10-2020

Attribute-based fine-grained access control for outscored private set intersection computation

Mohammad ALI
Amirkabir University of Technology-Iran

MOHAJERI Javad
Sharif University of Technology

Mohammad-Reza SADEGHI
Amirkabir University of Technology-Iran

Ximeng LIU
Singapore Management University, xmliu@smu.edu.sg

Attribute-based fine-grained access control for outscored private set intersection computation

Mohammad Ali[a] Javad Mohajeri[b] Mohammad-Reza Sadeghi[a] XimengLiu[cd]

[a] Department of Mathematics and Computer Science, Amirkabir University of Technology, Tehran, Iran
[b] Electronics Research Institute, Sharif University of Technology, Tehran, Iran
[c] College of Mathematics and Computer Science, Fuzhou University, Fuzhou 350108, PR China
[d] School of Information Systems, Singapore Management University, Singapore 178902, Singapore

Published in Information Sciences, Volume 536, October 2020, Pages 222-243
https://doi.org/10.1016/j.ins.2020.05.041

Abstract

Private set intersection (PSI) is a fundamental cryptographic protocol which has a wide range of applications. It enables two clients to compute the intersection of their private datasets without revealing non-matching elements. The advent of cloud computing drives the ambition to reduce computation and data management overhead by outsourcing such computations. However, since the cloud is not trustworthy, some cryptographic methods should be applied to maintain the confidentiality of datasets. But, in doing so, data owners may be excluded from access control on their outsourced datasets. Therefore, to control access rights and to interact with authorized users, they have to be online during the protocol. On the other hand, none of the existing cloud-based PSI schemes support fine-grained access control over outsourced datasets. This paper, for the first time, proposes an attribute-based private set intersection (AB-PSI) scheme providing fine-grained access control. AB-PSI allows a data owner to control intersection computations on its outsourced dataset by defining an access control policy. We also provide security definitions for an AB-PSI scheme and prove the security of our scheme in the standard model. We implement our scheme and report performance evaluation results.

Keywords

Fine-grained access control, Private set intersection, Cloud computing, Attribute-based encryption

1. Introduction

Using cloud computing [20], [27], [29], clients with limited computational and storage resources can outsource their personal data to a cloud service provider and, at a later time, ask it to perform computations on their data. This type of computing and storage paradigm enables users to obtain and release computing and storage resources rapidly. Thus, one can access such various and convenient resources on demand.

A private set intersection (PSI) scheme [12], [13], [34], [41] enables two clients to compute the intersection of their datasets, without revealing any information about the non-matched items. PSI has many real-world applications such as botnet detection [30], fully-sequenced human genome testing [7], privacy-preserving data mining [2], discovery of common friends [31], proximity testing [32], and user matching [17], etc.

Considering the advantages of cloud computing, one realizes that it is worthwhile to use its capabilities to improve the qualities of PSI protocols. In this regard, several cloud-based PSI protocols are given in some recent works [1], [21], [22], [23], [25], [35], [42].

In these works, clients outsource their datasets on a cloud service provider to benefit from its storage and computing services. Since the cloud is considered as a semi-trusted entity, to provide confidentiality, some cryptographic operations are performed on the datasets before outsourcing them. But, the cryptographic operations may cause a great challenge for data owners to control the access rights to their outsourced datasets. In this case, data owners have to be online to manage their data and to interact with authorized users in PSI computations, see Fig. 1(a). To the best of our knowledge, none of the existing works provide fine-grained access control over the outsourced encrypted data. Indeed, in these schemes, a data owner cannot determine the authorized users for PSI computation by defining an access control policy.

In the following, we present two application scenarios in which clients require to have access control on their outsourced data in PSI computation.

1.1. Motivation scenarios

1.1.1. Document similarity

Several editors of some scientific journals and conferences want to verify that none of the received manuscripts is under review in another journal or conference. To ease the accessibility of the documents, the editors outsource their received manuscripts to a cloud service provider. As the chairs are not allowed to share the received manuscripts, they encrypt their data before outsourcing them. They need a mechanism to approximate the similarities in a privacy-preserving way. In [10], PSI protocol is used to solve this problem. However, without access control on the data, the chairs have to be online to run PSI protocol with the other chairs which may be time-consuming and increase their administrative costs, see Fig. 1(a). While, if a PSI scheme providing fine-grained access control is used, then each chair can determine the desired user to compute the similarities of their manuscripts with his/her documents, by defining an access policy, and as long as he/she wants, he/she can be offline, see Fig. 1(b).

1.1.2. Search of similar patients in genomic data

The rapid development of genomic data affords a lot of new ways to improve medicine and research. However, it comes with burdens since it may cause some issues more noteworthy than its advantages. A genome can uniquely specify its owner and contains a lot of individual and sensitive information. The availability of genomic data provides several new ways to improve medicine and research. For example, a doctor holding the genome of his/her patient may interact with the other doctors around the world to find some individuals with similar genomic data and uses the individuals' data to diagnose and to find the effective treatment according to the patient?s conditions [5,40]. However, because of the privacy exposure implications, the genomic data must not be announced to the other doctors. By using a PSI protocol, doctors can find similar genomic data [3,40]. On the other hand, if they want to share their data via a cloud, they must encrypt their data before outsourcing them. But, the encryption operation may exclude them from controlling access rights on their data. Therefore, they may have to spend a lot of time to run the PSI protocols with the other doctors.

1.2. Technical roadmap

To address the mentioned problems, we introduce a cloud-based cryptographic primitive called *attribute-based private set intersection* (*AB-PSI*). Our scheme offers data confidentiality and fine-grained access control. In our scheme, by defining an access control policy, a data owner can control PSI computation on its dataset. Only data users whose attributes satisfy the access control policy can ask for intersection computation. Also, the cloud service provider cannot learn any information about the protocol results and contents of the datasets. Moreover, in our AB-PSI, the intersection computation does not need the cooperation of data owners, and they can be offline after outsourcing their datasets, see Fig. 1(b).

1.3. Contributions

The main contributions of this paper can be summarized as follows:

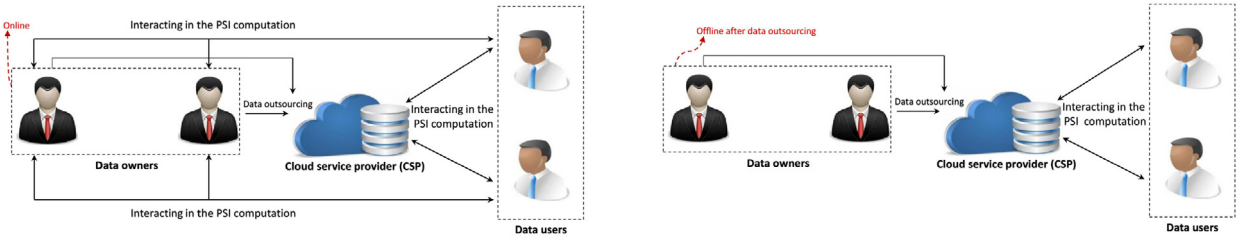

Fig. 1. (a) Dataset matching without fine-grained access control; (b) Dataset matching with fine-grained access control.

1. **Fine-grained access control**: We propose the first fine-grained access control system for private set intersection computation in cloud computing, called AB-PSI. In this system, a data owner can control the intersection computation on its outsourced dataset by defining an access control policy.
2. **Non-interactive PSI computation**: In the existing scheme, if one of the clients cannot interact with the other client, the PSI computation cannot be performed. Indeed, both clients must be online and interact with each other. It may not be desirable and may rise the administrative costs. In contrast with existing works, in our scheme, data owners can be offline after outsourcing their datasets, and the intersection computation can be done without their participation, see Fig. 1.
3. **One-to-many communication**: Our proposed system supports one-to-many communication; this means that an outsourced encrypted dataset can be used in an arbitrary number of PSI computations, such that no re-preparation on the encrypted data is required.
4. **Providing security definitions**: We provide formal security definitions for an AB-PSI scheme according to security aims in ABE schemes and PSI protocols. Also, two security definitions called *adaptively secure against chosen dataset attack* and *data secrecy* are defined. The first one is to capture that the cloud service provider and unauthorized data user colluding with each other are unable to learn any partial information about the contents of the outsourced datasets. The second one says that for a data owner with dataset X outsourced to the cloud and an authorized data user with dataset Y, the data user only can learn $X \cap Y$ and nothing else.
5. **Security and performance analysis**: According to the security definition, we formally prove that AB-PSI is secure in the standard model, under standard hardness assumptions. Also, we implement AB-PSI and evaluate the execution-time, storage cost, and communication overhead on the entities of the system.

1.4. Organization

The remaining of this paper is structured as follows: In Section 2, we introduce the related works on ABE schemes and PSI protocols. We give the preliminaries in Section 3. In Section 4, we provide the system model, threat model, and system and security definitions. The detailed construction is given in Section 5. In Sections 6 and 7, we analyze our proposed AB-PSI in terms of security and performance, respectively. Finally, we conclude this work in Section 8. The Appendix describes detailed correctness proof of AB-PSI system.

2. Related work

In this section, we review relevant prior works falling into two categories: (1) ABE schemes, and (2) PSI protocols.

2.1. ABE schemes

The concept of identity-based encryption (IBE) was initially put forth by Shamir [37] to reduce key and certificate management costs. In an IBE scheme, since the identity of users (such as their phone numbers or email addresses) are their public-keys, there is no need to verify the validity of the public-keys and certificates. The idea of IBE schemes inspired Sahai and Waters to design attribute-based encryption schemes [36] by replacing identity in IBE schemes with an attribute set. In an ABE scheme, a transmitter can specify authorized receivers by determining an attribute set and a threshold value. Users who the number of their common attributes with the specified attribute set is equal or more than the threshold value can recover the message. Subsequently, Goyal et al. improve the capability of ABE schemes by proposing the notion of key-policy ABE (KP-ABE) [18]. In a KP-ABE any ciphertext is labeled by an attribute set, and users' secret-keys are associated with an access control policy defined by the key-generator authority. A user's secret key is able to decrypt a ciphertext if and only if the attributes of the ciphertext satisfy the access control policy associated with the secret-key. In a KP-ABE scheme, access rights of data users are specified by the key generator authority. This feature may reduce control of data owners on their data. To address this problem, Bethencourt et al. proposed the concept of ciphertext-policy ABE (CP-ABE) scheme [9]. In a CP-ABE scheme, data users' secret-keys are associated with their attributes, and each ciphertext is associated with an access control policy defined by the transmitter. Secret-key of a data user can decrypt a ciphertext if and only if the data user's attributes satisfy the access control policy associated with the ciphertext. It seems that the features of CP-ABE schemes are more compatible with our design goals. So, we apply these schemes to this paper.

To realize fine-grained access control over outsourced encrypted data, attribute-based systems are used in a wide range of cryptographic schemes such as encryption schemes [4,29], searchable encryption schemes [27,28], signature schemes [11], signcryption schemes [8], authentication schemes [26], proxy re-encryption schemes [19], and broadcast encryption schemes [43]. However, to the best of our knowledge, the capabilities of attribute-based schemes in providing fine-grained access control have not been used in the could-based PSI schemes yet. Furthermore, there is no PSI scheme providing fine-grained access control.

2.2. PSI protocol

Private set intersection (PSI) was initially proposed by Freedman et al. [16]. Subsequently, some PSI protocols were introduced in [6,12–15,24,33]. In [24], several methods for computing union, intersection, and element reduction operations, in a

privacy-preserving way, was proposed. In [6,12–15] several PSI protocols with linear computation and communication complexities were designed. Among them, [15] provides a scalable, efficient PSI protocol based on oblivious Bloom intersection, and [6,14] afford PSI protocol hiding dataset sizes from the other entities. Later on, using permutation-based hashing, the scheme presented in [15] was improved in [33]. Nonetheless, in all the mentioned PSI protocols, clients jointly compute the intersections, and therefore, they have to be online when the protocol is run. In principle, it seems that these PSI protocols are not fit to realize security and privacy in outsourcing data and computations to a third party.

In [1,21–23,25,35,42], several PSI protocols enabling users to outsource data and computations to a third party (e.g., cloud computing) have been introduced. However, none of these works provides fine-grained access control over outsourced data, and clients have to interact with each other online, two by two, in PSI computations. Moreover, in schemes [21–23,25,35,42], the outsourced encrypted dataset to the cloud for PSI computing is disposable. This means that, if Alice has run the PSI protocol with Bob, then she has to re-prepare her dataset for running the protocol with Charlie. Also, In [25,35,42] the cloud service provider is able to understand whether the intersection of two datasets is empty or not. This is in contrast with one of the most basic security requirements in PSI protocols [6,12,13,24]. Moreover, the schemes [1,21,25] are secure only if the cloud service provider does not collude with the other parties. Indeed, in their scheme, an unauthorized data user can compute the intersection of its dataset with any data owner if he colludes with the cloud service provider. A comparison between AB-PSI and some existing cloud-based PSI protocol is given in Section 7. Table 4 summarizes characteristics comparison between our proposed scheme and some current cloud-based PSI schemes.

3. Preliminaries

Let $e \leftarrow S$ denote the random selection of an element e from a set S. In the following, we briefly introduce some fundamental concepts and assumptions needed for presenting our work.

3.1. Cryptographic structures and assumptions

Bilinear map: Let G_1 and G_2 be two cyclic groups of a prime order q. A function $\hat{e} : G_1 \times G_1 \rightarrow G_2$ satisfying the following properties is said to be a bilinear map:

- **Bilinearity**: For any two elements $a, b \in \mathbb{Z}_q$ and $g \in G_1$, we get $\hat{e}(g^a, g^b) = \hat{e}(g^b, g^a) = \hat{e}(g, g)^{ab}$.
- **Non-degeneracy**: There exists an element $g \in G_1$ that $\hat{e}(g, g) \neq 1$.
- **Computability**: There exists an efficient algorithm to compute $\hat{e}(g, h)$, for any $g, h \in G_1$.

Consider a probabilistic polynomial time (PPT) algorithm $\mathcal{G}$ that $(n, q, G_1, G_2, \hat{e}) \leftarrow \mathcal{G}(1^n)$, where n is the security parameter of the system, and q, G_1, G_2 and $\hat{e}$ are the same as above. In this paper, we consider the following cryptographic assumptions on $\mathcal{G}$:

Discrete Logarithm Assumption (DL): Given $(n, q, g, g^\alpha, G_1, G_2, \hat{e})$, where $(n, q, G_1, G_2, \hat{e}) \leftarrow \mathcal{G}(1^n), g \leftarrow G_1$, and $\alpha \leftarrow \mathbb{Z}_q$, this assumption says that any PPT adversary $\mathcal{A}$ can compute α with a negligible advantage in the security parameter n. In other words, for any PPT adversary $\mathcal{A}$, there exists a negligible function *negl* such that:

$$\Pr(\mathcal{A}(n, q, g, g^\alpha, G_1, G_2, \hat{e}) = \alpha) \leqslant negl(n), \tag{1}$$

where the probability is taken over $(n, q, G_1, G_2, \hat{e}) \leftarrow \mathcal{G}(1^n), g \leftarrow G_1, \alpha \leftarrow \mathbb{Z}_q$, and the randomness used by $\mathcal{A}$.

Decisional Bilinear Diffie Hellman Assumption (DBDH): Given $(n, q, g, g^\alpha, g^\beta, g^\gamma, g^z, G_1, G_2, \hat{e})$, where $(n, q, G_1, G_2, \hat{e}) \leftarrow \mathcal{G}(1^n), g \leftarrow G_1, \alpha, \beta, \gamma \leftarrow \mathbb{Z}_q$, and z either is equal to $\alpha\beta\gamma$ or $z \leftarrow \mathbb{Z}_q$, this assumption says that the advantage of any PPT adversary in determining the case of z is negligible in the security parameter n. In other words, for any PPT adversary $\mathcal{A}$, there exists a negligible function *negl* such that:

$$\left|\Pr\left(\mathcal{A}\left(n, q, g, g^\alpha, g^\beta, g^\gamma, g^{\alpha\beta\gamma}, G_1, G_2, \hat{e}\right) = 1\right) - \Pr\left(\mathcal{A}\left(n, q, g, g^\alpha, g^\beta, g^\gamma, g^z, G_1, G_2, \hat{e}\right) = 1\right)\right| \leqslant negl(n), \tag{2}$$

where the probabilities are taken over $(n, q, G_1, G_2, \hat{e}) \leftarrow \mathcal{G}(1^n), \alpha, \beta, \gamma, z \leftarrow \mathbb{Z}_q$, and the randomness of $\mathcal{A}$.

3.2. Access trees

Access trees are convenient to represent access control policies [18]. In an access tree, any leaf node is associated with an attribute, and each inner node represents a threshold value. The threshold value of each leaf node is assumed to be 1. Consider an access tree $\mathcal{T}$. Let v_a denote the leaf node associated with the attribute a, k_v denote the threshold value of a node v, ch_v denote the childeren set of a node $v, R_\mathcal{T}$ denote the root node of $\mathcal{T}, L_\mathcal{T}$ denote the leaf node set of the access tree $\mathcal{T}$, and $\mathcal{T}_v$ denote a subtree of $\mathcal{T}$ rooted at a node v.

Let $\mathbb{U}$ be the universal attribute set. For an access tree $\mathcal{T}$ and a node v of the tree, consider a function $F_{\mathcal{T}_v} : 2^\mathbb{U} \rightarrow \{0, 1\}$, where for an attribute set *Att* the evaluation of $F_{\mathcal{T}_v}(Att)$ is performed as follows:

- When v is a leaf node corresponding to an attribute $a, F_{\mathcal{T}_v}(Att) = 1$ if and only if $a \in Att$.
- When v is an inner node with threshold value $k_v, F_{\mathcal{T}_v}(Att) = 1$ if and only if there exist at least k_v children $c_1, \ldots, c_{k_v}$ of v that $F_{\mathcal{T}_{c_i}}(Att) = 1$, for $i = 1, \ldots k_v$.

We say that an attribute set Att satisfies an access tree $\mathcal{T}$ and denote it by $F_{\mathcal{T}}(Att) = 1$ if $F_{\mathcal{T}_{R_{\mathcal{T}}}}(Att) = 1$. Also, we use $F_{\mathcal{T}}(Att) = 0$ to indicate that Att does not satisfy $\mathcal{T}$.

Consider a prime number q, an access tree $\mathcal{T}$, and a secret $r \in \mathbb{Z}_q$. We denote an algorithm for sharing the secret r according to $\mathcal{T}$ and q as:

$$\{q_v(0)\}_{v \in L_{\mathcal{T}}} \leftarrow \textbf{Share}(\mathcal{T}, q, r). \tag{3}$$

For any node v in $\mathcal{T}$, this algorithm assigns a $(k_v - 1)$-degree polynomial with coefficients in $\mathbb{Z}_q$ to v in a top-down style as follows:

- It assigns a $(k_{R_{\mathcal{T}}} - 1)$-degree polynomial $q_{R_{\mathcal{T}}}$ to the root node $R_{\mathcal{T}}$ such that $q_{R_{\mathcal{T}}}(0) = r$, and the rest of its coefficients are chosen randomly from $\mathbb{Z}_q$.
- For any node v with polynomial q_v and children set $ch_v = \{c_1, \ldots c_{|ch_v|}\}$, it generates a $(k_{c_i} - 1)$-degree polynomial q_{c_i} for c_i such that $q_{c_i}(0) = q_v(i)$, and the rest of its coefficients are chosen randomly from $\mathbb{Z}_q$, for any $i = 1, \ldots, |ch_v|$.

When this algorithm stops, it associates a value $q_v(0)$ to each $v \in L_{\mathcal{T}}$.

Given $(n, q, G_1, G_2, \hat{e}) \leftarrow \mathcal{G}(1^n)$, a secret $r \in \mathbb{Z}_q$, an access tree $\mathcal{T}$, an attribute set S which its elements are associated with some leaf nodes in $\mathcal{T}$, $\{q_v(0)\}_{v \in L_{\mathcal{T}}} \leftarrow \textbf{Share}(\mathcal{T}, q, r)$, two elements $g_1, g_2 \in G_1$, and a value set $\left\{\hat{e}(g_1, g_2)^{q_{v_a}(0)}\right\}_{a \in S}$, we denote the algorithm for recovering $\hat{e}(g_1, g_2)^r$ through the value set as:

$$\hat{e}(g_1, g_2)^r \leftarrow \textbf{Combain}\left(\mathcal{T}, q, \left\{\hat{e}(g_1, g_2)^{q_{v_a}(0)}\right\}_{a \in S}\right). \tag{4}$$

It executes the following steps in a bottom-top fashion according to the access tree $\mathcal{T}$ as follows:

- If Att does not satisfies $\mathcal{T}$, then this algorithm aborts.
- Otherwise, it executes the following steps:
  - [–] It assigns the value $\hat{e}(g_1, g_2)^{q_{v_a}(0)}$ to the leaf node v_a.
  - [–] For any inner node v, if there exist k_v children c_{i_j} getting a value $\hat{e}(g_1, g_2)^{q_{c_{i_j}}(0)}, j = 1, \ldots, k_v$, then it computes:

$$\prod_{j=1}^{k_v}\left(\hat{e}(g_1, g_2)^{q_{c_{i_j}}(0)}\right)^{l_{i_j}} = \hat{e}(g_1, g_2)^{\sum_{j=1}^{k_v} l_{i_j} q_{c_{i_j}}(0)} = \hat{e}(g_1, g_2)^{q_v(0)}, \tag{5}$$

  where $l_{i_j} = \prod_{\substack{t=1 \\ i_t \neq i_j}}^{k_v} \frac{-i_j}{i_t - i_j}$. (Note that, from the basic fact about polynomial interpolation technique, we have $\sum_{j=1}^{k_v} l_{i_j} q_{c_{i_j}}(x) = q_v(x)$, for any $x \in \mathbb{Z}_q$). Then, it assigns $\hat{e}(g_1, g_2)^{q_v(0)}$ to v.

If $\mathcal{T}$ is satisfied by Att, then this algorithm returns $\hat{e}(g_1, g_2)^r$.

4. Problem formulation

In this section, we first introduce the system model and threat model. Then, we define our proposed AB-PSI system and its security model.

4.1. System model

Given a cryptographic cloud storage system which supports both PSI computation and fine-grained access control, in this paper, we consider a system consists of a Central Authority (CA), a Cloud Service Provider (CSP), and several Data Owners and Data Users, see Fig. 2. The tasks of the mentioned entities are described in more details below:

1. **CA**: It initializes the public parameters of the system, and it is responsible for delegating the data users' secret-keys, according to their attributes.
2. **Data owners**: To prevent illegal access to its data, each data owner defines an access control policy on its dataset; it blinds the dataset under the access policy by using the public-parameters of the system, and it outsources the blinded datasets to the cloud.

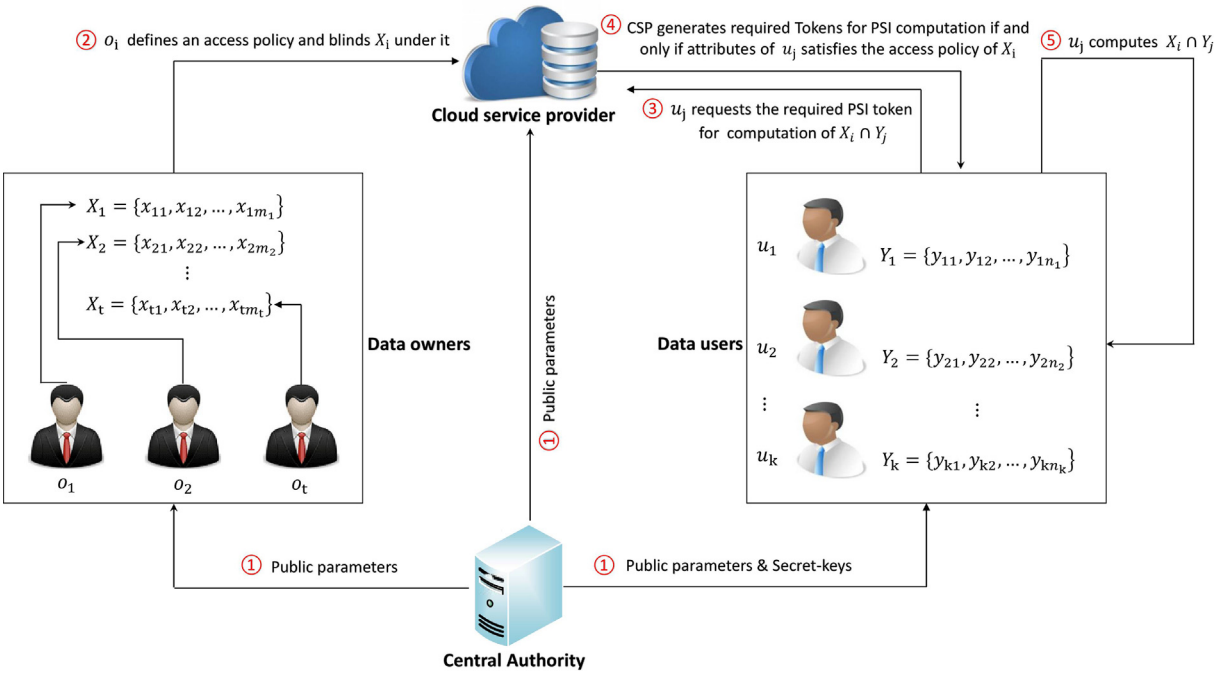

Fig. 2. Basic AB-PSI system model.

3. **Data users**: A data user whose attributes satisfy an access control policy defined by a data owner can request the CSP to generate the required token for PSI computation. Using the obtain token, the data user can compute the intersection between its dataset and the data owner's dataset.
4. **CSP**: The CSP provides storage, fine-grained access control, and PSI computation services for data owners and data users of the system.

In our proposed AB-PSI scheme, the intersection of two datasets is computed according to the following three rules:

- The access policy of the outsourced blinded dataset must be satisfied by the data user's attributes.
- Authorized data users learn the intersection of two datasets and nothing else.
- The CSP learns nothing about the contents of the two datasets, and it does not learn any information about the result of the PSI protocol. Indeed, the CSP cannot understand whether the intersection of two datasets is empty or not.

4.2. Threat model

Let X and Y be a data owner's dataset and a data user's dataset, respectively. In this work, the CSP is a semi-trusted entity [27,29,38]. It executes the given protocols correctly. But, it may try to derive some sensitive valuable information through collusion with some unauthorized data users or statistical analysis of the outsourced data. Unauthorized data users are assumed to be malicious. They may collude with each other and the CSP to get information about the outsourced datasets. But, authorized data users are semi-trusted. They do not collude with the other parties and do not reveal the contents of $X \cap Y$ and Y. But, they are always curious to obtain elements of $X \setminus Y$. The CA is assumed to be trusted. It assigns secret-keys of data users according to their attributes and does not give illegal access rights to them, at all. Also, it never colludes with the other parties. Data owners are assumed to be trusted. They never reveal the elements of their privet datasets. Table 1 summarizes the threat model of our proposed system.

4.3. Overview of our proposed AB-PSI

Our proposed scheme consists of the following six algorithms shown in Fig. 3 divided into four phases: **System setup**, **User registration and Key delegation**, **Dataset blinding**, and **PSI computation**. In the following, based on the introduced system model, we define our proposed scheme. The notations used in system definition and the construction presented in Section 5 are listed in Table 2.

Definition 1. An attribute-based privet set intersection (AB-PSI) scheme is a tuple of six PPT algorithms $\Pi = (\mathbf{Setup}, \mathbf{KeyGen}, \mathbf{Blind}, \mathbf{TokenGen}_1, \mathbf{TokenGen}_2, \mathbf{PSI})$ defined as follows:

Table 1
The treat model summery.

Entity	Type	Colludes with	Goals in its attack
CSP	Semi-trusted	Unauthorized data users	Learning the contents of X and the results of the protocol
Authorized data users	Semi-trusted	No one	Learning the contents of X/Y
Unauthorized data users	Malicious	CSP and data users	Learning the contents of X
CA	Trusted	–	–
Data owners	Trusted	–	–

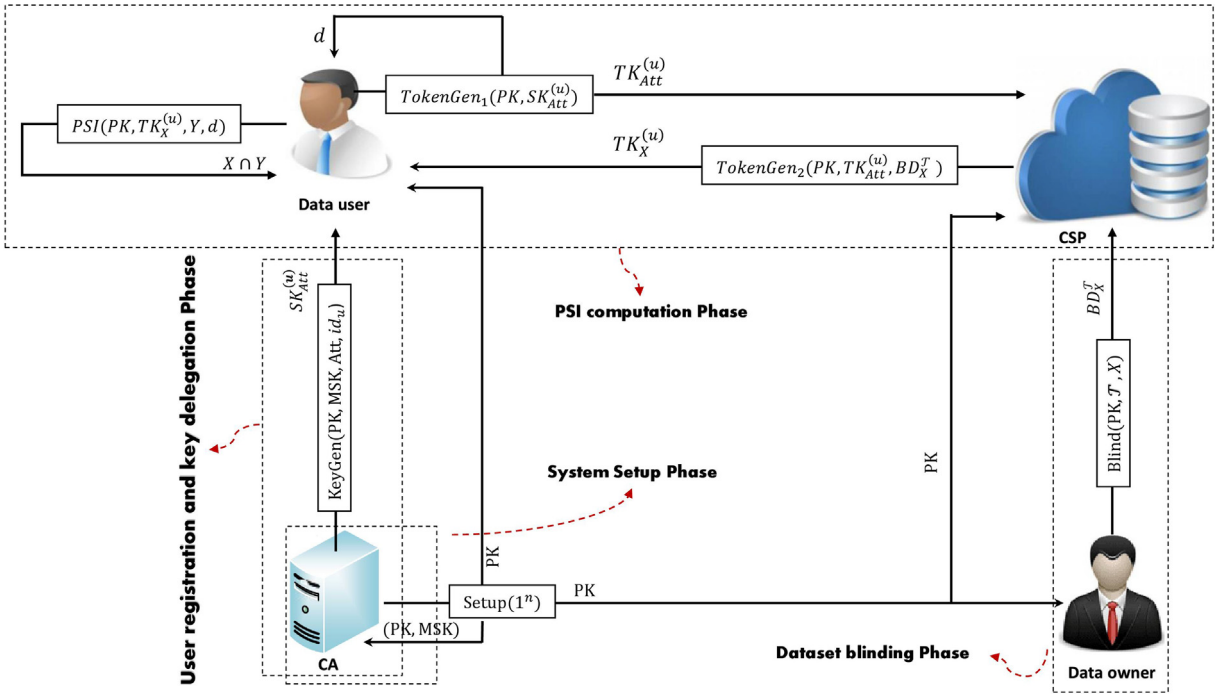

Fig. 3. Workflow of our proposed AB-PSI scheme.

Table 2
Notations.

Notation	Description
$\mathbb{U}$	The universal attribute set
n	Security parameter of the system
PK	Public parameters of the system
MSK	Master secret-key of the system
$\mathbf{u}$	A data user
$id_{\mathbf{u}}$	Identifier of the data user $\mathbf{u}$
Att	Attribute set of a data user
$SK_{Att}^{(\mathbf{u})}$	Secret-key of a data user $\mathbf{u}$ corresponding to an attribute set Att
$\mathcal{T}$	An access tree
$X = \{x_i\}_{i \in I}$	A data owner's dataset, where I is a finite index set
$Y = \{y_i\}_{j \in J}$	A data user's dataset, where J is a finite index set
$BD_X^{\mathcal{T}}$	Blinded dataset corresponding to a dataset X and an access tree $\mathcal{T}$
$TK_{Att}^{(\mathbf{u})}$	PSI token generated by $\mathbf{TokenGen}_1$ algorithm
d	A private-key generated by $\mathbf{TokenGen}_1$ algorithm
$TK_X^{(\mathbf{u})}$	PSI tokens generated by $\mathbf{TokenGen}_2$ algorithm
$\mathcal{A}$	A probabilistic polynomial time adversary
$\mathtt{AC-DA}_{\mathcal{A},\Pi}(n)$	Adaptively Chosen-Dataset Attack (AC-DA) indistinguishability experiment
$\mathtt{PC}_{\mathcal{A},\Pi}(n)$	Plaintext computing (PC) experiment

1. $(PK, MSK) \leftarrow$ **Setup** $(1^n, \mathbb{U})$: The CA operates this algorithm. It takes a security parameter 1^n and the universal attribute set $\mathbb{U}$ as inputs and initializes the public parameters, PK, and the master secret-key, MSK, of the system.
2. $SK_{Att}^{(\mathbf{u})} \leftarrow$ **KeyGen** $(PK, MSK, Att, id_{\mathbf{u}})$: The CA executes this algorithm. Inputs of the algorithm are public parameters and maser secret-key of the system; an attribute set Att, and an identifier $id_{\mathbf{u}}$ of a data user. It outputs a secret-key $SK_{Att}^{(\mathbf{u})}$ corresponding to Att and $id_{\mathbf{u}}$.
3. $BD_X^{\mathcal{T}} \leftarrow$ **Blind** $(PK, \mathcal{T}, X)$: Data owners run this algorithm. It takes public parameters PK, an access tree $\mathcal{T}$, and a dataset X as inputs. The output of the algorithm is a blinded dataset $BD_X^{\mathcal{T}}$ corresponding to the dataset X and the access tree $\mathcal{T}$.
4. $\left(TK_{Att}^{(\mathbf{u})}, d\right) \leftarrow$ **TokenGen**$_1$ $\left(PK, id_{\mathbf{u}}, SK_{Att}^{(\mathbf{u})}\right)$: This algorithm is run by a data user of the system. It takes public parameters of the system, and secret-key and identifier of the data user running this algorithm. It returns the corresponding PSI token $TK_{Att}^{(\mathbf{u})}$ and its associated private-key d.
5. $TK_X^{(\mathbf{u})} \leftarrow$ **TokenGen**$_2$ $\left(PK, TK_{Att}^{(\mathbf{u})}, BD_X^{\mathcal{T}}\right)$: The CSP executes this algorithm. Taking public parameters PK, a PSI token $TK_{Att}^{(\mathbf{u})}$, and a blinded dataset $BD_X^{\mathcal{T}}$, this algorithm returns the corresponding PSI token $TK_X^{(\mathbf{u})}$ or an error message $\perp$.
6. $X \cap Y \leftarrow$ **PSI** $\left(PK, TK_X^{(\mathbf{u})}, Y, d\right)$: By using this algorithm, a data user with a dataset Y can compute the intersection of its dataset with a data owner's dataset. On inputs public parameters PK, a valid PSI token $TK_X^{(\mathbf{u})}$, a dataset Y, and a private-key d, this algorithm outputs $X \cap Y$.

Definition 2. An AB-PSI system is said to be correct if for any two given datasets X and Y, $(PK, MSK) \leftarrow$ **Setup**$(1^n, \mathbb{U})$, access tree $\mathcal{T}$ satisfied by an attribute set Att, secret-key $SK_{Att}^{(\mathbf{u})} \leftarrow$ **KeyGen**$(PK, MSK, Att, id_{\mathbf{u}})$, blinded dataset $BD_X^{\mathcal{T}} \leftarrow$ **Blind**$(PK, \mathcal{T}, X)$, and PSI tokens $TK_{Att}^{(\mathbf{u})} \leftarrow$ **TokenGen**$_1\left(PK, id_{\mathbf{u}}, SK_{Att}^{(\mathbf{u})}\right)$ and $TK_X^{(\mathbf{u})} \leftarrow$ **TokenGen**$_2\left(PK, TK_{Att}^{(\mathbf{u})}, BD_X^{\mathcal{T}}\right)$, we have:

$$\mathbf{PSI}\left(PK, TK_X^{(\mathbf{u})}, Y, d\right) = X \cap Y. \tag{6}$$

4.4. Security models

AB-PSI security requires that the CSP does not learn any information about the protocol results and contents of the outsourced datasets, the PSI computation is not possible for unauthorized data users, and any authorized data user only can learn the intersection of the two datasets and nothing else. More precisely, an AB-PSI scheme should satisfy the following security requirements:

1. **Adaptively secure against chosen dataset attack**: This requirement is to capture any PPT adversary $\mathcal{A}$ modeling the semi-trusted CSP colluding with unauthorized data users is unable to deduce any partial information about the contents of a data owner's dataset from the corresponding blinded datasets and an arbitrary number of unmatched secret-keys and matched PSI tokens. Equivalently, any adversary allowing to adaptively ask for unauthorized data users' secret-keys and matched PSI tokens cannot distinguish between two challenge blinded datasets $BD_{X^{(0)}}^{\mathcal{T}}$ and $BD_{X^{(1)}}^{\mathcal{T}}$, for any two datasets $X^{(0)} = \left\{x_i^{(0)}\right\}_{i \in I}$ and $X^{(1)} = \left\{x_i^{(1)}\right\}_{i \in I}$, where $|x_i^{(0)}| = |x_i^{(1)}|$ for any $i \in I$.
2. **Data secrecy**: This requirement says that for any blinded dataset $BD_X^{\mathcal{T}}$ corresponding to $X = \{x_i\}_{i \in I}$ and any adversary $\mathcal{A}$ modeling an authorized data user, if x_{i_0} is unknown to $\mathcal{A}$, for an $i_0 \in I$, then the advantage of $\mathcal{A}$ in learning x_{i_0} from the blinded dataset as well as an arbitrary number of match secret-keys and PSI tokens is no more than that of one random element guess. Consequently, under this requirement, for an outsourced blinded dataset $BD_X^{\mathcal{T}}$ and a dataset Y belonging to an authorized data user, the data user only can learn $X \cap Y$ and nothing else about $X \setminus Y$.

In the following, we formalize the described security requirement in terms of the following two games:

4.4.1. Adaptively secure against chosen dataset attack

Let $\Pi = (\mathbf{Setup}, \mathbf{KeyGen}, \mathbf{Blind}, \mathbf{TokenGen}_1, \mathbf{TokenGen}_2, \mathbf{PSI})$ be an AB-PSI scheme, and $\mathcal{A}$ be a PPT adversary. Consider the following experiment:

Adaptively Chosen-Dataset Attack (AC-DA) indistifnguishability experiment $\mathtt{AC-DA}_{\mathcal{A},\Pi}(n)$:

1. **Setup**: The challenger selects a security parameter n and a universal attribute set $\mathbb{U}$ and runs $(PK, MSK) \leftarrow$ **Setup** $(1^n, \mathbb{U})$. It gives PK to the adversary $\mathcal{A}$.
2. **Phase 1**: $\mathcal{A}$ makes a polynomial number of queries to the following oracles, and the challenger keeps a list of attribute sets $L_{ATT}^{(\mathbf{u})}$, for any data user $\mathbf{u}$, which is initially empty:

- $\mathcal{O}_{\mathbf{KeyGen}}(Att, id_{\mathbf{u}})$: The challenger runs $SK_{Att}^{(\mathbf{u})} \leftarrow \mathbf{KeyGen}\ (PK, MSK, Att, id_{\mathbf{u}})$ and returns the requested secret-key to $\mathcal{A}$. It also adds Att to $L_{ATT}^{(\mathbf{u})}$.
- $\mathcal{O}_{\mathbf{TokenGen}_1}\left(Att, id_{\mathbf{u}}, \{y_j\}_{j\in J}\right)$: The challenger, at first, runs $SK_{Att}^{(\mathbf{u})} \leftarrow \mathbf{KeyGen}\ (PK, MSK, Att, id_{\mathbf{u}})$ and then using the obtained secret-key runs $\left(TK_{Att}^{(\mathbf{u})}, d\right) \leftarrow \mathbf{TokenGen}_1\left(PK, id_{\mathbf{u}}, SK_{Att}^{(\mathbf{u})}\right)$. It gives $TK_{Att}^{(\mathbf{u})}$ to the adversary $\mathcal{A}$.

3. **Challenge**: Adversary $\mathcal{A}$ selects an access tree $\mathcal{T}^*$, a finite index set I, and two datasets $X_0 = \left\{x_i^{(0)}\right\}_{i\in I}$ and $X_1 = \left\{x_i^{(1)}\right\}_{i\in I}$ such that:
   - For any data user $\mathbf{u}$ and $Att \in L_{ATT}^{(\mathbf{u})}, F_{\mathcal{T}^*}(Att) = 0$.
   - $|x_i^{(0)}| = |x_i^{(1)}|$, for any $i \in I$.
   - For any $t \in \{0,1\}$ and $i, k \in I, x_i^{(t)} \neq x_k^{(t)}$.

   Then, the challenger selects $b \leftarrow \{0,1\}$ and runs $BD_{X_b}^{\mathcal{T}^*} \leftarrow \mathbf{Blind}(PK, \mathcal{T}^*, X_b)$. $BD_{X_b}^{\mathcal{T}^*}$ is returned to $\mathcal{A}$.
4. **Phase 2**: $\mathcal{A}$ submits more queries to oracles $\mathcal{O}_{\mathbf{KeyGen}}(Att, id_{\mathbf{u}})$ and $\mathcal{O}_{\mathbf{TokenGen}_1}\left(Att, id_{\mathbf{u}}, \{y_j\}_{j\in J}\right)$ as in **Phase 1**. The only restriction is that $(Att, id_{\mathbf{u}})$ cannot be the input of $\mathcal{O}_{\mathbf{KeyGen}}$ if $F_{\mathcal{T}^*}(Att) = 1$.
5. **Guess**: $\mathcal{A}$ outputs a bit $b\prime \in \{0,1\}$.

The output of the experiment is defined to be 1 if $b\prime = b$, and 0 otherwise. We write $\mathtt{AC} - \mathtt{DA}_{\mathcal{A},\Pi}(n) = 1$ if the output is 1. In this case, we say that the adversary $\mathcal{A}$ succeeds.

Definition 3. An AB-PSI scheme $\Pi = (\mathbf{Setup}, \mathbf{KeyGen}, \mathbf{Blind}, \mathbf{TokenGen}_1, \mathbf{TokenGen}_2, \mathbf{PSI})$ is adaptively secure against chosen-dataset attack if for every PPT adversary $\mathcal{A}$, there is a negligible function *negl* such that:

$$\Pr(\mathtt{AC} - \mathtt{DA}_{\mathcal{A},\Pi}(n) = 1) \leqslant \frac{1}{2} + negl(n). \tag{7}$$

4.4.2. Data secrecy

For a PPT adversary $\mathcal{A}$ and an AB-PSI construction $\Pi = (\mathbf{Setup}, \mathbf{KeyGen}, \mathbf{Blind}, \mathbf{TokenGen}_1, \mathbf{TokenGen}_2, \mathbf{PSI})$, consider the following experiment:

Plaintext computing (PC) experiment $\mathtt{PC}_{\mathcal{A},\Pi}(n)$:

1. **Setup**: The challenger chooses a security parameter n and the universal attribute set $\mathbb{U}$. Then, it runs $(PK, MSK) \leftarrow \mathbf{Setup}(1^n, \mathbb{U})$ and returns PK to the adversary $\mathcal{A}$.
2. **Phase 1**: $\mathcal{A}$ makes a polynomial number of queries to the oracle $\mathcal{O}_{\mathbf{KeyGen}}(Att, id_{\mathbf{u}})$. When the challenger receives a request, it runs $SK_{Att}^{(\mathbf{u})} \leftarrow \mathbf{KeyGen}(PK, MSK, Att, id_{\mathbf{u}})$ and gives $SK_{Att}^{(\mathbf{u})}$ to $\mathcal{A}$.
3. **Challenge**: Adversary $\mathcal{A}$ returns a non-trivial access policy $\mathcal{T}^*$ to the challenger. The challenger selects a dataset $X^* = \{x_i^*\}_{i\in I^*}$, where $x_i^* \leftarrow \mathbb{Z}_q$, for any $i \in I^*$. Then, it runs $BD_{\{x_i^*\}_{i\in I^*}}^{\mathcal{T}^*} \leftarrow \mathbf{Blind}(PK, \mathcal{T}^*, X^*)$ and gives the generated blinded dataset to the adversary.
4. **Phase 2**: $\mathcal{A}$ makes more queries to the oracle $\mathcal{O}_{\mathbf{KeyGen}}(Att, id_{\mathbf{u}})$, without any restriction.
5. **Guess**: $\mathcal{A}$ outputs a plaintext x.

The output of the experiment is defined to be 1 if $x \in \{x_i^*\}_{i\in I^*}$, and 0 otherwise. We write $\mathtt{PC}_{\mathcal{A},\Pi} = 1$ when the output of the experiment is 1, and in this case we say the adversary succeeds.

Definition 4. An AB-PSI scheme $\Pi = (\mathbf{Setup}, \mathbf{KeyGen}, \mathbf{Blind}, \mathbf{TokenGen}_1, \mathbf{TokenGen}_2, \mathbf{PSI})$ achieves data secrecy if for every PPT adversary $\mathcal{A}$ there exists a negligible function *negl* such that:

$$\Pr(\mathtt{PC}_{\mathcal{A},\Pi}(n) = 1) \leqslant negl(n). \tag{8}$$

As will be shown in Theorems 2 and 3, under the **DBDH** and **DL** assumptions described in SubSection 3.1, our proposed scheme is secure against adaptively chosen-dataset attack, and it achieves data secrecy.

5. Our proposed scheme

In this section, we present the concrete construction of the basic AB-PSI scheme providing fine-grained access PSI computation. Then, we show that our proposed scheme is correct according to Definition 1. The notations used in our scheme are presented in Table 2.

5.1. Concrete construction of AB-PSI

Our proposed construction consists of the following four phases:

5.1.1. System Setup

To initialize public parameters and master secret-keys of the system, the CA considers a universal attribute set $\mathbb{U}$ and selects a security parameter n. Then, it executes **Setup** algorithm as follows, see Fig. 4.

Setup$(1^n, \mathbb{U})$: On input a security parameter 1^n and the universal attribute set $\mathbb{U}$, this algorithm runs $(n, q, G_1, G_2, \hat{e}) \leftarrow \mathcal{G}(1^n)$. Then, it selects $g_0, g_1, g_2, g_3 \leftarrow G_1$, $x_0, x_1 \leftarrow \mathbb{Z}_q$, and a collision-resistant hash function $H : G_2 \rightarrow \{0,1\}^m$, where m is a positive integer. Afterwards, for any attribute $a \in \mathbb{U}$, it selects $sk_a \leftarrow \mathbb{Z}_q$. This algorithm returns the public parameters

$$PK = \left(n, G_1, G_2, H, \hat{e}, g_0, g_1, g_2, h_0 = g_0^{x_0}, h_1 = g_0^{x_1}, h_2 = g_3 h_1^{x_0}, h_3 = \hat{e}(h_0, g_1), h_4 = \hat{e}\left(h_0^{-1}, h_1\right), \quad h_5 = \hat{e}(g_0, g_2), \left\{pk_a = g_0^{sk_a}\right\}_{a \in \mathbb{U}}\right) \tag{9}$$

and master secret-key

$$MSK = \left(x_0, x_1, g_3, \{sk_a\}_{a \in \mathbb{U}}\right).$$

5.1.2. ***User registration and key delegation***

When a data user **u** with an attribute set Att joins to the system, it first registers and selects a unique identifier $id_{\mathbf{u}}$. Then, it asks the CA to generate the secret-key corresponding to its attribute set. The CA runs **KeyGen** algorithm as follows and returns the queried secret-key to the data user, see Fig. 5.

KeyGen $(PK, MSK, Att, id_{\mathbf{u}})$: This algorithm generates secret-key of a data user with an identifier $id_{\mathbf{u}}$, and an attribute set Att, as follows:

$$SK_{Att}^{(\mathbf{u})} = \left\{sk_{a,\mathbf{u}} = g_1^{x_0} g_3 id_{\mathbf{u}}^{sk_a}\right\}_{a \in Att}. \tag{11}$$

5.1.3. Dataset blinding

When a data owner wants to outsource a dataset $X = \{x_i\}_{i \in I}$ to the CSP, to provide data confidentiality and fine-grained access control, it first defines an access control policy $\mathcal{T}$ and blinds X under $\mathcal{T}$ by using **Blind** algorithm as follows, see Fig. 6:

Blind$(PK, \mathcal{T}, X)$: Given a dataset $X = \{x_i\}_{i \in I}$ and an access tree $\mathcal{T}$, this algorithm chooses $r \leftarrow \mathbb{Z}_q$ and runs $\{q_{v_a}(0)\}_{v_a \in L_{\mathcal{T}}} \leftarrow$ **Share**$(\mathcal{T}, q, r)$. Then, it blinds X as follows:

$$BD_X^{\mathcal{T}} = \left(\mathcal{T}, C_1 = g_2^r, C_2 = h_4^r, \left\{C_{v_a} = g_0^{q_{v_a}(0)}\right\}_{v_a \in L_{\mathcal{T}}}, \quad \left\{C\prime_{v_a} = \left(pk_a g_2^{-1}\right)^{q_{v_a}(0)}\right\}_{v_a \in L_{\mathcal{T}}}, \quad C_{\{x_i\}_{i \in I}} = \left\{H\left(h_3^{rx_i}\right)\right\}_{i \in I}\right). \tag{12}$$

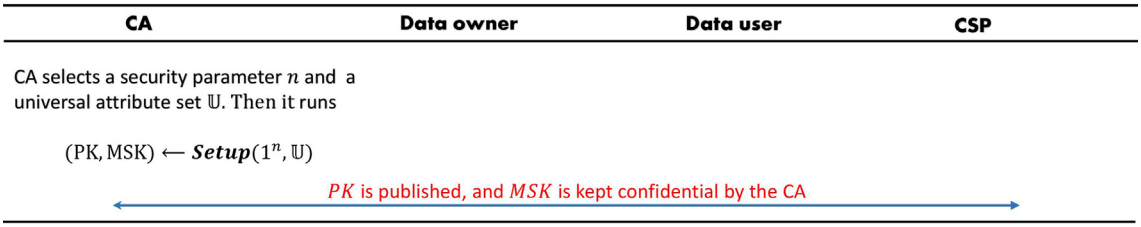

Fig. 4. System setup phase.

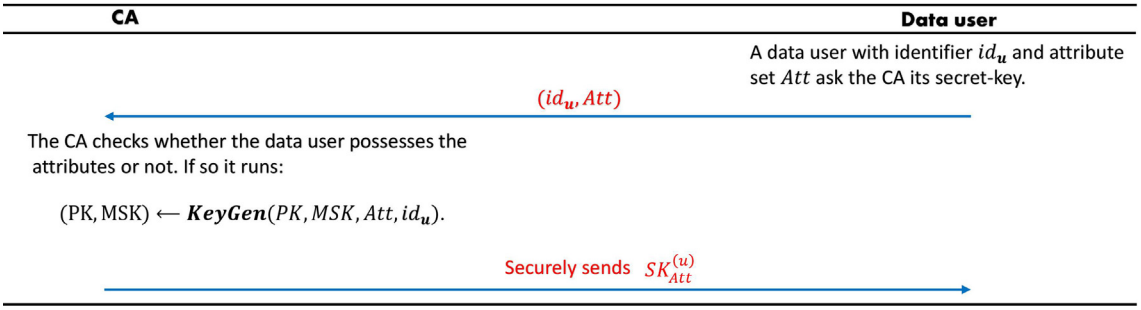

Fig. 5. User Registration and Key Delegation phase.

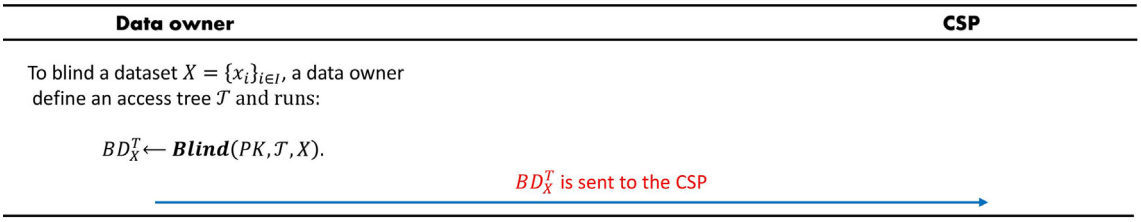

Fig. 6. Dataset blinding phase.

5.1.4. PSI computation

When a data user wants to compute the intersection between its private dataset and a data owner's dataset corresponding to $BD_X^{\mathcal{T}}$, at first, it runs $\textbf{TokenGen}_1$ algorithm to generate a PSI token for the CSP. When the CSP receives a valid PSI token from a data user, it runs $\textbf{TokenGen}_2$ algorithm and gives the data owner the required token for the PSI computation. The data user obtaining the PSI token can execute **PSI** algorithm and calculate the intersection of the two datasets, see Fig. 7. The mentioned three algorithms in this phase are defined as follows:

$\textbf{TokenGen}_1\left(PK, id_{\mathbf{u}}, SK_{Att}^{(\mathbf{u})}\right)$: On input an identifier and a data user's secret-key, this algorithm chooses $s, d \leftarrow \mathbb{Z}_q$ and returns a PSI token:

$$TK_{Att}^{(\mathbf{u})} = \left(tk_1 = g_0^d, tk_2 = h_1^d, tk_3 = id_{\mathbf{u}}^d, tk_4 = g_0^s, tk_5 = h_5^{sd}, \left\{tk_{a,\mathbf{u}} = sk_{a,\mathbf{u}} g_2^s\right\}_{sk_{a,\mathbf{u}} \in SK_{Att}^{(\mathbf{u})}}\right). \tag{13}$$

$\textbf{TokenGen}_2\left(PK, TK_{Att}^{(\mathbf{u})}, BD_X^{\mathcal{T}}\right)$: Given a PSI token $TK_{Att}^{(\mathbf{u})}$ and a blinded dataset $BD_X^{\mathcal{T}}$, this algorithm checks whether there is any attribute set $S \in 2^{Att}$ satisfying $\mathcal{T}$. If not, then this algorithm returns an error message $\perp$. Otherwise, for any $a \in S$, it computes:

$$\begin{aligned} TK_{a,\mathbf{u},d} &= \frac{\hat{e}\left(C_{v_a} tk_1, tk_{a,\mathbf{u}}\right)}{\hat{e}\left(C_{v_a} tk_1, h_2\right).\hat{e}\left(pk_a g_2^{-1}, tk_3\right).\hat{e}\left(C_{v_a}, id_{\mathbf{u}}\right).tk_5} \\ &= \hat{e}(h_0, g_1)^{q_{v_a}(0)+d}.h_4^{q_{v_a}(0)+d}.\hat{e}(g_2, id_{\mathbf{u}})^{q_{v_a}(0)+d}.\hat{e}\left(g_0^s, g_2\right)^{q_{v_a}(0)}. \end{aligned} \tag{14}$$

Then it runs $TK_{r,\mathbf{u},d} \leftarrow \textbf{Combine}\left(\mathcal{T}, q, \left\{T_{a,\mathbf{u},d}\right\}_{a\in S}\right)$, where

$$TK_{r,\mathbf{u},d} = \hat{e}(h_0, g_1)^{r+d}.h_4^{r+d}.\hat{e}(g_2, id_{\mathbf{u}})^{r+d}.\hat{e}\left(g_0^s, g_2\right)^r. \tag{15}$$

Afterwards, it obtains:

$$\begin{aligned} TK_{r,d} &= \frac{TK_{r,\mathbf{u},d}}{C_2.\hat{e}\left(h_0^{-1}, tk_2\right).\hat{e}\left(C_1, id_{\mathbf{u}} tk_4\right).\hat{e}\left(g_2, tk_3\right)} \\ &= h_3^{r+d}. \end{aligned} \tag{16}$$

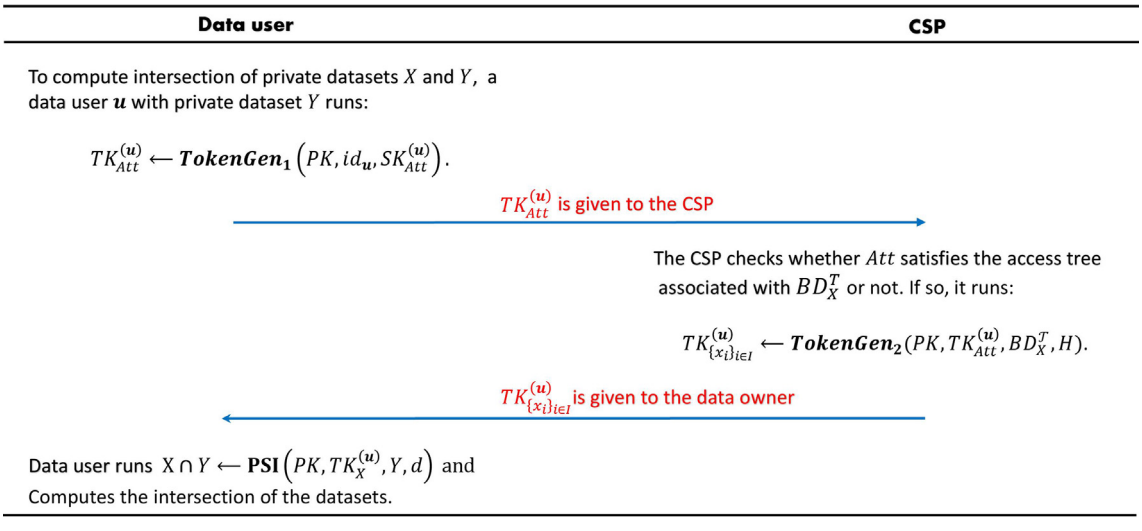

Fig. 7. PSI computation phase.

Finally, the algorithm outputs another PSI token:

$$TK^{(\mathbf{u})}_{\{x_i\}_{i\in I}} = \left(TK_{r,d}, C_{\{x_i\}_{i\in I}}\right). \tag{17}$$

PSI$\left(PK, TK^{(\mathbf{u})}_X, Y, d\right)$: To compute the intersection between a data owner's dataset $X = \{x_i\}_{i\in I}$ corresponding to a PSI token $TK^{(\mathbf{u})}_X$ and a data user's dataset $Y = \{y_j\}_{j\in J}$, this algorithm computes

$$\begin{aligned} H\left(\left(TK_{r,d}.h_3^{-d}\right)^{y_j}\right) &= H\left(\left(h_3^{r+d}.h_3^{-d}\right)^{y_j}\right) \\ &= H\left(h_3^{ry_j}\right), \end{aligned} \tag{18}$$

for any $j \in J$. Then, it returns $\{y_k\}_{k\in K} \subseteq Y$ as the final output, where

$$K = \left\{k \in J | H\left(h_3^{ry_k}\right) \in C_{\{x_i\}_{i\in I}} \cap \left\{H\left(h_3^{ry_j}\right)\right\}_{j\in J}\right\}. \tag{19}$$

5.2. Correctness analysis

In the following, we show the correctness of the proposed scheme.

Theorem 1. If H is a collision resistant hash function, then AB-PSI scheme is correct.

Proof.
The detailed proof is given in Appendix. □

6. Security analysis

In this section, we show that AB-PSI scheme satisfies the security requirements mentioned in Section 5.

Theorem 2. *If the* **DBDH** *problem is hard relative to* $\mathcal{G}$, then AB-PSI scheme is adaptively secure against chosen-dataset attack in the standard model.

Proof.
Let $\Pi = (\mathbf{Setup}, \mathbf{KeyGen}, \mathbf{Blind}, \mathbf{TokenGen}_1, \mathbf{TokenGen}_2, \mathbf{PSI})$ be our proposed AB-PSI scheme and $\mathcal{A}$ be a PPT adversary in $\mathtt{AC} - \mathtt{DA}_{\mathcal{A},\Pi}(n)$ experiment introduced in SubSection 4.4.1. Consider another adversary $\mathcal{A}\prime$ that aims to solve **DBDH** problem. Given $(n, q, G_1, G_2, \hat{e}, g, g^\alpha, g^\beta, g^\gamma, \hat{e}(g,g)^z)$, where $(n, q, G_1, G_2, \hat{e}) \leftarrow \mathcal{G}(1^n), g \leftarrow G_1, \alpha, \beta, \gamma \leftarrow \mathbb{Z}_q$, and $z = \alpha\beta\gamma$ or $z \leftarrow \mathbb{Z}_q$, the goal of $\mathcal{A}\prime$ is to determine the case of z. $\mathcal{A}\prime$ tries to solve the problem by running $\mathcal{A}$ as a subroutine as follows:

1. **Setup**: $\mathcal{A}\prime$ considers a universal attribute set $\mathbb{U}$ and a hash function $H : G_2 \to \{0,1\}^m$. It also selects $h_2 \leftarrow G_1, t_1, t_2 \leftarrow \mathbb{Z}_q$, and $sk_a \leftarrow \mathbb{Z}_q$ for any $a \in \mathbb{U}$. It gives

$$\begin{aligned} PK = \Big(&n, G_1, G_2, H, \hat{e}, g_0 = g, g_1 = g^\alpha, g_2 = g^{t_1}, h_0 = g^\beta, h_1 = g^\alpha g^{-t_2} = g^{\alpha - t_2}, h_2, h_3 = \hat{e}(h_0, g_1), h_4 = \hat{e}\left(h_0^{-1}, h_1\right), \\ &h_5 = \hat{e}(g_0, g_2), \{pk_a = g^{sk_a}\}_{a\in\mathbb{U}}\Big) \end{aligned} \tag{20}$$

   to the adversary $\mathcal{A}$. Comparing Eqs. (20) and (9), one concludes that if the master secret-key x_0 in (10) is equal to β, then the system public parameters in (20) are selected correctly.
   Note that, for an unknown value $g_3 \in G_1$, we have

$$h_2 = g_3 h_1^\beta = g_3 h_1^{x_0}, \tag{21}$$

   so by comparing (21) and (9) we see that h_2 is chosen correctly.
2. **Phase 1**: $\mathcal{A}$ is allowed to make adaptive queries to the following oracles, for polynomially-many times, and adversary $\mathcal{A}\prime$ holds a list $L^{(\mathbf{u})}_{ATT}$, for each data user $\mathbf{u}$, which is initially empty.
   $\mathcal{O}_{\mathbf{KeyGen}}\ (Att, id_{\mathbf{u}})$: For any attribute $a \in Att$, $\mathcal{A}\prime$ sets

$$sk_{a,\mathbf{u}} = h_2 \left(g^\beta\right)^{t_2} id_{\mathbf{u}}^{sk_a} \tag{22}$$

   and returns $SK^{(\mathbf{u})}_{Att} = \{sk_{a,\mathbf{u}}\}_{a\in Att}$ to the adversary $\mathcal{A}$. It also adds Att to $L^{(\mathbf{u})}_{ATT}$.
   $\mathcal{O}_{\mathbf{TokenGen}_1}(Att, id_{\mathbf{u}})$: At first, $\mathcal{A}\prime$ executes $\mathcal{O}_{\mathbf{KeyGen}}(Att, id_{\mathbf{u}})$ and generates $SK^{(\mathbf{u})}_{Att}$. Then, it selects $d, s \leftarrow \mathbb{Z}_q$ and gives a PSI token

$$TK_{Att}^{(\mathbf{u})} = \left(tk_1 = g_0^d, tk_2 = h_1^d, tk_3 = id_{\mathbf{u}}^d, tk_4 = g_0^s, tk_5 = h_5^{sd}, \left\{tk_{a,\mathbf{u}} = sk_{a,\mathbf{u}} g_2^s\right\}_{sk_{a,\mathbf{u}} \in SK_{Att}^{(\mathbf{u})}}\right) \tag{23}$$

to the adversary $\mathcal{A}$.
Note that, we have:

$$\begin{aligned} h_2(g^\beta)^{t_2} id_{\mathbf{u}}^{sk_a} &= h_2 g^{t_2\beta} id_{\mathbf{u}}^{sk_a} \overset{(21)}{=} g_3 h_1^\beta g^{t_2 beta} id_{\mathbf{u}}^{sk_a} \overset{(20)}{=} g_3 (g^{\alpha - t_2})^\beta g^{t_2\beta} id_{\mathbf{u}}^{sk_a} = g_3 g^{\alpha\beta} id_{\mathbf{u}}^{sk_a} \overset{(20)}{=} g_3 g_1^\beta id_{\mathbf{u}}^{sk_a} \\ &\overset{x_0 = \beta}{=} g_3 g_1^{x_0} id_{\mathbf{u}}^{sk_a} \overset{(11)}{=} sk_{a,\mathbf{u}}. \end{aligned} \tag{24}$$

So, the secret-key generated in (22) is valid.

3. **Challenge**: Adversary $\mathcal{A}$ returns an access tree $\mathcal{T}^*$ and two datasets $X_0 = \left\{x_i^{(0)}\right\}_{i\in I}, X_1 = \left\{x_i^{(1)}\right\}_{i\in I}$ to adversary $\mathcal{A}\prime$, where I is a finite index set, and $|x_i^{(0)}| = |x_i^{(1)}|, x_i^{(t)} \neq x_{i\prime}^{(t)}$, for any $i, i\prime \in I$ and $t \in \{0,1\}$. $\mathcal{A}\prime$ checks whether $F_{\mathcal{T}^*}(Att) = 0$ or not, for any $Att \in L_{ATT}^{(\mathbf{u})}$ and data user $\mathbf{u}$. If not, $\mathcal{A}\prime$ aborts. Otherwise, it selects $b \leftarrow \{0,1\}$ and defines:

$$\begin{cases} Q_R : \mathbb{Z}_q \to G_1 \\ Q_R(x) = g^\gamma g^{q_R(x)}, \end{cases} \tag{25}$$

where $q_R(x) : \mathbb{Z}_q \to \mathbb{Z}_q$ is a random $(k_R - 1)$-degree polynomial that $q_R(0) = 0$, and k_R is the threshold value of the root node of $\mathcal{T}^*$. Then, for any i-th child c_i of the root node R, it defines the following function:

$$\begin{cases} Q_{c_i} : \mathbb{Z}_q \to G_1 \\ Q_{c_i}(x) = g^{q_R(i)} g^{q_{c_i}(x)}, \end{cases} \tag{26}$$

where q_{c_i} is a random $(k_{c_i} - 1)$-degree polynomial that $q_{c_i}(0) = 0$. This process is continued until a function

$$\begin{cases} Q_v : \mathbb{Z}_q \to G_1 \\ Q_v(x) = g^{q_{p_v}(i)} g^{q_v(x)}, \end{cases} \tag{27}$$

is assigned to any node v of $\mathcal{T}^*$, where p_v is the parent of v; $q_v(0) = 0$, and q_v's degree is equal to $k_v - 1$.
Since the threshold value of each leaf node in the access tree $\mathcal{T}^*$ is equal to one, therefore the assigned functions to any leaf node of the access tree is a constant function. So, for an unknown value $q_{v_a} \in \mathbb{Z}_q$, we have $Q_{v_a} = g^{q_{v_a}}$, where v_a is the leaf node corresponding to the attribute a. It is not hard to see that $\left\{q_{v_a}\right\}_{v_a \in L_\mathcal{T}}$ is a valid output of **Share**$(\mathcal{T}, q, \gamma)$ algorithm. So, if we assume that the random element $r \in \mathbb{Z}_q$ chosen in **Blind** algorithm described in SubSection (5.1) is equal to γ, then for any $v_a \in L_\mathcal{T}$:

$$C_{v_a} = Q_{v_a}(0) = g_0^{q_{v_a}}, \tag{28}$$

and

$$C_{v_a}\prime = C_{v_a}^{sk_a} C_{v_a}^{-t_1} = \left(g_0^{q_{v_a}}\right)^{sk_a} \left(g_0^{q_{v_a}}\right)^{-t_1} = \left(g_0^{sk_a}\right)^{q_{v_a}} \left(g_0^{t_1}\right)^{-q_{v_a}} \overset{(20)}{=} pk_a^{q_{v_a}} g_2^{-q_{v_a}} = \left(pk_a g_2^{-1}\right)^{q_{v_a}} \tag{29}$$

are valid components of any blinded dataset with access tree $\mathcal{T}^*$.
Finally, $\mathcal{A}\prime$ returns

$$BD_{X_b}^{\mathcal{T}^*} = \left(\mathcal{T}^*, C_1, C_2, \{C_{v_a}\}_{v_a \in L_\mathcal{T}}, \{C\prime_{v_a}\}_{v_a \in L_\mathcal{T}}, C_{\left\{x_i^{(b)}\right\}_{i\in I}} = \{c_i\}_{i\in I}\right) \tag{30}$$

to the adversary $\mathcal{A}$, where

$$C_1 = (g^\gamma)^{t_1} = \left(g^{t_1}\right)^\gamma \overset{(20)}{=} g_2^\gamma, \tag{31}$$

$$C_2 = \hat{e}(g,g)^{-z}.\hat{e}\left(g^\beta, g^\gamma\right)^s, \tag{32}$$

$$c_i = H\left(\left(\hat{e}(g,g)^z\right)^{x_i^{(b)}}\right) = H\left(\hat{e}(g,g)^{z x_i^{(b)}}\right), \tag{33}$$

and C_{v_a} and $C_{v_a}\prime$ are the same as (28) and (29), respectively, for any $v_a \in L_\mathcal{T}$.

Remark 1.
Note that if $z = \alpha\beta\gamma$, then

$$\hat{e}(g,g)^{-z}.\hat{e}\left(g^\beta, g^\gamma\right)^{t_2} = \hat{e}(g,g)^{-\alpha\beta\gamma}.\hat{e}(g,g)^{t_2\beta\gamma} = \hat{e}(g,g)^{-\beta(\alpha - t_2)\gamma} = \hat{e}\left(g^{-\beta}, g^{\alpha - t_2}\right)^\gamma \overset{(20)}{=} \hat{e}\left(h_0^{-1}, h_1\right)^\gamma, \tag{34}$$

and

$$c_i = H\left(\hat{e}(g,g)^{z x_i^{(b)}}\right) = H\left(\hat{e}(g,g)^{\alpha\beta\gamma x_i^{(b)}}\right) = H\left(\hat{e}\left(g^\alpha, g^\beta\right)^{\gamma x_i^{(b)}}\right) = H\left(\hat{e}(g_1, h_0)^{\gamma x_i^{(b)}}\right) = H\left(h_3^{\gamma x_i^{(b)}}\right). \tag{35}$$

Assuming the uniform element r chosen in $\textbf{Blind}(PK, \mathcal{T}, \{x_i\}_{i\in I})$ algorithm is equal to the unknown value γ, from Eqs. (33)–(35), one concludes that (30) is a valid blinded dataset corresponding to X_b.

4. **Phase 2**: The adversary $\mathcal{A}$ makes more queries to the oracles $\mathcal{O}_{\textbf{KeyGen}}(Att, id_{\mathbf{u}})$ and $\mathcal{O}_{\textbf{TokenGen}_1}(Att, id_{\mathbf{u}})$ as in **Phase 1**, with the same restriction mentioned in SubSection 4.4.1.
5. **Guess**: Adversary $\mathcal{A}$ outputs a bit $b\prime \in \{0,1\}$. Then, $\mathcal{A}\prime$ checks whether $b = b\prime$ or not. If so, it outputs 1, and 0 otherwise.

As we mentioned in Remark 1, if $z = \alpha\beta\gamma$, then the returned blinded dataset to the adversary $\mathcal{A}$ is valid. So, in this case,

$$\Pr\left(\mathcal{A}\prime\left(n, q, G_1, G_2.\hat{e}, g, g^{\alpha}, g^{\beta}, g^{\gamma}, \hat{e}(g,g)^{\alpha\beta\gamma}\right) = 1\right) = \Pr(\mathtt{AC} - \mathtt{DA}_{\mathcal{A},\Pi}(n) = 1). \tag{36}$$

On the other hand, if z is a uniform element of $\mathbb{Z}_q$, then C_2 in (31) is also a uniform element of G_2. Therefore, it is independent of X_b. In this case, the adversary $\mathcal{A}$ cannot learn any information about X_b. Equivalently,

$$\Pr(b\prime = b) = \frac{1}{2}. \tag{37}$$

Thus:

$$\Pr\left(\mathcal{A}\prime(n, q, G_1, G_2.\hat{e}, g, g^{\alpha}, g^{\beta}, g^{\gamma}, \hat{e}(g,g)^{z}) = 1\right) = \frac{1}{2}. \tag{38}$$

Combining Eqs. (36) and (38), and the assumption that the **DBDH** problem is hard relative to $\mathcal{G}$ algorithm, one concludes that there is a negligible function *negl* that:

$$\begin{aligned} &|\Pr(\mathtt{AC} - \mathtt{DA}_{\mathcal{A},\Pi}(n) = 1) - \tfrac{1}{2}| = \\ &|\Pr\left(\mathcal{A}\prime\left(n, q, G_1, G_2.\hat{e}, g, g^{\alpha}, g^{\beta}, g^{\gamma}, \hat{e}(g,g)^{\alpha\beta\gamma}\right) = 1\right) - \Pr(\mathcal{A}\prime(n, q, G_1, G_2.\hat{e}, g, g^{\alpha}, g^{\beta}, g^{\gamma}, \hat{e}(g,g)^{z}) = 1)| \leqslant negl(n). \end{aligned} \tag{39}$$

Corollary 1. *Our proposed scheme is resistant against collusion attack of unauthorized data users.*

Proof. As we have seen in the proof of Theorem 2, any adversary $\mathcal{A}$ entitled to request all secret-keys of unauthorized data users is unable to break the security of our scheme. So, any group of unauthorized data users cannot learn any partial information about contents of data associated with a blinded dataset.

Theorem 3. *If the* **DL** *problem is hard relative to $\mathcal{G}$, then our proposed AB-PSI scheme gains data secrecy in the standard model.*

Proof.

Let $\Pi = (\textbf{Setup}, \textbf{KeyGen}, \textbf{Blind}, \textbf{TokenGen}_1, \textbf{TokenGen}_2, \textbf{PSI})$ be our AB-PSI scheme; $\mathcal{A}$ be the PPT adversary in $\mathtt{PC}_{\mathcal{A},\Pi}(n)$ experiment, and $\mathcal{A}\prime$ be another PPT adversary attempting to solve **DL** problem. Consider a challenger that runs $(n, q, G_1, G_2, \hat{e}) \leftarrow \mathcal{G}(1^n)$ and returns $(n, q, G_1, G_2, \hat{e}, g, g^{\alpha})$ to adversary $\mathcal{A}\prime$, where $g \leftarrow G_1, \alpha \leftarrow \mathbb{Z}_q$, and n is the security parameter of the system. The goal of $\mathcal{A}\prime$ is to compute α. $\mathcal{A}\prime$ runs $\mathcal{A}$ as a subroutine as follows:

1. **Setup**: The adversary $\mathcal{A}\prime$ considers a universal attribute set $\mathbb{U}$ and selects a hash function $H : G_2 \rightarrow \{0,1\}^m$, then it chooses $x_0, x_1 \leftarrow \mathbb{Z}_q, g_1, g_2, g_3 \leftarrow G_1$, and for any $a \in \mathbb{U}$ it selects $sk_a \leftarrow \mathbb{Z}_q$. It returns the public parameters $PK = \left(n, G_1, G_2, H, \hat{e}, g_0 = g, g_1, g_2, h_0 = g^{x_0}, h_1 = g^{x_1}, h_2 = g_3 h_1^{x_0}, h_3 = \hat{e}(h_0, g_1), h_4 = \hat{e}\left(h_0^{-1}, h_1\right), \{pk_a = g^{sk_a}\}_{a\in\mathbb{U}}\right)$ to $\mathcal{A}$. Also, it keeps the master secret-key $MSK = (x_0, x_1, g_3, \{sk_a\}_{a\in\mathbb{U}})$ to itself.
2. **Phase 1**: Adversary $\mathcal{A}$ makes adaptive queries to the oracle $\mathcal{O}_{\textbf{KeyGen}}(Att, id_{\mathbf{u}})$ for polynomially-many times. When $\mathcal{A}\prime$ receives a request $(Att, id_{\mathbf{u}})$, it runs $SK_{Att}^{(\mathbf{u})} \leftarrow \textbf{KeyGen}(PK, MSK, Att, id_{\mathbf{u}})$ and gives $SK_{Att}^{(\mathbf{u})}$ to $\mathcal{A}$.
3. **Challenge**: Adversary $\mathcal{A}$ returns an access tree $\mathcal{T}^*$ to adversary $\mathcal{A}\prime$. Then, for a positive integer k, $\mathcal{A}\prime$ selects a dataset $X^* = \{x_i^*\}_{i=1}^k$, where $x_i^* \leftarrow \mathbb{Z}_q$. It returns a blinded dataset

$$BD^{\mathcal{T}^*}_{\{\alpha x_i^*\}_{i=1}^k} = \left(C_1, C_2, \{C_{v_a}\}_{v_a\in L_{\mathcal{T}}}, \{C\prime_{v_a}\}_{v_a\in L_{\mathcal{T}}}, \{c_i\}_{i=1}^k\right), \tag{40}$$

where

$$c_i = H\left(\hat{e}(g^{\alpha}, g_1)^{rx_i^* x_0}\right) = H\left(\hat{e}(g^{x_0}, g_1)^{r\alpha x_i^*}\right) = H\left(\hat{e}(h_0, g_1)^{r\alpha x_i^*}\right) = H\left(h_3^{r(\alpha x_i^*)}\right), \tag{41}$$

for a uniform element $r \leftarrow \mathbb{Z}_q$, and the other components of $BD^{\mathcal{T}^*}_{\{\alpha x_i^*\}_{i=1}^k}$ are generated same as the **Blind** algorithm presented in Section 5.1. Comparing Eqs. (41) and (12), one concludes that $BD^{\mathcal{T}^*}_{\{\alpha x_i^*\}_{i=1}^k}$ is a valid blinded dataset corresponding to the access tree $\mathcal{T}^*$ and dataset $\alpha X^* = \{\alpha x_i^*\}_{i=1}^k$.

4. **Phase 2**: Adversary $\mathcal{A}$ can make more queries to oracle $\mathcal{O}_{\mathbf{KeyGen}}(Att, id_{\mathbf{u}})$, and $\mathcal{A}\prime$ answers them as in **Phase 1**.
5. **Guess**: Adversary $\mathcal{A}$ returns $\tilde{x} \in \mathbb{Z}_q$ to $\mathcal{A}\prime$.

Adversary $\mathcal{A}\prime$ selects $x^* \leftarrow \{x_1^*, \ldots, x_k^*\}$ and outputs $x^{*-1}\tilde{x}$. Now, if the adversary $\mathcal{A}$ wins the $\mathtt{PC}_{\mathcal{A},\Pi}$ experiment, then $\tilde{x} = \alpha x_{i_0}^*$, for an $i_0 \in \{1, \ldots, k\}$. So, the adversary $\mathcal{A}\prime$ solves the **DL** problem if and only if $x^* = x_{i_0}^*$. Therefore,

$$\begin{aligned}\Pr(\mathcal{A}\prime(n, q, g, g^{\alpha}, G_1, G_2, \hat{e}) = \alpha) &\geqslant \Pr\left(x^* = x_{i_0}^*\right)\Pr(\mathtt{PC}_{\mathcal{A},\Pi}(n) = 1) \\ &= \tfrac{1}{k}\Pr(\mathtt{PC}_{\mathcal{A},\Pi}(n) = 1).\end{aligned} \tag{42}$$

By the assumption that the **DL** problem is hard relative to $\mathcal{G}$, we have:

$$\Pr(\mathcal{A}\prime(n, q, g, g^{\alpha}, G_1, G_2, \hat{e}) = \alpha) \leqslant negl(n), \tag{43}$$

for a negligible function *negl*. Combining (42) and (43), one concludes that for a negligible function $negl\prime = k.negl$

$$\Pr(\mathtt{PC}_{\mathcal{A},\Pi}(n) = 1) \leqslant negl\prime(n). \tag{44}$$

Corollary 2. *For any outsourced blinded dataset to the CSP, no party even the CA can recover an element of the corresponding dataset.*

Proof. Consider an adversary $\mathcal{B}$ possessing a secret-key set of data users $\mathbf{SK}_{\mathcal{B}}$ in an AB-PSI system. Let $\Pi = (\mathbf{Setup}, \mathbf{KeyGen}, \mathbf{Blind}, \mathbf{TokenGen}_1, \mathbf{TokenGen}_2, \mathbf{PSI})$ be an AB-PSI scheme, $X^* = \{x_i^*\}_{i=1}^k$ be a dataset such that $\mathcal{B}$ does not have any information about its elements, and $BD_{X^*}^{\mathcal{T}}$ be a blinded dataset corresponding to X^*. Let us consider the following three cases:

1. $\mathcal{B}$ models the CSP that colludes with a group of unauthorized data users.
2. $\mathcal{B}$ models an authorized data user.
3. $\mathcal{B}$ models the CA.

Assume that $\mathtt{Deblind}_{\mathcal{B}}(BD_{X^*}^{\mathcal{T}})$ denotes the event that $\mathcal{B}$ recovers an element of X^*. In the following, we show that, in all the above three cases, $\Pr(\mathtt{Deblind}_{\mathcal{B}}(BD_{X^*}^{\mathcal{T}}))$ is a negligible function of the security parameter of the system.

According to Theorem 2, in the first case, $\mathcal{B}$ even is not able to distinguish between $BD_{X^*}^{\mathcal{T}}$ and another arbitrary blinded dataset. Therefore, the proof is straightforward. Also, if $\mathcal{B}$ models an authorized data user, by Theorem 3, one concludes $\Pr(\mathtt{Deblind}_{\mathcal{B}}(BD_X^{\mathcal{T}}))$ is negligible. Indeed, if $\mathcal{A}$ is the PPT adversary in $\mathtt{PC}_{\mathcal{A},\Pi}(n)$ experiment, it can query for any secret-key $SK_{Att}^{(\mathbf{u})} \in \mathbf{SK}_{\mathcal{B}}$. Therefore, we have:

$$\Pr\left(\mathtt{Deblind}_{\mathcal{B}}\left(BD_{X^*}^{\mathcal{T}}\right)\right) \leqslant \Pr(\mathtt{PC}_{\mathcal{A},\Pi}(n) = 1). \tag{45}$$

Combining (45) and (44), we conclude that $\Pr(\mathtt{Deblind}_{\mathcal{B}}(BD_{X^*}^{\mathcal{T}}))$ is a negligible function.

Now, assume that $\mathcal{B}$ models the CA. In this case, it must have the master secret-key of the system. So, it cannot be compared with adversaries in $\mathtt{AC-DA}_{\mathcal{A},\Pi}(n)$ or $\mathtt{PC}_{\mathcal{A},\Pi}(n)$ experiments. So, we have to consider another experiment denoted by $\widetilde{\mathtt{PC}}_{\mathcal{A},\Pi}(n)$ as follows:

1. **Setup**: The challenger selects a security parameter n and the universal attribute set $\mathbb{U}$. Then, it runs $(PK, MSK) \leftarrow \mathbf{Setup}\ (1^n, \mathbb{U})$ and returns (PK, MSK) to the adversary $\mathcal{A}$.
2. **Challenge**: Adversary $\mathcal{A}$ returns a non-trivial access policy $\mathcal{T}^*$ to the challenger. The challenger selects a dataset $X^* = \{x_i^*\}_{i \in I^*}$, where $x_i^* \leftarrow \mathbb{Z}_q$, for any $i \in I^*$. Then, it runs $BD_{\{x_i^*\}_{i\in I^*}}^{\mathcal{T}^*} \leftarrow \mathbf{Blind}(PK, \mathcal{T}^*, X^*)$ and returns $BD_{\{x_i^*\}_{i\in I^*}}^{\mathcal{T}^*}$ to $\mathcal{A}$.
3. **Guess**: $\mathcal{A}$ outputs a plaintext x.

The output of the experiment is defined to be 1 if $x \in \{x_i^*\}_{i\in I^*}$, and 0 otherwise. We write $\widetilde{\mathtt{PC}}_{\mathcal{A},\Pi}(n) = 1$ when the output of the experiment is 1, and we say the adversary succeeds.

The only difference between $\mathtt{PC}_{\mathcal{A},\Pi}(n)$ and $\widetilde{\mathtt{PC}}_{\mathcal{A},\Pi}(n)$ is that in $\widetilde{\mathtt{PC}}_{\mathcal{A},\Pi}(n)$ the master secret-key of the system is given to $\mathcal{A}$ instead of the oracle $\mathcal{O}_{\mathbf{KeyGen}}(Att, id_{\mathbf{u}})$. Note that, in $\widetilde{\mathtt{PC}}_{\mathcal{A},\Pi}(n)$, the adversary determines the access control policy of the blinded dataset, but in $\mathtt{Deblind}_{\mathcal{B}}(BD_{X^*}^{\mathcal{T}})$ the access policy is defined by a data owner not $\mathcal{B}$. Therefore, we have

$$\Pr\left(\mathtt{Deblind}_{\mathcal{B}}\left(BD_{X^*}^{\mathcal{T}}\right)\right) \leqslant \Pr\left(\widetilde{\mathtt{PC}}_{\mathcal{A},\Pi}(n) = 1\right). \tag{46}$$

In the following, we show that, under the hardness assumption of **DL** problem, $\Pr\left(\widetilde{\mathtt{PC}}_{\mathcal{A},\Pi}(n)=1\right)$ is a negligible function of n which proves this corollary. Let $\mathcal{A}\prime$ and $(n,q,G_1,G_2,\hat{e},g,g^{\alpha})$ be the same as in the proof of Theorem 3, and let $\mathcal{A}$ be an adversary in $\widetilde{\mathtt{PC}}_{\mathcal{A},\Pi}(n)$ experiment. Suppose that $\mathcal{A}\prime$ runs $\mathcal{A}$ as a subroutine as follows:

- $\mathcal{A}\prime$ provide $\mathcal{A}$ with (PK,MSK) by running **Setup**$(1^n,\mathbb{U})$ algorithm.
- $\mathcal{A}$ determines an access tree $\mathcal{T}^*$, and $\mathcal{A}\prime$ returns a blinded data set $BD^{\mathcal{T}^*}_{\{\alpha x_i^*\}_{i=1}^k}$ corresponding to $\alpha X^*=\{\alpha x_i^*\}_{i=1}^k$ which is generated in the same way as in the proof of Theorem 3.
- Adversary $\mathcal{A}$ returns an element $\tilde{x}\in\mathbb{Z}_q$ to $\mathcal{A}\prime$.

By a similar argument to that in the proof of Theorem 3, we can see that

$$\Pr(\mathcal{A}\prime(n,q,g,g^{\alpha},G_1,G_2,\hat{e})=\alpha)\geqslant\frac{1}{k}\Pr\left(\widetilde{\mathtt{PC}}_{\mathcal{A},\Pi}(n)=1\right). \tag{47}$$

Combining (47) and (46), we see that $\Pr\left(\mathtt{Deblind}_{\mathcal{B}}(BD_X^{\mathcal{T}})\right)$ is a negligible function.

Corollary 3. *In contrast with some existing IBE and ABE schemes, AB-PSI scheme does not suffer from key escrow problem. Indeed, assuming the CA is a semi-trusted entity that is curious to learn the elements of outsourced datasets, one can see that the scheme is still secure.*

Proof. As shown in Corollary 2, the CA cannot learn any information about the contents of data associated with blinded datasets. So, the proof is straightforward.

7. Characteristics and performance analysis

In this section, we first compare the characteristics of AB-PSI scheme with some current cloud-based PSI schemes. Then, we evaluate the efficiency of AB-PSI construction, including time execution, storage cost, and communication overhead, in terms of both asymptotic complexity and actual implementation of the proposed system. The notations used in this section are described in Table 3.

Notice that to make a fair comparison, we give up comparing the execution time and storage cost of AB-PSI scheme with the other works. The reason is that AB-PSI and current PSI protocols are fundamentally different in terms of underlying primitives and designed goals. However, our implementation result shows that AB-PSI is quite efficient and applicable.

We implemented AB-PSI scheme on an Ubuntu 18.04 laptop with an Intel Core i5-2410M Processor 2.3 GHz, 6 GB RAM using Python and python Pairing-Based Cryptography (pyPBC) library [39]. In our implementation, we used a 64-bit hash function and the bilinear map with Type A pairing ($l=512$) providing a level of security equivalent to 1024-bit discrete logarithm problem. Also, we assume that the access control policies are in the form of $(a_1$ **AND** $\ldots$ **AND** $a_N)$, where N is the number of attributes in the access policy, and a_i is an attribute, $1\leqslant i\leqslant N$. Indeed, according to the notations given in Table 3, we have $N=|L_{\mathcal{T}}|=|S|$.

7.1. Characteristics analysis

In this section, we evaluate AB-PSI scheme by comparing its features with the other cloud-based PSI schemes. The summary of the comparison results is given in Table 4.

Since the CSP is assumed to be honest but curious, protecting the privacy of outsourced data is more critical when some parts of the PSI computation are delegated to the CSP. In schemes presented in [25,35,42], the CSP can learn whether or not the intersection of two datasets is non-empty. While, in AB-PSI and [1,21–23], the CSP does not gain any information about the result of the protocol.

In a cloud-based PSI protocol, it is expected that the CSP cannot compute on outsourced datasets. However, in schemes [25,42], the CSP can compute the intersection of the outsourced datasets without the permission of data owners. This problem has been resolved in AB-PSI and [1,21–23,35].

Because the CSP is assumed to be semi-trusted, to obtain some unauthorized information about the outsourced datasets, it is reasonable that it colludes with some malicious data users. However, the schemes presented in [1,21,25] are vulnerable to the collusion attack. Indeed, in these schemes, any unauthorized data user colliding with the CSP can compute the intersection of its dataset with any outsourced dataset to the CSP. However, in AB-PSI and [22,23,35,42], colluding the CSP with data users does not threaten the security of the schemes.

Among the mentioned PSI schemes, the only ones providing multi PSI computation without needing to re-prepare the outsourced datasets are AB-PSI and [1]. In these schemes, clients generate tokens corresponding to their datasets, and the created tokens can be used whenever they want to run the PSI protocol. However, in the other schemes, clients have to re-prepare their tokens after each intersection computation.

Table 3
Notation used in our numerical comparison.

Notation	Description
$\lvert\mathbb{U}\rvert$	Cardinality of the universal attribute set of the system
$\lvert Att\rvert$	Cardinality of a data user's attribute set
$\lvert L_{\mathcal{T}}\rvert$	Cardinality of the leaf node set of an access tree $\mathcal{T}$
$\lvert S\rvert$	Cardinality of the minimal attribute set satisfying the access tree
$\lvert Y\rvert$	Cardinality of the data user's dataset
$\lvert X\rvert$	Cardinality of the data owner's dataset
T_{e_1}	Exponentiation operation time in G_1
T_{e_2}	Exponentiation operation time in G_2
T_P	Pairing operation time
T_{o_1}	Time of the group operation in G_1
T_{o_2}	Time of the group operation in G_2
T_H	Hash operation time
$T_{I(\lvert X\rvert,\lvert Y\rvert)}$	Time needed to compute the intersection between two datasets with cardinalities $\lvert X\rvert$ and $\lvert Y\rvert$
l_{G_1}	Bit length of an element in G_1
l_{G_2}	Bit length of an element in G_2
l_H	Bit length of an output of H
$l_{\mathbb{Z}_q}$	Bit length of an element in $\mathbb{Z}_q$

Table 4
Comparison of caractristics.

Features	AB-PSI	[22]	[23]	[25]	[42]	[21]	[35]	[1]
Private against the CSP	✓	✓	✓	×	×	✓	×	✓
PSI computation authorization	✓	✓	✓	×	×	✓	✓	✓
Secure against collusion of cloud and data users	✓	✓	✓	×	✓	×	✓	×
Supporting one-to-many communication without needing to re-prepare	✓	×	×	×	×	×	×	✓
None interactive with offline data owners	✓	×	×	×	×	×	×	×
Fine-grained access control	✓	×	×	×	×	×	×	×

In AB-PSI, the intersection computation can be done without the participation of data owners, so they can be offline after outsourcing their datasets to the CSP. However, in [1,21–23,25,35,42], both of the clients should be online and have interaction with each other.

Finally, AB-PSI is the only scheme providing fine-grained access control over the outsourced encrypted datasets. In AB-PSI, by defining an access control policy, a data owner with a dataset X can determine authorized data users to compute the intersection between their datasets and X. However, in schemes [1,21–23,25,35,42], clients cannot specify the authorized users before outsourcing their datasets, and they have to grant online permissions for intersection computation.

7.2. Execution-time overhead

In the following subsection, we analyze the execution-time overhead of AB-PSI scheme. We present our results in terms of the asymptotic complexity and execution time of the implemented algorithms.

Table 5 presents the execution-time overhead on the entities of the system. The asymptotic results are measured in terms of: Bilinear pairing operation, hash operation, exponentiation operation, and group operation.

7.2.1. Execution-time overhead on the CA

From Table 5, the execution time of **Setup** and **KeyGen** algorithms are functions of the cardinalities of the universal attribute set, $\lvert\mathbb{U}\rvert$, and the data user's attribute set, $\lvert Att\rvert$, respectively. By implementing these two algorithms, we computed their running time when $\lvert\mathbb{U}\rvert$ and $\lvert Att\rvert$ are ranged between 10 and 50. The obtained performance graphs are given in Fig. 8(a) and (b).

Table 5
Computation time overhead.

Algorithm	Running time	Executor
Setup	$(\lvert\mathbb{U}\rvert+3)T_{e_1}+3T_p+T_{o_1}$	CA
KeyGen	$(1+\lvert Att\rvert)(T_{e_1}+T_{o_1})$	CA
Blind	$(1+2\lvert L_{\mathcal{T}}\rvert)\lvert T_{e_1}+(\lvert X\rvert+1)T_{e_2}+\lvert X\rvert T_H+\lvert L_{\mathcal{T}}\rvert T_{o_1}$	Data owner
TokenGen$_1$	$5T_{e_1}+T_{e_2}+\lvert Att\rvert T_{o_1}$	Data user
TokenGen$_2$	$\lvert S\rvert T_{e_2}+(4\lvert S\rvert+4)T_p+(2\lvert S\rvert+1)T_{o_1}+(5\lvert S\rvert+3)T_{o_2}$	CSP
PSI	$\lvert Y\rvert(T_{e_2}+T_H)+T_{o_2}+T_{I(\lvert X\rvert,\lvert Y\rvert)}$	Data user

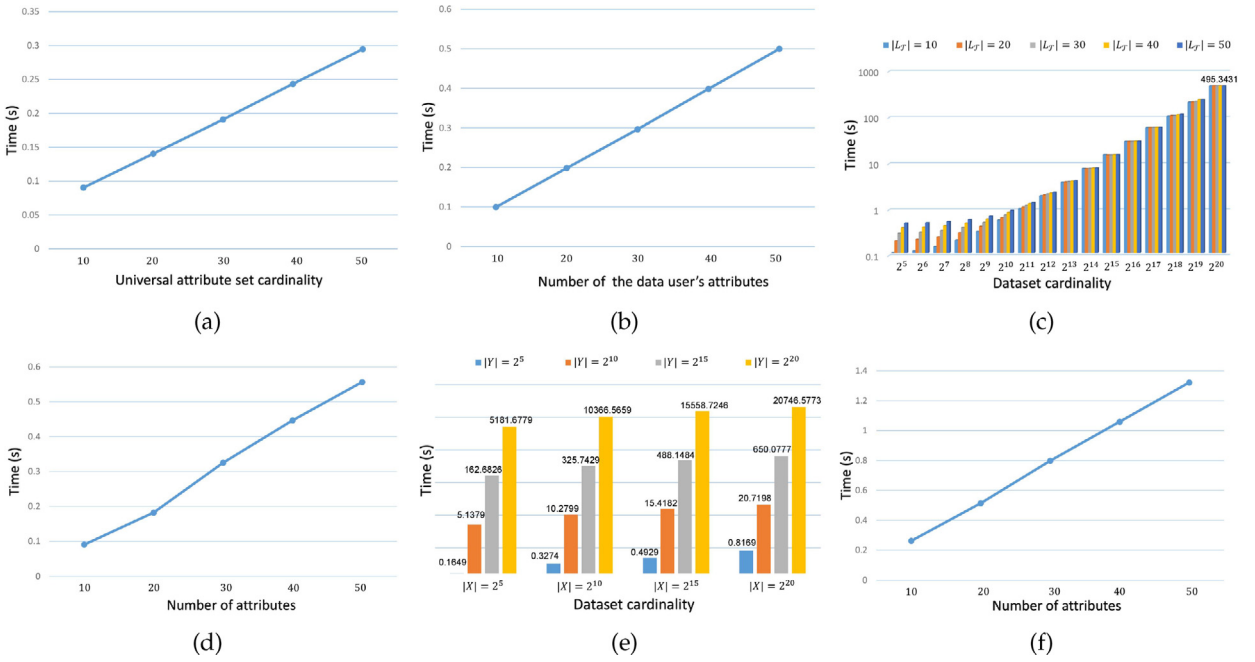

Fig. 8. Execution-time overhead on: (a) CA for running **Setup** algorithm; (b) CA for running **KeyGen** algorithm; (c) data owners in **Dataset Blinding** phase; (d) data users for running $\mathbf{TokenGen}_1$ algorithm; (e) data users for running **PSI** algorithm; (f) the CSP for running $\mathbf{TokenGen}_2$ algorithm.

7.2.2. Execution-time overhead on data owners

As we presented in Table 5, the execution time of **Blind** algorithm depends on the data owner's dataset cardinality, $|X|$, and the number of leaf nodes in the access tree, $|L_T|$. To observe variations of the running time, we implement **Blind** algorithm for $|X| \in \left\{2^5, 2^6, \ldots, 2^{20}\right\}$ and $|L_T| \in \{10, 20, \ldots, 50\}$. The obtained results are given in Fig. 8(c).

As we can see in this figure, when $2^5 \leqslant |X| \leqslant 2^{10}$, the execution time of the algorithm is less than one second. Also, the execution time is satisfactory when the dataset cardinality is very large. For example, when $|X| = 2^{18}, |X| = 2^{19}$, and $|X| = 2^{20}$, the execution times are not more than $121, 250$, and 495 seconds, respectively. Also, we observed that the execution time grows slowly by increasing the number of leaf nodes. In fact, it grows 0.1 s when the number of leaf nodes increased by 10.

7.2.3. Execution-time overhead on data users

From Table 5, the running time of $\mathbf{TokenGen}_1$ algorithm depends on the number of the data user's attributes. Also, the execution-time of **PSI** algorithm is affected by the data user's dataset cardinality, $|Y|$, and data owner's dataset cardinality, $|X|$. We executed these two algorithm when $|Att| \in \{10, 20, \ldots, 50\}$ and $|X|, |Y| \in \left\{2^5, 2^{10}, 2^{15}, 2^{20}\right\}$. The execution-time of $\mathbf{TokenGen}_1$ and PSI algorithms are illustrated in parts (d) and (e) of Fig. 8, respectively.

7.2.4. Execution-time overhead on the CSP

According to Table 5, the execution-time of $\mathbf{TokenGen}_2$ algorithm is a function of the data user's minimal attribute set satisfying the access tree, $|S|$. Fig. 8(f) presents the execution-time of this algorithm when $|S|$ is ranged between 10 and 50.

7.3. Storage overhead

In this subsection, we analyze the storage overhead of our AB-PSI system. We evaluate storage cost on the CSP for storing outsourced blinded datasets and storage cost on data users for maintaining their secret-keys and their datasets. We do not consider the storage cost on data owners for maintaining their datasets, because they can become offline after outsourcing their dataset forever, so they do not need their dataset for running the algorithm of the system.

In the following, we present our asymptotic complexity and implementation results.

Table 6
Storage overhead.

Stored data	Size	Stored by
blinded dataset	$l_{G_2} + (2\|L_T\| + 1)l_{G_1} + \|X\|l_H$	CSP
Secret-key	$\|Att\|l_{G_1}$	Data user
Dataset	$\|Y\|l_{\mathbb{Z}_q}$	Data user

7.3.1. Storage overhead on the CSP

The storage complexity on the CSP is demonstrated in Table 6. As the table shows, it grows linearly with the cardinality of the data owner's dataset, $|X|$, and the number of leaf nodes in the access tree, $|L_T|$. By implementing **Blind** algorithm, we evaluate the storage cost when $|X| \in \left\{2^5, 2^6, \ldots, 2^{20}\right\}$ and $|L_T| \in \{10, 20, \ldots, 50\}$. Fig. 9(a) presents the resulted performance graph.

One can see that the storage cost is low even for high dataset cardinality. Indeed, from Fig. 9(a), the volume of a blinded dataset is less than 1 MB when $|X| \leq 2^{16}$, and the volume is not more than 10 MB when $|X| = 2^{20}$.

7.3.2. Storage overhead on data users

Table 6 demonstrates the storage overhead on data users. The storage costs are due to storing their secret-keys and their datasets. As we see in the table, the length of the secret-keys depends on the number of data users' attributes. Also, in our scheme, any data user should map each element in their dataset to an element in $\mathbb{Z}_q$ and save it for running the **PSI** and **TokenGen**$_1$ algorithms. So, $|Y|l_{\mathbb{Z}_q}$-bit memories are needed for the storage.

Fig. 9 reveals the implementation results. In part (b) of the figure, we evaluated the size of a data user's secret-key when the number of its attributes is ranged between 10 and 50. Also, part (c) presents the storage cost on a data user for maintaining its dataset when the dataset cardinality is ranged between 2^5 and 2^{20}.

7.4. Communication overhead

In this subsection, we discuss the theoretical and experimental results of the communication overhead in AB-PSI system that is mainly incurred by transmitting the secret-keys, blinded datasets, and PSI tokens. Specifically, we calculate the communication overhead in terms of the size of secret-keys transmitted by the CA to a data user, size of blinded dataset out-

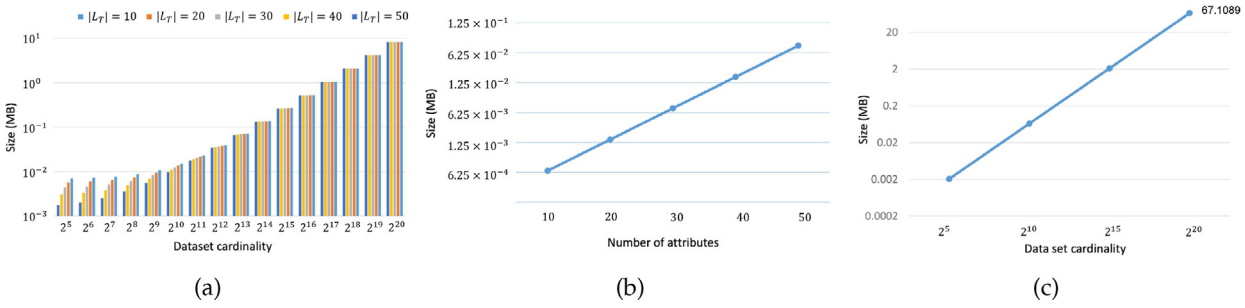

Fig. 9. Storage overhead on: (a) the CSP for maintaining a blinded dataset ; (b) data users for storing their datasets; (c) data users for keeping their secret-keys.

Table 7
Communication overhead in ordinary executions.

Communication overhead from	Size	Algorithm
CA to data user	$\|Att\|l_{G_1}$	**KeyGen**
Data owner to CSP	$l_{G_2} + (2\|L_T\| + 1)l_{G_1} + \|X\|l_H$	**Blind**
Data user to CSP	$(\|Att\| + 4)l_{G_1} + l_{G_2}$	**TokenGen**$_1$
CSP to data user	$l_{G_2} + \|X\|l_H$	**TokenGen**$_2$

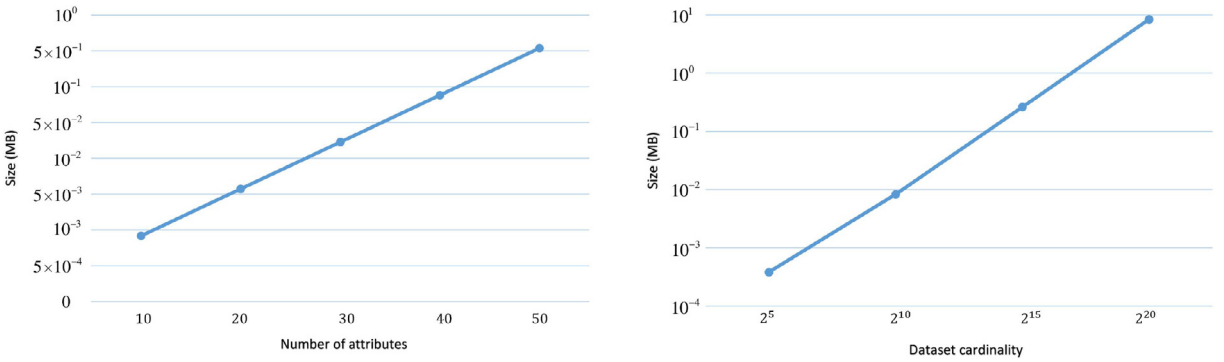

Fig. 10. Communication cost from: (a) Data users to the CSP; (b) The CSP to a data user.

sourced to the CSP by a data owner, size of a PSI token generated by $\mathbf{TokenGen}_1$ algorithm and sent to the CSP by a data user, and size of the output of $\mathbf{TokenGen}_2$ algorithm transmitted from the CSP to a data user.

For secret-keys and blinded datasets, the results on the storage and communication overhead are the same. Therefore, since in the last subsection, we analyzed the storage overhead of secret-keys and blinded dataset, here we only consider the communication overhead caused by transmitting the PSI tokens.

From Table 7, the communication overhead caused by transmitting the output of $\mathbf{TokenGen}_1$ algorithm is a function of the data user's attribute set cardinality, $|Att|$. We evaluated the communication overhead when $|Att|$ is ranged in $10-50$. The implementation results are given in Fig. 10(a).

Also, from the table, the communication cost caused by transmitting the output of $\mathbf{TokenGen}_2$ from the CSP to a data user depends on the cardinality of the underlying dataset, $|X|$. In Fig. 10(b), we computed the size of the transmitted data when $|X|$ is ranged between 2^5 and 2^{20}.

8. Conclusion

In this paper, we designed the first fine-grained access control system for the private set intersection (PSI) computation in cloud computing. The scheme achieves the following advantages: i) Data owners can control intersection computation on their datasets by defining an access control policy. ii) The scheme is not interactive, and data owners can be offline during the PSI protocol. iii) It supports one-to-many communication. Indeed, In our scheme, blinded outsourced datasets to the cloud can be used in PSI computations for arbitrary times. Also, we provided security definitions for the AB-PSI scheme, and we formally proved its security in the standard model. We also demonstrated the utility of AB-PSI scheme by evaluating its performance. Security analyses and performance evaluations indicated that the new tool is efficient and practical.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Appendix A. Appendix. Correctness proof

Theorem 1: If H is a collision resistant hash function, then AB-PSI scheme is correct.

Proof. Consider two datasets $X=\{x_i\}_{i\in I}$ and $Y=\left\{y_j\right\}_{j\in J}$, $(PK,MSK)\leftarrow\mathbf{Setup}(1^n,\mathbb{U})$, an access tree $\mathcal{T}$ satisfied by an attribute set Att, $SK_{Att}^{(\mathbf{u})}\leftarrow\mathbf{KeyGen}(PK,MSK,Att,id_{\mathbf{u}})$, $BD_X^{\mathcal{T}}\leftarrow\mathbf{Blind}(PK,\mathcal{T},X)$, $TK_{Att}^{(\mathbf{u})}\leftarrow\mathbf{TokenGen}_1\left(PK,id_{\mathbf{u}},SK_{Att}^{(\mathbf{u})}\right)$, and $TK_X^{(\mathbf{u})}\leftarrow\mathbf{TokenGen}_2\left(PK,TK_{Att}^{(\mathbf{u})},BD_X^{\mathcal{T}}\right)$. In the following, we first prove Eqs. (14) and (16). Then, we conclude

$$\mathbf{PSI}\left(PK,TK_X^{(\mathbf{u})},Y,d\right)=X\cap Y \tag{48}$$

provided the hash function H is a collision resistant hash function. We have:

$$
\begin{aligned}
TK_{a,\mathbf{u},d} = & \frac{\hat{e}\left(C_{v_a}tk_1, tk_{a,\mathbf{u}}\right)}{\hat{e}\left(C_{v_a}tk_1, h_2\right).\hat{e}\left(pk_a g_2^{-1}, tk_3\right).\hat{e}\left(C'_{v_a}, id_{\mathbf{u}}\right).tk_5} \\
\overset{(12),(13)}{=} & \frac{\hat{e}\left(g_0^{q_{v_a}(0)} g_0^{d}.sk_{a,\mathbf{u}} g_2^{s}\right)}{\hat{e}\left(C_{v_a}tk_1, h_2\right).\hat{e}\left(pk_a g_2^{-1}, id_{\mathbf{u}}^{d}\right).\hat{e}\left(\left(pk_a g_2^{-1}\right)^{q_{v_a}}, id_{\mathbf{u}}\right).tk_5} \\
\overset{(11)}{=} & \frac{\hat{e}\left(g_0^{q_{v_a}(0)+d} g_1^{x_0} g_3 id_{\mathbf{u}}^{sk_a} g_2^{s}\right)}{\hat{e}\left(C_{v_a}tk_1, h_2\right).e\left(pk_a g_2^{-1}, id_{\mathbf{u}}\right)^{d}.e\left(pk_a g_2^{-1}, id_{\mathbf{u}}\right)^{q_{v_a}(0)}.tk_5} \\
= & \frac{\hat{e}\left(g_0^{q_{v_a}(0)+d} g_1^{x_0}\right).\hat{e}\left(g_0^{q_{v_a}(0)+d}.g_3\right).\hat{e}\left(g_0^{q_{v_a}(0)+d}, id_{\mathbf{u}}^{sk_a}\right).e\left(g_0, g_2^{s}\right)^{q_{v_a}(0)+d}}{\hat{e}\left(C_{v_a}tk_1, h_2\right).e\left(pk_a g_2^{-1}, id_{\mathbf{u}}\right)^{q_{v_a}(0)+d}.tk_5} \\
= & \frac{e\left(g_0^{x_0}.g_1\right)^{q_{v_a}(0)+d}.e(g_0.g_3)^{q_{v_a}(0)+d}.e\left(g_0^{sk_a}, id_{\mathbf{u}}\right)^{q_{v_a}(0)+d}.e\left(g_0, g_2^{s}\right)^{q_{v_a}(0)}.e\left(g_0, g_2^{s}\right)^{d}}{\hat{e}\left(C_{v_a}tk_1, h_2\right).e\left(pk_a g_2^{-1}, id_{\mathbf{u}}\right)^{q_{v_a}(0)+d}.h_5^{sd}} \\
\overset{(9)}{=} & \frac{e(h_0, g_1)^{q_{v_a}(0)+d}.e(g_0, g_3)^{q_{v_a}(0)+d}.e\left(pk_a g_2^{-1} g_2, id_{\mathbf{u}}\right)^{q_{v_a}(0)+d}.e\left(g_0^{s}, g_2\right)^{q_{v_a}(0)}.h_5^{sd}}{\hat{e}\left(C_{v_a}tk_1, h_2\right).e\left(pk_a g_2^{-1}, id_{\mathbf{u}}\right)^{q_{v_a}(0)+d}.h_5^{sd}} \\
= & \frac{e(h_0, g_1)^{q_{v_a}(0)+d}.e(g_0, g_3)^{q_{v_a}(0)+d}.e\left(pk_a g_2^{-1}, id_{\mathbf{u}}\right)^{q_{v_a}(0)+d}.e(g_2, id_{\mathbf{u}})^{q_{v_a}(0)+d}.e\left(g_0^{s}, g_2\right)^{q_{v_a}(0)}}{\hat{e}\left(C_{v_a}tk_1, h_2\right)e\left(pk_a g_2^{-1}, id_{\mathbf{u}}\right)^{q_{v_a}(0)+d}} \\
= & \frac{\hat{e}(h_0, g_1)^{q_{v_a}(0)+d}.\hat{e}\left(g_0 g_3 h_1^{x_0} h_1^{-x_0}\right)^{q_{v_a}(0)+d}.\hat{e}(g_2, id_{\mathbf{u}})^{q_{v_a}(0)+d}.\hat{e}\left(g_0^{s}, g_2\right)^{q_{v_a}(0)}}{\hat{e}\left(C_{v_a}tk_1, h_2\right)} \\
= & \frac{\hat{e}(h_0, g_1)^{q_{v_a}(0)+d}.\hat{e}\left(g_0 g_3 h_1^{x_0}\right)^{q_{v_a}(0)+d}.\hat{e}\left(g_0, h_1^{-x_0}\right)^{q_{v_a}(0)+d}.\hat{e}(g_2, id_{\mathbf{u}})^{q_{v_a}(0)+d}.\hat{e}\left(g_0^{s}, g_2\right)^{q_{v_a}(0)}}{\hat{e}\left(C_{v_a}tk_1, h_2\right)} \\
\overset{(9)}{=} & \frac{\hat{e}(h_0, g_1)^{q_{v_a}(0)+d}.\hat{e}\left(g_0{}^{q_{v_a}(0)+d}, h_2\right).\hat{e}(g_0{}^{-x_0}, h_1)^{q_{v_a}(0)+d}.\hat{e}(g_2, id_{\mathbf{u}})^{q_{v_a}(0)+d}.\hat{e}\left(g_0^{s}, g_2\right)^{q_{v_a}(0)}}{\hat{e}\left(C_{v_a}tk_1, h_2\right)} \\
\overset{(9),(12),(13)}{=} & \frac{\hat{e}(h_0, g_1)^{q_{v_a}(0)+d}.\hat{e}\left(C_{v_a}tk_1, h_2\right).\hat{e}\left(h_0{}^{-1}, h_1\right)^{q_{v_a}(0)+d}.\hat{e}(g_2, id_{\mathbf{u}})^{q_{v_a}(0)+d}.\hat{e}\left(g_0^{s}, g_2\right)^{q_{v_a}(0)}}{\hat{e}\left(C_{v_a}tk_1, h_2\right)} \\
\overset{(9)}{=} & \hat{e}(h_0, g_1)^{q_{v_a}(0)+d}.h_4^{q_{v_a}(0)+d}.\hat{e}(g_2, id_{\mathbf{u}})^{q_{v_a}(0)+d}.\hat{e}\left(g_0^{s}, g_2\right)^{q_{v_a}(0)}.
\end{aligned}
\tag{49}
$$

So Eq. (14) is correct. Moreover,

$$
\begin{aligned}
TK_{r,d} = & \frac{TK_{r,\mathbf{u},d}}{C_2.\hat{e}\left(h_0^{-1}, tk_2\right).\hat{e}(C_1, id_{\mathbf{u}} tk_4).\hat{e}(g_2, tk_3)} \\
\overset{(15)}{=} & \frac{e(h_0, g_1)^{r+d}.h_4^{r+d}.e(g_2, id_{\mathbf{u}})^{r+d}.e\left(g_0^{s}, g_2\right)^{r}}{C_2.\hat{e}\left(h_0^{-1}, tk_2\right).\hat{e}(C_1, id_{\mathbf{u}} tk_4).\hat{e}(g_2, tk_3)} \\
= & \frac{e(h_0, g_1)^{r+d}.h_4^{r}.h_4^{d}.e(g_2, id_{\mathbf{u}})^{r}.e(g_2, id_{\mathbf{u}})^{d}.\hat{e}\left(g_0^{s}, g_2^{r}\right)}{C_2.\hat{e}\left(h_0^{-1}, tk_2\right).\hat{e}(C_1, id_{\mathbf{u}} tk_4).\hat{e}(g_2, tk_3)} \\
\overset{(9),(12)}{=} & \frac{e(h_0, g_1)^{r+d}.C_2.e\left(h_0^{-1}, h_1\right)^{d}.\hat{e}\left(g_2^{r}, id_{\mathbf{u}}\right).\hat{e}\left(g_2, id_{\mathbf{u}}^{d}\right).\hat{e}\left(g_0^{s}, g_2^{r}\right)}{C_2.\hat{e}\left(h_0^{-1}, tk_2\right).\hat{e}(C_1, id_{\mathbf{u}} tk_4).\hat{e}(g_2, tk_3)} \\
\overset{(12),(13)}{=} & \frac{e(h_0, g_1)^{r+d}.C_2.\hat{e}\left(h_0^{-1}, h_1^{d}\right).\hat{e}(C_1, id_{\mathbf{u}}).\hat{e}(g_2, tk_3).\hat{e}(tk_4, C_1)}{C_2.\hat{e}\left(h_0^{-1}, tk_2\right).\hat{e}(C_1, id_{\mathbf{u}} tk_4).\hat{e}(g_2, tk_3)} \\
\overset{(9),(13)}{=} & \frac{h_3^{r+d}.C_2.\hat{e}\left(h_0^{-1}, tk_2\right).\hat{e}(C_1, id_{\mathbf{u}} tk_4).\hat{e}(g_2, tk_3)}{C_2.\hat{e}\left(h_0^{-1}, tk_2\right).\hat{e}(C_1, id_{\mathbf{u}} tk_4).\hat{e}(g_2, tk_3)} \\
= & h_3^{r+d}.
\end{aligned}
\tag{50}
$$

Thus, Eq. (16) is correct. Also, the correctness of Eq. (18) can easily be checked. On the other hands, for any two elements $x, y \in \mathbb{Z}_q$, we have $x = y$ if and only if $h_3^x = h_3^y$. Therefore, by the assumption that H is a collision-resistant hash function, we have

$$\{y_k\}_{k\in K} = \{x_i\}_{i\in I} \cap \{y_j\}_{j\in J},$$

where $K = \left\{k \in J | H\left(h_3^{ry_k}\right) \in C_{\{x\}_{i\in I}} \cap \left\{H\left(h_3^{ry_j}\right)\right\}_{j\in J}\right\}$. □

References

[1] A. Abadi, S. Terzis, R. Metere, C. Dong, Efficient delegated private set intersection on outsourced private datasets, IEEE Trans. Depend. Secure Comput. (2017).

[2] R. Agrawal, R. Srikant, Privacy-preserving data mining, in: ACM Sigmod Record, vol. 29, No. 2, 2000, ACM, pp. 439–450 .

[3] M.M. Al Aziz, D. Alhadidi, N. Mohammed, Secure approximation of edit distance on genomic data, BMC Med. Genom. 10 (2) (2017) 41.

[4] M. Ali, J. Mohajeri, M.-R. Sadeghi, A fully distributed revocable ciphertext-policy hierarchical attribute-based encryption without pairing, IACR Cryptol. ePrint Arch. 2018 (2018) 1102.

[5] G. Asharov, S. Halevi, Y. Lindell, T. Rabin, Privacy-preserving search of similar patients in genomic data, Proc. Privacy Enhanc. Technol. 2018 (4) (2018) 104–124.
[6] G. Ateniese, E. De Cristofaro, G. Tsudik, (If) size matters: size-hiding private set intersection, in: International Workshop on Public Key Cryptography, Springer, Berlin, Heidelberg, 2011, pp. 156–173.
[7] P. Baldi, R. Baronio, E. De Cristofaro, P. Gasti, G. Tsudik, Countering gattaca: efficient and secure testing of fully-sequenced human genomes, in: Proceedings of the 18th ACM Conference on Computer and Communications Security, 2011, ACM, pp. 691–702 .
[8] S. Belguith, N. Kaaniche, M. Laurent, A. Jemai, R. Attia, Constant-size threshold attribute based signcryption for cloud applications, in: SECRYPT 2017: 14th International Conference on Security and Cryptography, vol. 6, 2017, Scitepress, pp. 212–225.
[9] J. Bethencourt, A. Sahai, B. Waters, Ciphertext-policy attribute-based encryption, in: 2007 IEEE Symposium on Security and Privacy (SP'07), 2017, IEEE, pp. 321–334.
[10] C. Blundo, E. De Cristofaro, P. Gasti, EsPRESSo: efficient privacy-preserving evaluation of sample set similarity, in: Data Privacy Management and Autonomous Spontaneous Security, Springer, Berlin, Heidelberg, 2012, pp. 89–103.
[11] H. Cui, G. Wang, R.H. Deng, B. Qin, Escrow free attribute-based signature with self-revealability, Inf. Sci. 367 (2016) 660–672.
[12] E. De Cristofaro, J. Kim, G. Tsudik, Linear-complexity private set intersection protocols secure in malicious model, in: International Conference on the Theory and Application of Cryptology and Information Security, Springer, Berlin, Heidelberg, 2010, pp. 213–231.
[13] E. De Cristofaro, G. Tsudik, Practical private set intersection protocols with linear complexity, in: International Conference on Financial Cryptography and Data Security, Springer, Berlin, Heidelberg, 2010, pp. 143–159.
[14] S.K. Debnath, R. Dutta, Secure and efficient private set intersection cardinality using bloom filter, in: International Conference on Information Security, Springer, Cham, 2015, pp. 209–226.
[15] C. Dong, L. Chen, Z. Wen, November. When private set intersection meets big data: an efficient and scalable protocol, in: Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security, 2013, ACM, pp. 789–800.
[16] M.J. Freedman, K. Nissim, B. Pinkas, Efficient private matching and set intersection, in: International Conference on the Theory and Applications of Cryptographic Techniques, Springer, Berlin, Heidelberg, 2004, pp. 1–19.
[17] P. Gasti, K.B. Rasmussen, Privacy-preserving user matching, in: Proceedings of the 14th ACM Workshop on Privacy in the Electronic Society, 2015, ACM, pp. 111–120 .
[18] V. Goyal, O. Pandey, A. Sahai, B. Waters, Attribute-based encryption for fine-grained access control of encrypted data, in: Proceedings of the 13th ACM Conference on Computer and Communications Security, 2006, ACM, pp. 89–98.
[19] H. Hong, X. Liu, Z. Sun, A fine-grained attribute based data retrieval with proxy re-encryption scheme for data outsourcing systems, Mobile Networks Appl. (2018) 1–6.
[20] J. Hua, G. Shi, H. Zhu, F. Wang, X. Liu, H. Li, CAMPS: efficient and privacy-preserving medical primary diagnosis over outsourced cloud, Inf. Sci. 527 (2020) 560–575.
[21] S. Kamara, P. Mohassel, M. Raykova, S. Sadeghian, Scaling private set intersection to billion-element sets, in: International Conference on Financial Cryptography and Data Security, Springer, Berlin, Heidelberg, 2014, pp. 195–215.
[22] F. Kerschbaum, Outsourced private set intersection using homomorphic encryption, in: Proceedings of the 7th ACM Symposium on Information, Computer and Communications Security, 2012, ACM, pp. 85–86.
[23] F. Kerschbaum, Collusion-resistant outsourcing of private set intersection, in: Proceedings of the 27th Annual ACM Symposium on Applied Computing, 2012, ACM, pp. 1451–1456.
[24] L. Kissner, D. Song, Privacy-preserving set operations, in: Annual International Cryptology Conference, Springer, Berlin, Heidelberg, 2005, pp. 241–257.
[25] F. Liu, W.K. Ng, W. Zhang, S. Han, Encrypted set intersection protocol for outsourced datasets, in: 2014 IEEE International Conference on Cloud Engineering, 2014, IEEE, pp. 135–140 .
[26] H. Liu, H. Ning, Q. Xiong, L.T. Yang, Shared authority based privacy-preserving authentication protocol in cloud computing, IEEE Trans. Parallel Distrib. Syst. 26 (1) (2015) 241–251.
[27] Y. Miao, X. Liu, K.K.R. Choo, R.H. Deng, J. Li, H. Li, J. Ma, Privacy-preserving attribute-based keyword search in shared multi-owner setting, IEEE Trans. Depend. Secure Comput. (2019).
[28] Y. Miao, J. Ma, X. Liu, X. Li, Q. Jiang, J. Zhang, Attribute-based keyword search over hierarchical data in cloud computing, IEEE Trans. Serv. Comput. (2017).
[29] Y. Miao, J. Weng, X. Liu, K.K.R. Choo, Z. Liu, H. Li, Enabling verifiable multiple keywords search over encrypted cloud data, Inf. Sci. 465 (2018) 21–37.
[30] S. Nagaraja, P. Mittal, C.Y. Hong, M. Caesar, N. Borisov, BotGrep: finding P2P bots with structured graph analysis, in: USENIX Security Symposium, 2010, vol. 10, pp. 95–110.
[31] M. Nagy, E. De Cristofaro, A. Dmitrienko, N. Asokan, A.R. Sadeghi, Do I know you?: Efficient and privacy-preserving common friend-finder protocols and applications, in: Proceedings of the 29th Annual Computer Security Applications Conference, 2013, ACM, pp. 159–168.
[32] A. Narayanan, N. Thiagarajan, M. Lakhani, M. Hamburg, D. Boneh, Location privacy via private proximity testing, in: NDSS, vol. 11, 2011.
[33] B. Pinkas, T. Schneider, G. Segev, M. Zohner, Phasing: Private set intersection using permutation-based hashing, in: 24th USENIX Security Symposium (USENIX Security 15), 2015, pp. 515–530.
[34] B. Pinkas, T. Schneider, M. Zohner, Scalable private set intersection based on ot extension, ACM Trans. Privacy Security (TOPS) 21 (2) (2018) 7.
[35] S. Qiu, J. Liu, Y. Shi, M. Li, W. Wang, Identity-based private matching over outsourced encrypted datasets, IEEE Trans. Cloud Comput. (2015).
[36] A. Sahai, B. Waters, Fuzzy identity-based encryption, in: Annual International Conference on the Theory and Applications of Cryptographic Techniques, Springer, Berlin, Heidelberg, 2005, pp. 457–473.
[37] A. Shamir, Identity-based cryptosystems and signature schemes, in: Workshop on the Theory and Application of Cryptographic Techniques, 1984, Springer, Berlin, Heidelberg, pp. 47–53.
[38] J. Shen, T. Zhou, X. Chen, J. Li, W. Susilo, Anonymous and traceable group data sharing in cloud computing, IEEE Trans. Inf. Forensics Security 13 (4) (2018) 912–925.
[39] The python pairing based cryptography library. https://github.com/debatem1/pypbc.
[40] X.S. Wang, Y. Huang, Y. Zhao, H. Tang, X. Wang, D. Bu, Efficient genome-wide, privacy-preserving similar patient query based on private edit distance, in: Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, 2015, pp. 492–503.
[41] E. Zhang, F. Li, B. Niu, Y. Wang, Server-aided private set intersection based on reputation, Inf. Sci. 387 (2017) 180–194.
[42] Q. Zheng, S. Xu, Verifiable delegated set intersection operations on outsourced encrypted data, in: 2015 IEEE International Conference on Cloud Engineering, 2015, IEEE, pp. 175–184.
[43] Z. Zhou, D. Huang, Z. Wang, Efficient privacy-preserving ciphertext-policy attribute based-encryption and broadcast encryption, IEEE Trans. Comput. 64 (1) (2015) 126–138.