal{T}$, an attribute set Att, the blinded dataset BT, and a node from $\mathcal{T}$.

For each node in $\mathcal{T}$, cloud server executes a recursive algorithm $\mathrm{Tok}_2(\cdot)$ as follows:

(1) If $\theta_i \in \mathcal{T}$ is a leaf node and $\mathrm{att}(\theta_i) \in \mathrm{Att}$, where $w_i = \mathrm{att}(\theta_i)$ is an attribute corresponding to leaf node θ_i, then it computes

$$\begin{aligned}\mathrm{Tok}_2(\mathcal{T}, \mathrm{Att}, BT, \theta_i) &= e\left(C_i, \mathrm{sk}_i^{\mu}\right)\\ &= e\left(g_p^{t_i f_i(0)}, g_p^{rt_i^{-1}}\right)^{\mu} e\left(R_i^{f_i(0)} \cdot R_i', g_p^{rt_i^{-1}}\right)^{\mu}\\ &= e\left(g_p^{t_i f_i(0)}, g_p^{\mu r t_i^{-1}}\right)\\ &= e\left(g_p, g_p^{r}\right)^{\mu f_i(0)},\end{aligned} \tag{8}$$

where sk_i^{μ} is the i-th element in PSI-token 1 ToK_1 and C_i is the i-th in $\{C_j\}_{j \in \mathcal{T}}$.

(2) If $\theta_i \in \mathcal{T}$ is a nonleaf node, the recursive algorithm runs as follows: for all child node $\mathrm{child}(\theta_i)$ of node θ_i, it calls $\mathrm{Tok}_2(\mathcal{T}, \mathrm{Att}, BT, \mathrm{index}(j, \mathrm{child}(\theta_i)))$ for $j = 1, \ldots, \omega$ and stores them, where $\mathrm{child}(\theta_i)$ denotes all children of θ_i and $\mathrm{index}(j, \mathrm{child}(\theta_i))$ denotes the j-th child node of θ_i. The algorithm is run until children nodes of a node are leaf nodes, and then we compute

$$\mathrm{Tok}_2(\mathcal{T}, \mathrm{Att}, BT, \theta_i) = \prod_{k=1}^{\omega} \mathrm{Tok}_2\left(\mathcal{T}, \mathrm{Att}, BT, \mathrm{index}\left(k, \mathrm{child}(\theta_i)\right)\right)^{\Delta_{\omega,k}(0)}, \tag{9}$$

where $\mathrm{child}(\theta_i)$ is the child node of node θ_i and w is threshold value of parent node $\mathrm{parent}(\theta_i)$, and for θ_i, its Lagrange coefficient is

$$\Delta_{\omega,k}(x) = \prod_{j \in [1,\omega], j \neq k} \frac{x - j}{k - j}. \tag{10}$$

Through recursively running $\mathrm{Tok}_2(\mathcal{T}, \mathrm{Att}, BT, \theta_i)$ in the bottom-to-top manner in access tree $\mathcal{T}$, we can obtain the PSI–token 2,

$$\mathrm{Tok}_2 = e\left(g_p, g_p^{rt}\right)^{\mu}. \tag{11}$$

In the end, the algorithm returns the PSI-token 2 Tok_2, $\{C_i^x\}_{i \in I}$ and the signature δ_s of $\{C_i^x\}_{i \in I}$.

5.3.3. PSI Computation $(Param, ToK_2, \{C_i^x\}_{i \in I} Y = \{y_j\}_{j \in \{1,\ldots,|Y|\}}, \delta_s)$. Upon receiving ToK_2 and $\{C_i^x\}_{i \in I}$, the algorithm takes ToK_2, $\{C_i^x\}_{i \in I}$, Y, and δ_s as inputs and executes Algorithm 1. In this algorithm, lines 1–3 are used to check the validity of the signature δ_s, which can achieve the integrity checking of the returned dataset by cloud server. Lines 7–17 are used to seek the intersection of two datasets. If $C_i^x == \tau_j$ holds, then the element of the intersection is found.

5.4. Correctness. In the subsection, we show that the proposed scheme is correct, because in the blinding stage, the dataset $X = \{x_i\}_{i \in I}$ is blinded into BT, and each C_i^x in BT has the following format $C_i^x = H_0(H(C \| \mathrm{pk}_0) \oplus H_0(x_i))$.

In addition, in the stage of intersection computation, data user can use PSI-token 2 to obtain the following relation:

$$\begin{aligned}\mathrm{Tmp} &= \mathrm{Tok}_2^{\mu^{-1}} \cdot e\left(C_0, \mathrm{sk}_0\right)\\ &= e\left(g_p, g_p\right)^{rt} e\left(P_x^t \cdot R_0', g_p^{\alpha - r}\right)\\ &= e\left(g_p, g_p\right)^{rt} e\left(g_p^t, g_p^{\alpha - r}\right)\\ &= e\left(g_p, g_p\right)^{\alpha t} = P_y^t.\end{aligned} \tag{12}$$

Thus, it can obtain $X \cap Y$ by running Algorithm 1 in PSI computation phase. It means that the proposed scheme is correct.

6. Security Analysis

In this section, we show that the proposed PSI scheme satisfies data secrecy and resists the adaptively chosen-dataset attack.

Theorem 1. *Supposed that the decisional bilinear Diffie-Hellman problem (DBDHP) in $\mathbb{G}_1$ is difficult to solve, then the proposed PSI scheme is adaptively secure against chosen-dataset attack in the standard model.*

Input: Param, $\delta_s, \mu, \mathrm{pk}_0, \mathrm{ToK}_2, \{C_i^x\}_{i\in I}, Y = \{y_j\}_{j\in\{1,\ldots,|Y|\}}$
Output: the intersection $X \cap Y$ of dataset X and dataset Y
(1) **if** $(\delta_1! = g_1^{\delta_2} \cdot \mathrm{pk}_o^{H_1(C_1^x \| \cdots \| C_{|I|}^x, \delta_1)})$, **then**
(2) printf ("the returned dataset is not intact");
(3) exit (0);
(4) **end**
(5) Compute $\mathrm{Tmp} = \mathrm{Tok}_2^{\mu^{-1}} e(C_0, \mathrm{sk}_0)$;
(6) Set $y = |Y|$;
(7) **for** $i = 1$: I **do**
(8) **for** $j = 1$: y **do**
(9) Compute $\tau_j = H_0(H(\mathrm{Tmp} \| \mathrm{pk}_0) \oplus H_0(y_j))$;
(10) **if** $(C_i^x == \tau_j)$, **then**
(11) printf ("%d," y_j);
(12) **else**
(13) j++;
(14) **end**
(15) **end**
(16) i++;
(17) **end**

Algorithm 1: PSI computation.

Proof. Let A be a (probabilistic polynomial time) PPT adversary who launches an attack on the proposed PSI scheme. If it breaks the proposed PSI scheme in a non-negligible probability ϵ, then we are able to construct a challenger $\mathscr{CH}$ which solves the DBDHP problem.

First of all, let us recall the DBDHP problem in subgroup $\mathbb{G}_p$ of group $\mathbb{G}$. Assume that $\overrightarrow{Y} = (g_p, g_p^a, g_p^b, g_p^c, Z)$ is an instance of the BDHP problem, where a, b, c are random numbers and $Z \in_R \mathbb{G}_T$, its goal is to determine whether the case $e(g_p, g_p)^{abc} \stackrel{?}{=} Z$ holds. In the following interactive game, $\mathscr{CH}$ attempts to solve the BDHP problem by invoking A as a subroutine. □

6.1. Init Phase. In this phase, the adversary $\mathscr{A}$ randomly selects the challenged access structure $\mathscr{T}^*$ and an attribute set $\mathscr{U}$ and sends them to the challenger.

6.2. Setup. In this phase, $\mathscr{CH}$ initializes system parameters based on the instance of the DBDH problem. Firstly, it chooses $\alpha\prime \in Z_N$ to compute

$$P_y = e\left(g_p^a, g_p^b\right) e\left(g_p, g_p\right)^{\alpha\prime}. \tag{13}$$

It implies $\alpha = ab + \alpha'$.

Also, for $i = 1$ to n, it randomly chooses $t_i \in Z_p$, $R_0 \in \mathbb{G}_q$, and $R_i \in \mathbb{G}_q$ to compute

$$\forall w_i \in \mathscr{U}\colon T_i = \begin{cases} g_p^{b/t_i} \cdot R_i, & \text{text } w_i \notin \mathscr{T}^*, \\ g_p^{t_i} \cdot R_i, & \text{text } w_i \in \mathscr{T}^*, \end{cases} \quad 1 \le i \le n. \tag{14}$$

Additionally, it also sets $P_x = g_p \cdot R_0$ and chooses $\mathrm{sk}_o \in Z_p$ to set data owner's public key as $\mathrm{pk}_o = g_p^{\mathrm{sk}_o}$. Finally, it sends $(P_x, P_y, T_1, \ldots, T_n)$ to the adversary A.

6.3. Phase 1. To simulate the game, $\mathscr{A}$ can adaptively make a series of queries in this phase.

(i) KeyGen query: while the adversary A makes a KeyGen query with an attribute set $W = (w_1, \ldots, w_l)$, where $w_i \in \mathscr{T}^*$, to response it, CH executes the following steps:

(1) First of all, it randomly chooses $r' \in Z_p$ to compute

$$\begin{aligned} sk_0 &= g_p^{\alpha'} \left(g_p^b\right)^{-r'} = g_p^{\alpha' - r'b} \\ &= g_p^{\alpha' + ab - (ab + r'b)} = g_p^{\alpha - (ab + r'b)} \\ &= g_p^{\alpha - r}. \end{aligned} \tag{15}$$

It implicitly defines $r = ab + r'b$.

(2) For $i = 1$ to l, because all $w_i \notin \mathscr{T}^*$, we can compute

$$sk_i = \left(g_p^a\right)^{t_i} g_p^{r' t_i} = g_p^{r \cdot \left(b/t_i^{-1}\right)}. \tag{16}$$

(3) Finally, the secret key of the corresponding attribute $W\mathrm{sk}_W = (\mathrm{sk}_1, \ldots, \mathrm{sk}_l, \mathrm{sk}_0)$ is returned to the adversary A.

(ii) TokenGen1 queries: when the adversary $\mathscr{A}$ issues a TokenGen1 query with attribute W, $\mathscr{CH}$ first makes a KeyGen query with attribute W to obtain secret key $\mathrm{sk}_W = (\mathrm{sk}_1, \mathrm{sk}_2, \ldots, \mathrm{sk}_0)$, and then it chooses a random number ζ to compute a PSI-token 1:

$$\mathrm{ToK}_1 = \left(\mathrm{sk}_1^\zeta, \mathrm{sk}_2^\zeta, \ldots, \mathrm{sk}_{|W|}^\zeta, \mathrm{sk}_0^\zeta\right), \tag{17}$$

and it sends ToK_1 to the adversary.

(iii) Decryption queries: for a decryption query of the ciphertext CT with attribute set W, $\mathscr{CH}$ parses

access tree T from CT and checks whether W matches the access structure T. If it matches, then $\mathscr{CH}$ performs a KeyGen query with W to obtain sk_W, and then it makes use of sk_W to call Decryption algorithm to retrieve data M.

6.4. Challenge. In this stage, to produce a challenge, the adversary $\mathscr{A}$ sends the challenged access tree $\mathscr{T}^*$ in which l leaf nodes are involved and correspond to attributes $(w_1, w_2, \ldots, w_l)$ and the two dataset X_0 and X_1 to the challenger $\mathscr{CH}$, where $X_0 = \{x_i^0\}$ and $X_1 = \{x_i^1\}$ are of the same size.

Subsequently, $\mathscr{CH}$ randomly flips a coin $\gamma \in \{0, 1\}$ to produce the following blinded dataset:

(1) It sets

$$\begin{aligned} C_0^* &= g_p^c \cdot R_0 \cdot R_0', \\ C^* &= Z \cdot e\left(g_p^c, g_p^{\alpha'}\right), \\ &\left\{C_i^x = H_0\left(H\left(C^* \| \mathrm{pk}_o\right) \oplus H_0\left(x_i^\gamma\right)\right)\right\}_{i \in I}. \end{aligned} \tag{18}$$

(2) Assume that root denotes the root node of $\mathscr{T}^*$. According to the principle from top to bottom, to compute leaf node information, we first construct a polynomial $f_{\mathrm{root}}(x)$ satisfying $g_p^{f_{\mathrm{root}}(0)} = g_p^c$. Although $f_{\mathrm{root}}(x)$ is unknown, we can compute the exponential form of $f_{\mathrm{root}}(x)$, namely,

$$g_p^{f_{\mathrm{root}}(x)} = g_p^{t'} \cdot \prod_{i=1}^{\pi} g_p^{a_i x^i}, \tag{19}$$

where $a_i \in Z_p$ is the random number and π is a threshold value of root node in access tree $\mathscr{T}^*$. For the children node v_i of root node root, we can compute $g_p^{f_{v_i}(0)} = g_p^{f_{\mathrm{root}}(i)}$, where $f_{v_i}(x)$ represents the polynomial corresponding to node v_i that is the i-child of root node.

According to the above method, we can obtain the following values by applying the manner from top to bottom, namely,

$$C_i^* = \left(g_p^{c \cdot f_i(0)}\right)^{t_i} \cdot R_i, \quad \text{for } \forall w_i \in \mathscr{T}^*, \tag{20}$$

where $f_i(x)$ denotes the polynomial of leaf node i in access tree $\mathscr{T}^*$. Because $T_i = g_p^{t_i} \cdot R_i$ and R_i is a random element, we have

$$\begin{aligned} C_i^* &= \left(g_p^{c \cdot f_i(0)}\right)^{t_i} \cdot R_i \\ &= T_i^{c f_i(0)} \cdot R_i, \end{aligned} \tag{21}$$

(3) and then it produces a signature $\delta_s^* = (\delta_1^*, \delta_2^*)$ by randomly choosing t_0^*, where $\delta_1^* = g_p^{r_0^*}$ and $\delta_2^* = r_0^* - \mathrm{sk}_o \cdot H_1(C_1^x \| \cdots \| C_{|I|}^x)$.

(4) At last, the blinded dataset BT^* is $BT^* = (\delta_s^*, C_0^*, \{C_i^{'*}\}, \{C_i^x\}_{i \in I})$ which is returned to A_I.

Obviously, when $Z = e(g_p, g_p)^{abc}$, we have

$$\begin{aligned} C^* &= Z \cdot e\left(g_p^c, g_p^{\alpha'}\right) = e\left(g_p, g_p\right)^{abc + \alpha' c} \\ &= e\left(g_p, g_p\right)^{(ab + \alpha')c} \\ &= e\left(g_p, g_p\right)^{\alpha c} = P_y^c. \end{aligned} \tag{22}$$

Thus, the produced blinded one of the dataset X_γ by the above way is valid.

6.5. Phase 2. In this phase, A can still issue a series of new queries as in Phase 1, but the following restriction conditions must been satisfied:

(1) $\mathscr{T}^*$ is not allowed to make the KeyGen queries.

(2) The blinded dataset BT^* is not allowed to make TokenGen1 query.

6.6. Guess. Eventually, the adversary A returns its guess $\gamma\prime \in \{0, 1\}$. If $\gamma = \gamma'$, then it outputs true. Otherwise, false is returned.

From the point of the adversary A's view, the simulation of $\mathscr{CH}$ is indistinguishable from the real game. When Z is a random element of $\mathbb{G}_1$, the produced blinded dataset BT^* has the same distribution as the real blinded dataset. It is independent of the choice of dataset X_γ. In this case, the probability of A guessing γ' is

$$\Pr(\gamma = \gamma') = \frac{1}{2}. \tag{23}$$

When $Z = e(g_p, g_p)^{abc}$ holds, the produced blinded dataset BT^* is a valid one. If the adversary $\mathscr{A}$ breaks the proposed scheme in nonnegligible probability ε, then it means that the adversary can solve the DBDH problem in groups $\mathbb{G}_1$ with the following probability:

$$\Pr\left(g_p, g_p a, g_p^b, g_p^c, e\left(g_p, g_p\right)^{abc}, Z\right) \geq \frac{1}{2} \cdot \frac{1}{2} + \frac{1}{2}\varepsilon. \tag{24}$$

However, due to the difficulty of solving the DBDH problem in groups $\mathbb{G}_1$, thus the probability of the adversary A breaking the proposed scheme is negligible.

Theorem 2. *If it is infeasible to generate a message that yields a given hash value for one-way hash function $H(\cdot)$, then our proposed PSI scheme can satisfy data secrecy in the standard model.*

Proof. Suppose that there exists an adversary A_{II} which breaks data secrecy in the proposed scheme, then we can construct an algorithm B to solve the one-way problem of hash function. Firstly, we review the one-way problem of hash function. Given a hash value y and one-way cryptographic hash function H_0, its goal is to find a number η such that it satisfies $H_0(\eta) = y$. To break the one-way problem of hash function, the algorithm $\mathscr{B}$ needs to initialize system parameters and plays an interactive game with the adversary $\mathscr{A}_{\mathscr{IF}}$. The detailed processes are given as follows:

(i) Setup: the algorithm $\mathcal{B}$ takes a security parameter λ as inputs and outputs system parameters $(\mathbb{G}_1, \mathbb{G}_T, N = p \cdot q, e, \mathbb{G}_p, \mathbb{G}_q, g_p, g_q) \leftarrow \mathbf{Gen}(1^\lambda)$. For the universal attribute set $\mathcal{U} = (w_1, \ldots, w_n)$, randomly choose $\alpha, t_1, t_2, \ldots, t_n \in Z_p$ and $R_0, R_1, \ldots, R_n \in \mathbb{G}_q$ to compute public keys

$$\begin{aligned} P_x &= g_p \cdot R_0, \\ P_y &= e\left(g_p, g_p\right)^\alpha, \\ T_i &= g_p^{t_i} \cdot R_i, \quad 1 \le i \le n. \end{aligned} \tag{25}$$

Additionally, KGC chooses two collision-resistant hash functions H and H_0 satisfying H: $\mathbb{G}_T \longrightarrow \{0,1\}^l$ and H_0: $\{0,1\}^* \longrightarrow \{0,1\}^l$. Finally, public parameters

$$\text{Param} = \left(\mathbb{G}_1, \mathbb{G}_T, N, e, H, H_0, e, P_x, P_y, T_i (1 \le i \le n)\right) \tag{26}$$

are sent to the adversary A_{II}.

(ii) Phase 1: in this phase, the adversary A_{II} is able to adaptively issue KeyGen queries and TokenGen1 queries. When the adversary A_{II} issues such queries, the algorithm B runs KeyGen() and TokenGen1() to response them since it has master secret key.

(iii) Challenge: to produce a challenge, the adversary A_{II} submits an access tree $\mathcal{T}^*$ and $k-1$-element dataset $X^* = \{x_1^*, \ldots, x_{k-1}^*\}$ to the algorithm B. The algorithm B randomly chooses a number $r \in Z_p$ to compute the blinded dataset BT^*.

(1) For $i = 1$ to $k-1$, it computes $C_i^{x*} = H_0(H(P_y^r \| \text{pk}_0) \oplus H_0(x_i^*))$.

(2) When $i = k$, it sets $C_k^{x*} = y$. It means that there is a number $d^* \in Z_p$ which satisfies $y = H_0(H(P_y^r \| \text{pk}_0) \oplus H_0(d^*))$.

(3) For the other components of BT^*, they are computed by the Blind algorithm since the algorithm B possesses master secret key and the private key of data owner.

(4) Finally, it returns $BT^* = (\mathcal{T}^*, \delta_s^*, C_0^*, \{C_i^{x*}\}_{i \in \{1,\ldots,k\}}, \{C_i^*\}_{i \in \mathcal{T}^*})$ to the adversary A_{II}.

(iv) Phase 2: in this phase, the adversary A_{II} is still able to issue the same queries as those in Phase 1.

(v) Guess: at last, the adversary A outputs its guess x_k^* in the position k for the dataset X^* in the non-negligible probability (note that $x_k^* = d^*$ or $x_k^* \neq d^*$). If the adversary A_{II} wins the game, then it means that x_k^* satisfies the following relation:

$$y = H_0\left(H\left(P_y^r \| pk_0\right) \oplus H_0(x_k^*)\right). \tag{27}$$

Thus, given a hash value y, the algorithm can find a value $H(P_y^r \| \text{pk}_0) \oplus H_0(y^*)$ to satisfy

$$y = H_0\left(H\left(P_y^r \| \text{pk}_0\right) \oplus H_0(x_k^*)\right). \tag{28}$$

Obviously, it is in contradiction with the one-way property of hash function. Therefore, for two datasets X^* and Y^*, the adversary A can obtain nothing about $(Y^* \setminus (X^* \cap Y^*))$ or $(X^* \setminus (X^* \cap Y^*))$, except the intersection in $X^* \cap Y^*$. □

Theorem 3. *The proposed scheme can ensure the integrity of the returned blinded dataset by the cloud assuming that the underlying Schnorr signature is unforgeable.*

Proof. Suppose that there exists an adversary A_{III} which breaks the integrity problem of the returned blinded dataset in the proposed scheme, then a challenger can construct an algorithm B which can break Schnorr signature; its goal is to output a new message-signature.

Let $\mathcal{CH}$ be a challenger; to break the security of Schnorr signature, the challenger $\mathcal{CH}$ runs as follows:

(i) Setup: the algorithm $\mathcal{B}$ takes security parameter λ as input and outputs system parameters $(\mathbb{G}_1, \mathbb{G}_T, N = p \cdot q, e, \mathbb{G}_p, \mathbb{G}_q, g_p, g_q) \leftarrow \text{Gen}(1^\lambda)$. For the universal attribute set $\mathcal{U} = (w_1, \ldots, w_n)$, randomly choose $\alpha, t_1, t_2, \ldots, t_n \in Z_p$ and $R_0, R_1, \ldots, R_n \in \mathbb{G}_q$ to compute public keys

$$\begin{aligned} P_x &= g_p \cdot R_0, \\ P_y &= e\left(g_p, g_p\right)^\alpha, \\ T_i &= g_p^{t_i} \cdot R_i, \quad 1 \le i \le n. \end{aligned} \tag{29}$$

Additionally, KGC chooses two collision-resistant hash functions H and H_0 satisfying H: $\mathbb{G}_T \longrightarrow \{0,1\}^l$ and H_0: $\{0,1\}^* \longrightarrow \{0,1\}^l$, and it chooses a number $\text{pk}_o \in Z_p$ as public key. Finally, public parameters

$$\text{Param} = \left(\mathbb{G}_1, \mathbb{G}_T, N, e, H, H_0, e, P_x, P_y, \text{pk}_0, T_i (1 \le i \le n)\right) \tag{30}$$

are sent to the adversary A_{III}, and it keeps the master private key $(\alpha, t_i, (1 \le i \le n))$ secretly.

(ii) Blinding dataset queries: the adversary A_{III} is able to adaptively issue Blinding Dataset queries with a dataset $X = \{x_i\}_{i \in |X|}$ and an access tree $\mathcal{T}$. When receiving a query $(X = \{x_i\}_{i \in |X|}, \mathcal{T})$, the challenger executes as follows:

(1) It picks a number $t \in Z_p$ at random to compute

$$\begin{aligned} C_0 &= P_x^t \cdot R_0', \\ C &= P_{y,}^t \\ \{C_i^x &= H_0(H(C \| pk_0) \oplus H_0(x_i))_{i \in I}\}. \end{aligned} \tag{31}$$

(2) Then, it sends $C_1^x \| \cdots \| C_{|I|}^x$ to algorithm $\mathcal{B}$, and algorithm $\mathcal{B}$ makes a signing query to signing oracle in the Schnorr signature security game with string $C_1^x \| \cdots \| C_{|I|}^x$. After obtaining the returned signature δ_s, $\mathcal{B}$ returns it to the challenger $\mathcal{CH}$.

(3) Next, the challenger $\mathscr{CH}$ computes the following values by adopting to access trees $\mathscr{T}$:

$$C_i = \left(T_i^{f_i(0)}\right) R_i', \tag{32}$$

where R_i' is a random element in group $\mathbb{G}_q$.

(4) Finally, it returns the blinded dataset $BT = (\mathscr{T}, \delta_s, C_0, \{C_i^x\}_{i\in I}, \{C_i\}_{i\in\mathscr{T}})$ to the adversary A_{III}.

(iii) Output: eventually, the adversary A_{III} outputs $(\mathrm{Tok}_2^*, \{C_i^{*x}\}_{i\in I}, \delta_s^*)$. The adversary A_{III} wins the game if δ_s is a valid signature on $C_1^{*x}\|\cdots\|C_{|X|}^{*x}$ and the string $C_1^{*x}\|\cdots\|C_{|X|}^{*x}$ is never made signing query to $\mathscr{B}$. If A_{III} wins this game, then $\mathscr{CH}$ sets δ_s^* as the output of $\mathscr{B}$ algorithm. Because δ_s is a valid signature on $C_1^{*x}\|\cdots\|C_{|X|}^{*x}$ and never makes a signing query with string $C_1^{*x}\|\cdots\|C_{|X|}^{*x}$, it means that algorithm $\mathscr{B}$ successfully breaks the unforgeability of Schnorr signature.

Obviously, it is in contradict with the unforgeability of Schnorr signature. Thus, the proposed PSI protocol can ensure the integrity of dataset. □

Theorem 4. *Our proposed PSI scheme can achieve hidden access attributes.*

Proof. In the proposed scheme, to achieve the attributes anonymously, we use the orthogonality property of composite order bilinear groups. In the encryption phase, the random elements R_0 and R_j are introduced into C_0 and $C_{j_{j\in\mathscr{T}}}$ in the blinded dataset. It can effectively prevent some malicious attackers from testing the access policy by a possible access structure w_j' and guessing the access structure. Thus, it achieves hidden access attributes. □

7. Performance Analysis

In this section, we evaluate the efficiency of the proposed PSI protocol in terms of computational costs. To give a fair comparison, we abandon the comparison with the other PSI protocols. The reason is that the goals of the proposed PSI protocol support fine-grained access control with hiding attribute and ensure the integrity of the returned dataset, which differs from most of the existing PSI protocols in which their goals are to be securely against semihonest adversaries or to improve efficiency of the PSI protocol. To our knowledge, it is the first PSI protocol with supporting attribute hidden fine-grained access control and integrity verification. Additionally, experiment results also show that the proposed PSI protocol is quite efficient.

To illustrate the effectiveness of the proposed PSI scheme, we also implement the experiment simulation based on an Ubuntu 18.04 laptop computer with the Intel(R) Core(TM) 4130 CPU@3.40 GHz, 4 GB RAM. All algorithms are written with C language in Linux system. Because the proposed scheme is based on composite order bilinear pairing, we adopt "Type A1 pairing" in PBC Library which provides a level of security equivalent to 1024-bit discrete logarithm problem. To be simple, we assume that access tree $\mathscr{T}$ only includes a root node and n leaf nodes, namely, it is in the form of (w_1 **AND** w_2 **AND** $\cdots$ **AND** w_n), where w_i is an attribute.

In the proposed protocol, private set intersection is only computed by data user, and data owner does not need to participate in this phase. In the whole protocol, after the dataset is outsourced to cloud server, the data owner is offline. However, in many PSI protocols [6, 18, 20, 27], both of data owner and data user have to be online and have interaction with each other.

Here, we show the performance of the proposed PSI protocol by evaluating the execution-time overhead of each algorithm.

The setup algorithm is used to initialize system parameters; the required execution times are mainly determined by the cardinalities of the universal attribute set. Its running time increases as the cardinalities of the universal attribute set increase; the corresponding performance graphs are given in Figure 2(a). For the KeyGen algorithm, its performance is shown in Figure 2(b). According to Figure 2, we can find that the running time in the two algorithms is greatly affected by the size of the data user's attribute set. Thus, they are two slashes that increase monotonically. When the number of attributes is 60, these two algorithm's running time is approximately 1.2 seconds. It is acceptable. For Blind algorithm, its running time is mainly from generation of fine-grained access structure and linear to the size of attribute set. However, the size of cardinality of the dataset has little influence on the runtime of algorithm after access attributes are fixed, the reason is that we blind the dataset by adopting XOR operation in the blinding process, and XOR operation's runtime is negligible. Their performance graphs are shown in Figures 3(a) and 3(b). From Figure 3(b), we can know when attribute number is 20 and the cardinality of dataset is 2^{20}; the runtime of algorithm is only about 371.578 ms. For TokenGen1 algorithm and TokenGen2 algorithm, their runtime is linear to the size of the data user's attribute set, the reason is that the two algorithms correspond to decryption process of attribute-based encryption scheme. Their performance graphs are shown in Figures 4(a) and 4(b). For PSI Computation algorithm, the runtime of algorithm is only related to the cardinality of dataset; it is shown in Figure 5, and we can know that for the cardinality of dataset, from 2^{10} to 2^{18}, the runtime of algorithm hardly changes since only 3 exponentiation in $\mathbb{G}_1$, 1 pairing operator, $|Y|$ hash operators, and $|Y|$ XOR operators are needed in this algorithm. However, XOR operator and hash operator are two kinds of lightweight operations that take almost no time. When the cardinality of dataset is 2^{20}, PSI computation runtime is 296.369 ms. Thus, it is very suitable for the resource-limited data user.

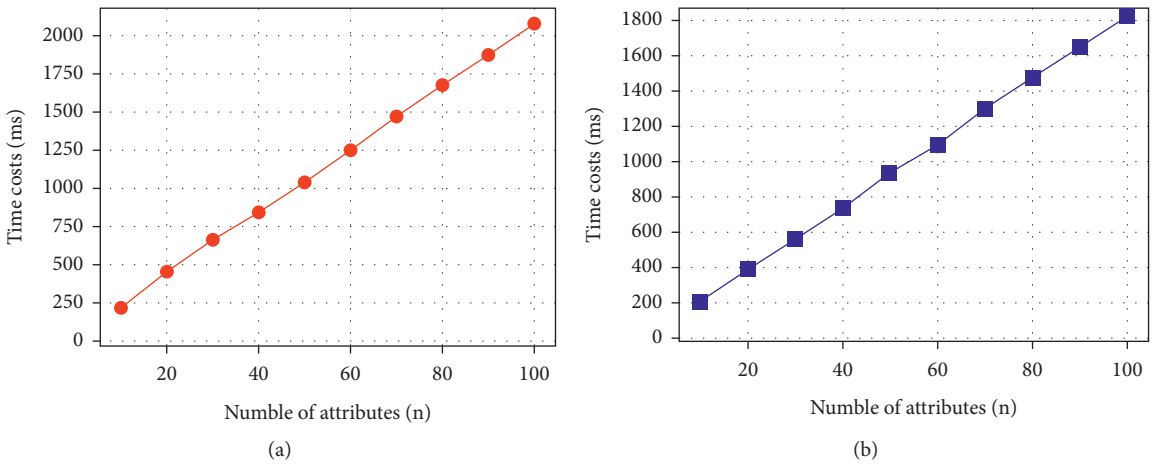

Figure 2: (a) Computational costs of system initialization. (b) Computational costs of KeyGen algorithm.

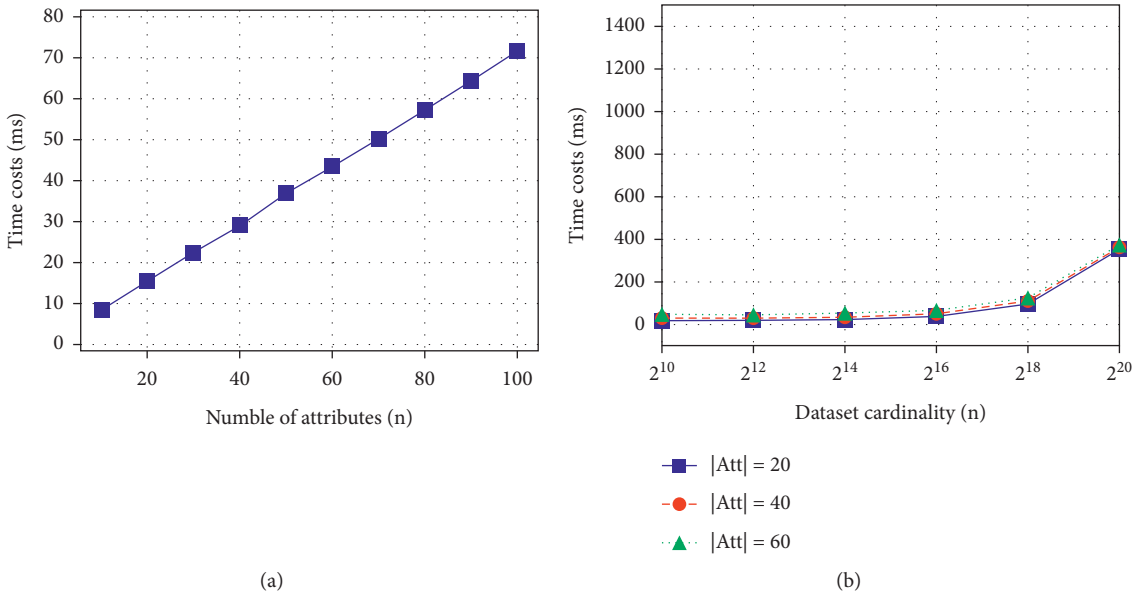

Figure 3: (a) The impact of the size of attributes on runtime in the Blind algorithm. (b) The impact of the cardinality of datasets on runtime in the Blind algorithm.

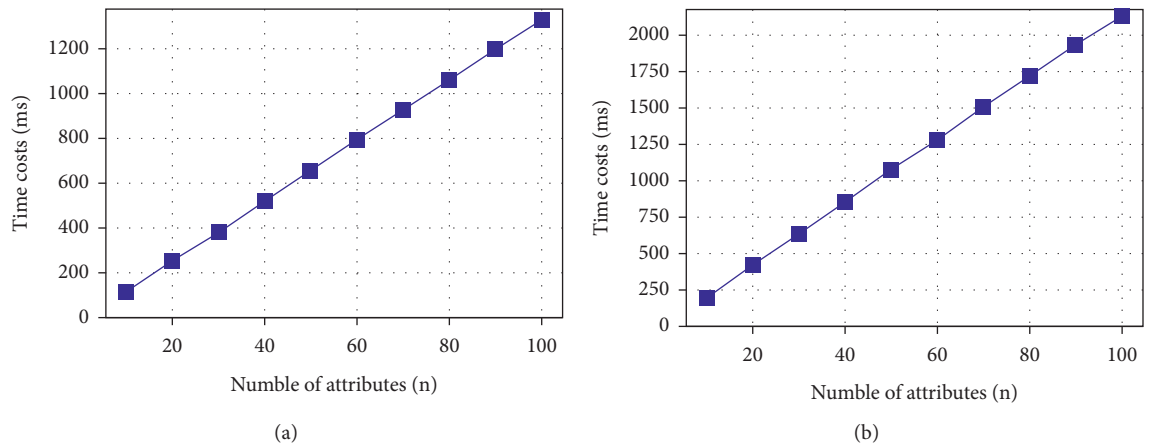

Figure 4: (a) The impact of the size of data user's attributes on runtime in the TokenGen1 algorithm. (b) The impact of the size of data user's attributes on runtime in the TokenGen2 algorithm.

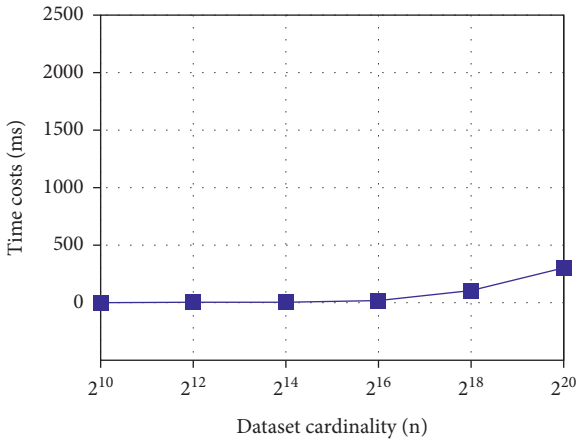

Figure 5: The impact of the cardinality of dataset on runtime in the PSI Computation algorithm.

8. Conclusion

In this work, we presented the first private set intersection (PSI) computation with supporting hidden attribute fine-grained access control and integrity verification based on attribute encryption. The main goal of the proposed scheme is to realize customized function to cater to practical application. Compared to most of the existing schemes, the proposed scheme has the following merits: (i) for data owner, it achieves dataset's access control by defining an access policy before the dataset is outsourced. (ii) It makes data owners to be offline during the whole PSI protocol. (iii) It ensures the integrity of the returned dataset from the cloud. (iv) The main PSI computation burden is transferred to the cloud. (v) It supports one-to-many PSI computations, that is to say, after the blinded dataset is outsourced to the cloud, data user can implement PSI computations with the cloud for arbitrary times. In addition, after giving the corresponding security analysis, we evaluate each algorithm of the proposed scheme by experiment simulation; the results show that the performance of the proposed scheme is efficient and practical. To reduce computational cost of the whole scheme, we will study fine-grained access control PSI scheme with constant computation in the future work.

Data Availability

This article contains the data that support the results of this study. If other data used to support the results of this study are needed, they can be obtained from the corresponding author.

Disclosure

The funders had no role in the design of the study; the collection, analyses, or interpretation of data; the writing of the manuscript; or the decision to publish the results.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by the Natural Science Foundation of Beijing (no. 4212019), National Natural Science Foundation of China (no. 62172005), Guangxi Key Laboratory of Cryptography and Information Security (no. GCIS201808), and Foundation of Guizhou Provincial Key Laboratory of Public Big Data (no. 2019BDKF JJ012).

References

[1] https://www.idc.com/getdoc.jsp?containerId=prUS45213219.

[2] M. J. Freedman, K. Nissim, and B. Pinkas, "Efficient private matching and set intersection," in *Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 1–19, Interlaken, Switzerland, June 2004.

[3] B. A. Huberman, M. Franklin, and T. Hogg, "Enhancing privacy and trust in electronic communities," in *Proceedings of the 1st ACM Conference on Electronic Commerce*, pp. 78–86, ACM Press, Denver Colorado USA, November 1999.

[4] C. Hazay and Y. Lindell, "Efficient protocols for set intersection and pattern matching with security against malicious and covert adversaries," in *Proceedings of the Theory of Cryptography Conference*, pp. 155–175, Springer, New York, NY, USA, March 2008.

[5] A. Kiss, J. Liu, T. Schneider, N. Asokan, and B. Pinkas, "Private set intersection for unequal set sizes with mobile applications," *Proceedings on Privacy Enhancing Technologies (PoPETs)*, vol. 2017, no. 4, pp. 97–117, 2017.

[6] S. Kamara, P. Mohassel, M. Raykova, and S. Sadeghian, "Scaling private set intersection to billion-element sets," in *Proceedings of the International Conference on Financial Cryptography and Data Security*, pp. 195–215, Springer, Christ Church, Barbados, March 2014.

[7] C. Hazay and M. Venkitasubramaniam, "Scalable multi-party private set-intersection," in *Proceedings of the IACR International Workshop on Public Key Cryptography*, S. Fehr, Ed., pp. 175–203, Amsterdam, The Netherlands, February 2017.

[8] H. Chen, L. Kim, and P. Rindal, "Fast private set intersection from homomorphic encryption," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pp. 1243–1255, Dallas, Texas, USA, October 2017.

[9] J. R. Troncoso-Pastoriza, S. Katzenbeisser, and M. Utku Celik, "Privacy preserving error resilient dna searching through oblivious automata," in *Proceedings of the 2007 ACM Conference on Computer and Communications Security*, pp. 519–528, Alexandria, Virginia, USA, October 2007.

[10] J. Brickell, D. E. Porter, V. Shmatikov, and E. Witchel, "Privacy-preserving remote diagnostics," in *Proceedings of the 14th ACM conference on Computer and communications security*, pp. 498–507, Alexandria, Virginia, USA, October 2007.

[11] M. Ion, B. Kreuter, E. Nergiz et al., *Private Intersection-Sum Protocol with Applications to Attributing Aggregate Ad Conversions*, ePrint Archive. 2017/738, 2017.

[12] F. Kerschbaum, "Outsourced private set intersection using homomorphic encryption," in *Proceedings of the 7th ACM Symposium on Information*, pp. 85-86, ACM, Seoul Korea, May 2012.

[13] F. Kerschbaum, "Collusion-resistant outsourcing of private set intersection," in *Proceedings of the 27th Annual ACM Symposium on Applied Computing*, pp. 1451–1456, ACM, Trento Italy, March 2012.

[14] M. Ali, J. Mohajeri, M.-R. Sadeghi, and X. Liu, "Attribute-based fine-grained access control for outscored private set intersection computation," *Information Sciences*, vol. 536, pp. 222–243, 2020.

[15] C. Meadows, "A more efficient cryptographic matchmaking protocol for use in the absence of a continuously available third party," in *Proceedings of the IEEE Symposium on Security and Privacy*, pp. 134–144, Oakland, CA, USA, April 1986.

[16] C. Hazay and K. Nissim, "Efficient set operations in the presence of malicious adversaries," *Journal of Cryptology*, vol. 25, no. 3, pp. 383–433, 2012.

[17] L. Kissner and D. Song, "Privacy-preserving set operations," in *Proceedings of the Annual International Cryptology Conference*, pp. 241–257, Springer, Berlin, Heidelberg, August 2005.

[18] S. Jarecki and X. Liu, "Efficient oblivious pseudorandom function with applications to adaptive ot and secure computation of set intersection," *Theory of Cryptography*, vol. 5444, pp. 343–356, 2009.

[19] E. De Cristofaro, J. Kim, and G. Tsudik, "Linear-complexity private set intersection protocols secure in malicious model," in *Proceedings of the International Conference on the Theory and Application of Cryptology and Information Security*, vol. 6477, pp. 213–231, LNCS, Singapore, December 2010.

[20] G. Ateniese, E. De Cristofaro, and G. Tsudik, "If size matters: size-hiding private set intersection," *Public Key Cryptograph-PKC*, vol. 6571, pp. 134–151, 2011.

[21] E. De Cristofaro and G. Tsudik, "Experimenting with fast private set intersection," in *Proceedings of the International Conference on Trust and Trustworthy Computing*, vol. 7344, pp. 55–73, LNCS, Vienna, Austria, June 2012.

[22] S. Lv, J. Ye, S. Yin et al., "Unbalanced private set intersection cardinality protocol with low communication cost," *Future Generation Computer Systems*, vol. 102, pp. 1054–1061, 2020.

[23] R. A. Mahdavi, T. Humphries, K. Bailey, and S. Krastnikov, "Practical over-threshold multi-party private set intersection," in *Proceedings of the Annual Computer Security Applications Conference (ACSAC 2020)*, pp. 772–783, Austin, USA, December 2020.

[24] D. Malkhi, N. Nisan, B. Pinkas, and Y. Sella, "Fairplay–a secure two-party computation system," in *USENIX Security Symposium*, 2004.

[25] Y. Huang, D. Evans, and J. Katz, *Private Set Intersection: Are Garbled Circuits Better than Custom protocols? In Network and Distributed System Security*, , pp. 1–15, NDSS'12), 2012.

[26] B. Pinkas, T. Schneider, and M. Zohner, "Faster private set intersection based on OT extension," in *The Proceedings of the 23rd USENIX Security Symposium*, pp. 797–812, 2014.

[27] C. Dong, L. Chen, and Z. Wen, "When private set intersection meets big data: an efficient and scalable protocol," in *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security*, pp. 789–800, ACM, Berlin Germany, November 2013.