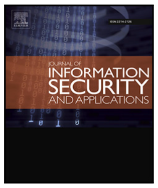
Journal of Information Security and Applications 84 (2024) 103823

Contents lists available at ScienceDirect

Journal of Information Security and Applications

journal homepage: www.elsevier.com/locate/jisa



Statistical privacy protection for secure data access control in cloud[☆]

Yaser Baseri[a,*], Abdelhakim Hafid[a], Mahdi Daghmehchi Firoozjaei[b], Soumaya Cherkaoui[c], Indrakshi Ray[d]

[a] *Department of Computer Science and Operations Research, Universite de Montreal, Canada*
[b] *Department of Computer Science, MacEwan University, Canada*
[c] *Department of Computer Engineering and Software Engineering, Ecole Polytechnique Montréal, Canada*
[d] *Computer Science Department, Colorado State University, USA*

ARTICLE INFO

Keywords:
Attribute-based encryption
User anonymity
Attributes statistical analysis
Data privacy
Computation outsourcing

ABSTRACT

Cloud Service Providers (*CSPs*) allow data owners to migrate their data to resource-rich and powerful cloud servers and provide access to this data by individual users. Some of this data may be highly sensitive and important and *CSPs* cannot always be trusted to provide secure access. It is also important for end users to protect their identities against malicious authorities and providers, when they access services and data. *Attribute-Based Encryption* (*ABE*) is an end-to-end public key encryption mechanism, which provides secure and reliable fine-grained access control over encrypted data using defined policies and constraints. Since, in *ABE*, users are identified by their attributes and not by their identities, collecting and analyzing attributes may reveal their identities and violate their anonymity. Towards this end, we define a new anonymity model in the context of *ABE*. We analyze several existing anonymous *ABE* schemes and identify their vulnerabilities in user authorization and user anonymity protection. Subsequently, we propose a *Privacy-Preserving Access Control Scheme (PACS)*, which supports multi-authority, anonymizes user identity, and is immune against users collusion attacks, authorities collusion attacks and chosen plaintext attacks. We also propose an extension of *PACS*, called *Statistical Privacy-Preserving Access Control Scheme (SPACS)*, which supports statistical anonymity even if malicious authorities and providers statistically analyze the attributes. Lastly, we show that the efficiency of our scheme is comparable to other existing schemes. Our analysis show that *SPACS* can successfully protect against *Collision Attacks* and *Chosen Plaintext Attacks*.

1. Introduction

Cloud computing has emerged as a promising technology that makes use of on-demand and scalable computing resources and reduces the operational costs of individual users and enterprises. It enhances collaboration, agility and scalability, and provides a global computing model over the Internet infrastructure. However, because of the challenges associated with security and privacy, there is a widespread concern in using this type of technology. Secure data protection and authorized access provision are some of these challenges whose violation can cause unauthorized use of resources and services [1,2].

Attribute-Based Encryption (*ABE*) is a cryptographic technique that facilitates access decisions based on attributes and policies while maintaining encryption. It empowers data owners to define access policies and encrypt data accordingly, enabling fine-grained access control. One prominent form of ABE is *Ciphertext Policy ABE* (*CP-ABE*), which allows encrypted data to be decrypted only by users possessing the requisite set of attributes. This verification is typically managed by either a single trusted authority or multiple authorities, each responsible for a subset of attributes [3–8]. The attributes associated with users and access policies embedded in ciphertexts determine access privileges, making ABE suitable for large-scale applications [9].

Security and privacy are important for the data content stored in the cloud. The privacy protection of users and preservation of their anonymity against malicious authorities and providers is often needed. In some applications, such as e-health, there may be some attributes containing sensitive and private information like types of diseases that must be kept secret from malicious authorities and *Cloud Service Providers (CSPs)* [10,11]. Moreover, malicious authorities and providers

☆ This document is the results of the research project funded by the Fonds de recherche du Québec – Nature et technologies (FRQNT), Canada and the Natural Sciences and Engineering Research Council of Canada (NSERC).

* Corresponding author.

E-mail addresses: yaser.baseri@umontreal.ca (Y. Baseri), ahafid@iro.umontreal.ca (A. Hafid), mahdi.firoozjaei@macewan.ca (M.D. Firoozjaei), soumaya.cherkaoui@polymtl.ca (S. Cherkaoui), ray@colostate.edu (I. Ray).

https://doi.org/10.1016/j.jisa.2024.103823

Available online 27 June 2024

may combine a subset of users' attributes and utilize this combination to re-identify users with a high degree of probability. For example, 1990 U.S. Census summary data shows that 87% of the population in the United States had reported attributes that likely made them unique based only on 5-digit ZIP code, gender and date of birth [12].

Protecting critical and sensitive information of users can be done either by (1) anonymizing attributes containing critical and sensitive information or (2) anonymizing the combination of attributes containing users' individual-specific information, which may be used to re-identify them. Such combination of attributes is called the quasi-identifier of users, which can be used to re-identify them [13,14]. More specifically, in *ABE*, when a user sends his request to have access to a ciphertext encrypted and stored in the cloud, the adversary, with the help of malicious CSPs, can get the specific sets of attributes that a user possessed in order to have access. Thus, supporting anonymity of users requires providing anonymity for attributes or a subset of attributes of those users.

In this paper, first we introduce statistical anonymity model for attribute-based encryption. In particular, we define $\mathcal{K}$*-anonymity* of attributes, formulate quasi-identifier, and extend $\mathcal{K}$*-anonymity* model of attributes to quasi-identifier of users. Providing $\mathcal{K}$*-anonymity* for quasi-identifier of users makes users anonymous and protect them against attributes statistical analysis. It is worth noting that recognizing and protecting quasi-identifiers represent research topics in data mining as well [15–17]. Next, we provide cryptanalytic results for several existing contributions in anonymous ABE, specifically the works presented in [18–21].Our analysis reveals vulnerabilities in these approaches, highlighting their weaknesses and potential security risks. Then, we propose a *Privacy-Preserving Access Control Scheme (PACS)*, which supports multi-authority, anonymizes user identity (without trusting any authority or provider), and is immune against users collusion attacks, authorities collusion attacks and chosen plaintext attacks. To provide $\mathcal{K}$*-anonymity* and make *PACS* immune against statistical analysis, we propose an extension to *PACS*, called *Statistical Privacy-Preserving Fine-Grained Access Control (SPACS)*. *SPACS* supports user statistical anonymity without trusting authorities and providers. In this scheme, even if cloud service providers collude with adversary, they cannot guess the attributes embedded in quasi-identifier and consequently are unable to re-identify the user who has sent his request to access a ciphertext stored in the cloud.

The contributions of this paper can be summarized as follows:

1. We formulate statistical anonymity by introducing $\mathcal{K}$*-anonymity* model for attribute-based encryption. To the best of our knowledge, this is the first formal model to define anonymity for *ABE*.
2. We provide analytical results for evaluation of the existing works in the context and show their vulnerabilities in users collusion, authorities collusion, user authorization and user anonymity protection.
3. We propose *PACS* as a fine-grained attribute-based access control scheme and prove its anonymity against identity attacks, and immunity against users collusion attacks, authorities collusion attacks and chosen plaintext attacks.
4. We propose an extension to *PACS*, named *SPACS*, to support $\mathcal{K}$*-anonymity* for attributes and individual users in an untrusted network.

The structure of the rest of the paper is as follows: Section 2 discusses related works and the mechanisms and techniques they employ to preserve and protect anomalies in attribute-based encryption. Section 3 presents some preliminary information needed to understand our work. Section 4 describes the anonymity model we formulate for attribute-based encryption. Section 5 analyzes and evaluates the security of several related works. Section 6 discusses the system and security models. Section 7 describes the details of the proposed scheme *PACS*. Section 8 analyzes the security of the proposed *PACS*; it shows the security of the scheme against authorities collusion attacks, users collusion attacks and chosen plaintext attacks. It also discusses how *PACS* anonymizes users' identities. Section 9 presents *SPACS* as an extension of *PACS* to support statistical anonymity. It discusses how the proposed *SPACS* makes users' identities immune against statistical analysis of their attributes and keeps the scheme secure against chosen plaintext attacks. Section 10 evaluates the performance of the proposed schemes. Finally, Section 11 concludes the paper and presents future work.

2. Related works

Preserving the privacy and anonymity of the user is a crucial aspect of attribute-based encryption (ABE) to ensure secure and confidential communication. Several studies have focused on providing privacy and anonymity for attribute-based encryption [18–27]. These studies employ two main mechanisms to achieve anonymity in attribute-based encryption: (1) hiding the user's attributes or (2) hiding policy information.

The first mechanism involves concealing the attributes associated with the user, preventing unauthorized entities from deducing the user's identity or personal information. Various techniques are used to achieve attribute hiding in ABE. Common techniques for achieving attribute hiding in ABE include: (a) using proxy re-encryption to encrypt messages for specific attributes, while the user anonymizes the attributes through a proxy [18,19,22]. (b) encoding the attributes of the user in a Bloom filter and encrypting the filter along with the message [24,27]. (c) transforming attributes into an anonymous form by adding noise, such as via differential privacy techniques [25]. (d) using pseudonyms or arbitrary labels, or employing group-based techniques where multiple users share common attributes or access policies [20,21].

The second mechanism focuses on hiding the access policy associated with the ciphertext. By concealing the access structure, unauthorized entities are unable to infer the targeted attributes or decipher the encrypted message. Some examples of this are given below. (a) Policy encryption to conceal the access policy associated with the ciphertext. By encrypting the policy, unauthorized entities are unable to infer the targeted attributes or decipher the encrypted message [26]. (b) Obfuscation to hide the policy logic by transforming it into an unintelligible form; it can be achieved through techniques like homomorphic encryption, functional encryption, or group-based encryption [27].

In this study, we analyze and evaluate several privacy preserving anonymous attribute-based encryption schemes [18–23]. In [22], Zhang et al. proposed *match-then-decrypt* technique to anonymously verify the users' legitimacy, match the ciphertext policy by CSPs before decryption, and reduce the computation overhead of users in decryption process. In [18], Zhang et al. extended [22] and proposed *match-then-re-encrypt* technique to provide anonymous ciphertext-policy attribute-based proxy re-encryption. In [19], they proposed an anonymous *CP-ABE* scheme secure against adaptive chosen-ciphertext attack, which extends their previous contribution [22]. However, the verification test used to anonymously match the policy before decryption, in [18,19,22], can disclose the policy embedded in ciphertext and consequently reveal the quasi-identifiers of users. In [23], Nasiraee et al. proposed a mechanism that hides and protects the privacy of policies used for attributes (metadata) against colluding parties. Their approach ensures user anonymity against colluding untrusted (honest-but-curious) authorities. In their scheme, they did not consider statistical de-anonymization approaches, which could be used to identify users. In [20], Jung et al. extended [28] and proposed a multi-authority access control scheme and reported that their work supports data privacy, users' anonymity and data access privilege. To protect users' identities and achieve full anonymity, they introduced *1-Out-of-n Oblivious Transfer* crypto primitive to anonymously select a value for each attribute requested by a user. However, a malicious user can request to have any other attribute value from the same set, which may not necessarily belong to him. Moreover, in their scheme, a malicious user can generate secret corresponding to each attribute and have access to a file he is not eligible to (see Section 5 for more details). In [21], Jung et al. extended [28] to make it immune against the leakage of master secret key. However, their scheme suffers from user and authority collusion attacks. We can summarize the vulnerabilities of [18–22] as follows (see more details in Section 5): (1) user anonymity attacks [18,19,22]; (2) user authorization attacks [20]; (3) coarse-grained access control [20,21]; and (4) user/authority collusion attacks [21].

3. Preliminaries

In this section, first we briefly introduce prime order bilinear group. Next, we present concepts related to tree access structure and Pedersen commitment scheme which will be used to design the proposed schemes.

3.1. Bilinear group

Definition 1. Let $\mathbb{G}$ and $\mathbb{G}_T$ be two cyclic multiplicative groups of prime order p, g be a generator of $\mathbb{G}$, and α and β be two random exponents in $\mathbb{Z}_p$. We call $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ a bilinear pairing, if it is a map with the following properties:

- Bilinearity: $\forall u, v \in \mathbb{G} : e(u^\alpha, v^\beta) = e(u, v)^{\alpha\beta}$.
- Non-degeneracy: $e(g, g) \neq 1$.
- Computability: $\forall u, v \in \mathbb{G}$, there is an efficient algorithm to compute $e(u, v)$.

We refer to the quadruple $(p, \mathbb{G}, \mathbb{G}_T, e)$ as bilinear group of order p.

3.2. Tree access structure

To enforce fine-grained access control and describe encryption policy, we adapt tree access structure, defined in [29], to our scheme. Let $\mathbb{A}_u$ be the user-associated set of attributes, $\mathcal{T}$ be a tree with root R representing an access structure, $\mathcal{T}_x$ be a sub-tree of $\mathcal{T}$ rooted at node x, $att(x)$ denotes the attribute associated with leaf node x, num_x be the number of child nodes of non-leaf node x, and k_x be a threshold value $0 \leq k_x \leq num_x$. Node x will be assigned true if at least k_x child nodes of x are assigned true; otherwise, it will be assigned false. Particularly, the node becomes OR gate when $k_x = 1$ and AND gate when $k_x = num_x$.

Definition 2 (*Satisfying a Tree Access Structure*). We say that user-associated set of attributes $\mathbb{A}_u$ satisfies tree access structure $\mathcal{T}_x$, if and only if function $\mathcal{T}_x(\mathbb{A}_u)$ returns 1. $\mathcal{T}_x(\mathbb{A}_u)$ can be calculated recursively as follows: If x is a leaf node, then $\mathcal{T}_x(\mathbb{A}_u)$ is equal to 1 if and only if $att(x) \in \mathbb{A}_u$. If x is a non-leaf node, then $\mathcal{T}_x(\mathbb{A}_u)$ is equal to 1 when at least k_x child nodes $y_{i_k(1\leq k\leq k_x)}$ of x satisfy $\mathcal{T}_{y_{i_k}}(\mathbb{A}_u) = 1$. $\mathcal{T}(\mathbb{A}_u) = 1$ if and only if $\mathcal{T}_R(\mathbb{A}_u) = 1$.

3.3. Commitment scheme

A commitment scheme is a cryptographic primitive that allows one to commit to a chosen value (or chosen statement) while keeping it hidden from others, with the ability to reveal the committed value later [30]. It is formalized as a reactive two party protocol between a sender and a receiver. In the first phase, the sender holds a secret s, picks a random r, "encodes" s using r and sends the encoded message (i.e. the commitment to s) to the receiver. In the second phase, the sender sends random r to the receiver. The receiver can open the commitment and find out the content of secret s.

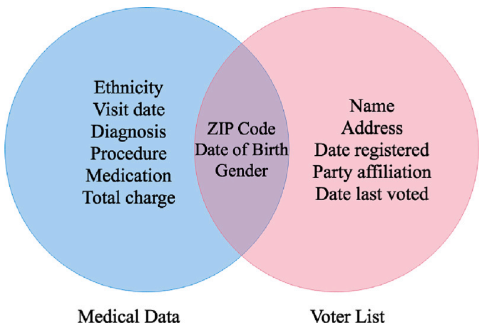


Fig. 1. Linking to re-identify data.

Definition 3 (*Pedersen Commitment Scheme [31]*). Let p and q be primes such that $q|p-1$, g and h be two generators for two subgroups of $\mathbb{Z}_p^*$ with order q. The Pedersen commitment scheme is defined as follows:

- To commit to $s \in \mathbb{Z}_q$, select a random number $r \in \mathbb{Z}_q$ and compute $Commit(s, r) = g^s h^r$.
- To open the commitment, simply reveal s and r.

Pedersen commitment scheme is an information theoretically hiding scheme and is computationally binding under the discrete logarithm assumption. The committer cannot open a commitment to s by $s' \neq s$ unless he can solve the discrete logarithm problem with a non-negligible advantage.

4. Anonymity model: From identity to attributes

ABE utilizes user attributes to enforce access policies over encrypted data. However, it is crucial to recognize that the statistical distributions of these attributes, when analyzed within the broader population, can construct an electronic profile that closely approximates or even uniquely identifies users. For example, Sweeney demonstrated in their work [32] that such statistical analysis can potentially lead to the identification of anonymous users. They illustrated that patients could be re-identified by cross-referencing the voting list of Cambridge, Massachusetts, with medical data released by the Group Insurance Commission (GIC), using personal information such as 5-digit ZIP code, gender, and date of birth. Thus, supporting anonymity of users requires providing anonymity for attributes or a subset of attributes of users.

The anonymity problem for attribute-based encryption can be modeled as follows: There are $\mathcal{N}$ attribute authorities $AA_{i(1\leq i\leq \mathcal{N})}$, each of them is in charge of a distinct subset of universal attribute set $\mathbb{A}$ associated with users. For each attribute, AA_i is responsible for, it knows the exact information of the users possessing that attribute. It also knows the value associated with that attribute for each user. Each user-specific set of attributes $\mathbb{A}_u = \{att(1), \ldots, att(n)\} \subseteq \mathbb{A}$ consists of two parts: (a) $\mathbb{A}_u^+ = \{att(i_1), \ldots, att(i_j)\}$, the quasi-identifier, whose assigned values contain identifying information about that user and their combination can uniquely identify the user with a high degree of probability, and (b) $\mathbb{A}_u^-$, other attributes whose values or combination of values do not disclose any identifying information about that user. Each of $\mathbb{A}_u^+$ and $\mathbb{A}_u^-$ can include critical or sensitive information of users.

Definition 4 (*Indistinguishable Attribute Values*). Let $\mathbb{A}$ be the set of attributes universe, $v_j(i)$ and $v_{j'}(i)$ be two values considered for attribute $att(i)$. $v_j(i)$ and $v_{j'}(i)$ are indistinguishable, if and only if there is no probabilistic polynomial time adversary $\mathcal{A}$ with non-negligible advantage to differentiate them.

Definition 5 (*Anonymous Equivalence Class*). An anonymous equivalence class $[v_{att}(i)]$ for attribute $att(i)$ is the set of all possible values $v_j(i)$, considered for $att(i)$ which are indistinguishable for adversary $\mathcal{A}$.

For example, in the case of $att(i) = Gender$, and $\{v_{att}(i)\} = \{male, female\}$, if the values $male$ and $female$, considered for attribute $Gender$ are indistinguishable, then $[v_{att}(i)] = \{male,\ female\}$.

Definition 6 (*$\mathcal{K}$-Anonymous Attribute*)**.** User U satisfies *$\mathcal{K}$-anonymity* with respect to attribute $att(i)$ if and only if its associated value $v_{att}(i)$ belongs to an equivalence class with at least $\mathcal{K}$ possible attribute values $\{v_j(i)\}_{1\leq j\leq \mathcal{K}}$.

$\mathcal{K}$-Anonymity, proposed by Samarati and Sweeney [33], guarantees that in a set of $\mathcal{K}$ objects with a similarity, the target object is indistinguishable from the other $\mathcal{K}$-1 objects. For example, in the case of $att(i) = Gender$, if $[v_{att}(i)] = \{male, female\}$ (i.e. two values $male$ and $female$, which are considered for attribute $Gender$, are indistinguishable), then attribute $Gender$ satisfies 2-*anonymity*.

Definition 7 (*Quasi-Identifier*)**.** Let $\mathbb{U}$ be a set of users, $\mathbb{A}$ be the set of attributes universe, $\langle v_{att}(1), \dots, v_{att}(n)\rangle$ be a sequence of values associated with $\langle att(1), \dots, att(n)\rangle$, $f_c : \mathbb{U} \to 2^{\mathbb{A}}$ is a function that maps each user to his associated attributes, and $f_g : 2^{\mathbb{A}} \to 2^{\mathbb{U}}$ is a map from a set of attributes to a subset of users supporting those attributes. Set of attributes $\mathbb{A}^+ = \{att(i_1), \dots, att(i_j)\} \subseteq \mathbb{A}$ is quasi-identifier of $\mathbb{A}$, if

1. $\exists u \in \mathbb{U}$ *such that* $f_g(f_c(u)\mid_{\mathbb{A}^+}) = \{u\}$ and
2. $\mathbb{A}^+$ is minimal.

where, $f_c(.)\mid_{\mathbb{A}^+}$ is the projection of $f_c(.)$ on quasi-identifier $\mathbb{A}^+$.

In the example shown in Fig. 1, $\mathbb{A}^+ = \{$*ZIP Code, Date of Birth, Gender*$\}$.

Definition 8 (*Indistinguishable Sequences of Attribute Values*)**.** Two sequences of attribute values $\langle v_{att}(i_1), \dots, v_{att}(i_j)\rangle$ and $\langle v'_{att}(i_1), \dots, v'_{att}(i_j)\rangle$, associated with the sequence of attributes $\langle att(i_1), \dots, att(i_j)\rangle$, are indistinguishable, if and only if there is no probabilistic polynomial time adversary $\mathcal{A}$ with non-negligible advantage to differentiate them.

Definition 9 (*$\mathcal{K}$-Anonymous User*)**.** Let $\mathbb{A}_u = \{att(1), \dots, att(n)\} \subseteq \mathbb{A}$ be the set of attributes associated with user U. User U is $\mathcal{K}$-anonymous, if and only if there are at least $\mathcal{K}$ possible indistinguishable sequences of attribute values $\langle v_{att}(i_1), \dots, v_{att}(i_j)\rangle$ for attributes sequence $\langle att(i_1), \dots, att(i_j)\rangle$, where $\mathbb{A}^+_u = \{att(i_1), \dots, att(i_j)\}$ is the projection of $\mathbb{A}_u$ on quasi-identifier $\mathbb{A}^+$.

Definition 10 (*$\mathcal{K}$-Anonymity*)**.** Let $\mathbb{A}$ be the set of attributes universe, $v_{att}(i)$ be an attribute value for attribute $att(i)$ and $\mathbb{A}^+ = \{att(i_1), \dots, att(i_j)\} \subseteq \mathbb{A}$ be a quasi-identifier associated with $\mathbb{A}$. We say that attribute-based access control scheme $\mathcal{P}$ satisfies $\mathcal{K}$-anonymity, if and only if there are at least $\mathcal{K}$ possible indistinguishable sequences of attribute values $\langle v_{att}(i_1), \dots, v_{att}(i_j)\rangle$ for attributes sequence $\langle att(i_1), \dots, att(i_j)\rangle$.

Proposition 1. *Let $\mathcal{P}$ be an attribute-based access control scheme with attribute universe $\mathbb{A}$, $\mathbb{A}^+ \subseteq \mathbb{A}$ be a quasi-identifier associated with $\mathbb{A}$, $\mathbb{A}_u = \{att(1), \dots, att(n)\} \subseteq \mathbb{A}$ be the set of attributes associated with user U, $\mathbb{A}^+_u \subseteq \mathbb{A}_u$ is the projection of $\mathbb{A}_u$ on quasi-identifier $\mathbb{A}^+$. If each user is $\mathcal{K}$-anonymous in $\mathcal{P}$, then $\mathcal{P}$ satisfies $\mathcal{K}$-anonymity.*

Proof. If $\mathcal{P}$ satisfies $\mathcal{K}$-anonymity for all users, then for all $\mathbb{A}^+_u$ ($\mathbb{A}^+_u \subseteq \mathbb{A}^+$), there are at least $\mathcal{K}$ possible indistinguishable sequences of attribute values. More specifically, since for any user U in the definition of quasi-identifier $\mathcal{K}$-anonymity does hold, we can deduce that there are at least $\mathcal{K}$ possible indistinguishable sequences of attribute values for $\mathbb{A}^+$ as well. Thus, $\mathcal{P}$ satisfies $\mathcal{K}$-anonymity. □

Each user, satisfying *$\mathcal{K}$-anonymity*, has a maximum probability $1/\mathcal{K}$ to be re-identified. Depending on statistical tolerance limit considered for re-identifying users in $\mathcal{P}$, different values can be considered for $\mathcal{K}$.

Table 1
Medical data information of users.

Ethnicity	Date of birth	Gender	Zip code	Diagnosis
African Americans	09/01/17	Male	02224	Short of breath
Caucasian Americans	09/05/00	Female	02224	Hypertension
African Americans	24/07/88	Male	02228	Chest pain
Caucasian Americans	09/11/67	Female	02228	Hypertension
Caucasian Americans	24/01/88	Female	02228	Hypertension
Caucasian Americans	29/10/76	Female	02228	Short of breath
African Americans	09/08/99	Male	02229	Obesity
African Americans	29/05/07	Female	02229	Short of breath
Caucasian Americans	11/03/01	Male	02226	Obesity
African Americans	18/06/13	Male	02226	Chest pain
Caucasian Americans	25/10/93	Female	02232	Short of breath
African Americans	02/04/73	Male	02231	Obesity

Example 1. Table 1 represents a sample of patients' medical records. For attribute $Ethnicity$, we have $v_{att}(i) = \{$*African Americans, Caucasian Americans*$\}$ and for attribute $Gender$, $v_{att}(i) = \{$*male, female*$\}$. It shows that the number of entries for *African Americans* is same as for *Caucasian Americans*. Same applies for the number of entries for $female$s and $male$s; but, in combination there is only one *Caucasian American female*. Moreover, of these attributes, $Gender$ alone could uniquely identify half of patients. Thus, only based on $Gender$, the adversary has non-negligible advantage to differentiate users (i.e. $P(Gender{=}male) - \frac{1}{2} = P(Gender{=}female) - \frac{1}{2} = \epsilon$). Consequently, values $male$ and $female$, considered for attribute $Gender$ are indistinguishable and $[v_{att}(i)] = \{$*male, female*$\}$. Furthermore, if the adversary only has access to $att(i) = Gender$, and if $[v_{att}(i)] = \{$*male, female*$\}$, then attribute $Gender$ satisfies 2-*anonymity*.

This example can be generalized for *ABE* in which attributes of a user can be deduced from a ciphertext, he requested to have access to, while the values of those attributes are embedded (hidden) in his own secret key. Moreover, the experiment, reported in [32], shows that one could uniquely identify 12% of the voters only with their *Date of Birth*, 29% with their *Date of Birth* and *Gender*, 69% with *Date of Birth* and *Zip Code* and 97% when their *Zip Code*, *Gender* and *Date of Birth* were used. Definition 7 states that a set of attributes is a quasi-identifier if a user can be uniquely identified by possessing certain attributes. According to this definition, such a combination of attributes, which could be used to uniquely identify users, is called quasi-identifier of user. For the running example, {*Zip Code*, *Date of Birth*, *Gender*} together construct the quasi-identifier for users. We say a scheme satisfies $\mathcal{K}$-anonymity, if and only if there are at least $\mathcal{K}$ possible indistinguishable values that could be considered for those attributes of user which belong to his quasi-identifier. For our example, there should be at least $\mathcal{K}$ possible indistinguishable values for the set {*Zip Code*, *Date of Birth*, *Gender*}, if we want to satisfy $\mathcal{K}$-anonymity of users.

5. Cryptanalysis and evaluation of related works

In order to address privacy protection of users and preservation of their anonymity against malicious authorities and providers, anonymous *ABE* has been studied in [18–22]. Zhang et al. [19] proposed an anonymous *CP-ABE* scheme, secure against adaptive chosen-ciphertext attack, which extends their contribution in [22]. More specifically, they introduced *match-then-decrypt* technique to provide a light-weight matching phase before decryption and verify whether the user's attributes match the hidden access policy embedded in ciphertext without engaging in decryption process. Their scheme [22] consists of four algorithms:

Setup(λ): Let $\mathbb{G}$, $\mathbb{G}_T$ be cyclic multiplicative groups of prime order p, $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ be a bilinear map. $H : \{0,1\}^* \to \mathbb{G}$ be a hash function which maps any attribute value to an element in $\mathbb{G}$. Assume that there are n attributes $\omega_1, \dots, \omega_n$ in attribute universe $\mathcal{U}$ (i.e. $\mathcal{U} = \{\omega_1, \dots, \omega_n\}$), each of them has n_i different values $S_i = \{v_{i,1}, \dots, v_{i,n_i}\}$.

Attribute authority AA chooses random elements $y \in_R \mathbb{Z}_p$, $g_1, g_2 \in_R \mathbb{G}$ and computes $Y = e(g_1, g_2)^y$. Then, it publishes $PK = (g, g_1, g_2, Y)$ as system public key and keeps $MK = (y)$ secret as its own master secret key.

KeyGeneration(PK, MK, L): Let $L = \left[L_1, \ldots, L_n\right]$ be the attribute list for the user who wants to obtain the corresponding attribute secret key SK_L, v_{i,k_i} be the attribute value whose index satisfies $L_i = v_{i,k_i}$, $(SigKeyGen, Sign, Verify)$ be a strongly existentially unforgeable signature scheme and (K_s, K_v) be the one-time signing-verification key pair generated by $SigKeyGen$ in which K_v has v bits. AA chooses random elements $r_1, r_2, \ldots, r_{n+v-1}, r, \lambda, \widehat{\lambda}, \lambda_v, \widehat{\lambda}_v, \{\widehat{r}_i\}_{1\le i\le n+v} \in_R \mathbb{Z}_p$, sets $r_{n+v} = y - \sum_{i=1}^{n+v-1} r_i \mod p$ and $\widehat{r} = \sum_{i=1}^{n+v} \widehat{r}_i \mod p$, and computes $[D_{\Delta,0}, \widehat{D}_{\Delta,0}] = [g_1^r, g_2^{y-\widehat{r}}]$, $D_0 = g_2^{\lambda}$, $\widehat{D}_0 = g_1^{\widehat{\lambda}}$, $\{[D_{\Delta,i}, D_{i,1}, \widehat{D}_{i,1}]\}_{1\le i\le n} = \{[g_2^{\widehat{r}_i} H(i \parallel v_{i,k_i})^r, g_1^{r_i} H(0\|i\|v_{i,k_i})^{\lambda}, g_2^{r_i} H(1\|i\|v_{i,k_i})^{\widehat{\lambda}}]\}_{1\le i\le n}$, $[D_{0,v}, \widehat{D}_{0,v}] = [g_2^{\lambda_v}, g_1^{\widehat{\lambda}_v}]$ and $\{[D_{\Delta,i}, D_{i,1}, \widehat{D}_{i,1}]\}_{n+1\le i\le n+v} = \{[g_2^{\widehat{r}_i} H(i \parallel v)^r, g_1^{r_i} H(0\|i\|v)^{\lambda}, g_2^{r_i} H(1\|i\|v)^{\widehat{\lambda}}]\}_{n+1\le i\le n+v}$. The attribute secret key SK_L would be $(D_{\Delta,0}, \widehat{D}_{\Delta,0}, D_0, \widehat{D}_0, [D_{0,v}, \widehat{D}_{0,v}], \{[D_{\Delta,i}, D_{i,1}, \widehat{D}_{i,1}]\}_{1\le i\le n}, \{[D_{\Delta,i}, D_{i,1}, \widehat{D}_{i,1}]\}_{n+1\le i\le n+v})$,

AnonEncrypt (PK, M, W): Let us assume that data owner (DO) encrypts message $M \in \mathbb{G}_T$ under access policy $W = \left[W_1, \ldots, W_n\right]$, where W_i is a policy corresponding to attribute ω_i. Then, DO chooses random elements $s, s', s\varepsilon \in_R \mathbb{Z}_p$ and sets $\widetilde{C} = MY^s$, $C_\Delta = Y^{s'}$, $\widehat{C}_0 = g_1^{s'}$, $C_1 = g_2^{s''}$, $\widehat{C}_1 = g_1^{s-s''}$, and computes $\{\{C_{i,t,\Delta}, C_{i,t,0}, \widehat{C}_{i,t,0}\}_{1\le t\le n_i}\}_{1\le i\le n}$, as follows:

1. If $v_{i,t} \in W_i$, then $[C_{i,t,\Delta}, C_{i,t,0}, \widehat{C}_{i,t,0}] = [\sigma_{i,\Delta} H(i \parallel v_{i,t})^{s'}, \sigma_{i,0} H(0\|i\|v_{i,t})^{s''}, \sigma_{i,1} H(1\|i\|v_{i,t})^{s-s''}]$,
2. If $v_{i,t} \notin W_i$, then $[C_{i,t,\Delta}, C_{i,t,0}, \widehat{C}_{i,t,0}]$ will be random elements;

where $\sigma_{i,\Delta}(1 \le i \le n) \in \mathbb{G}$ are random elements generated by DO such that $\prod_{i=1}^n \sigma_{i,\Delta} = 1_{\mathbb{G}}$. In addition, for $n+1 \le i \le n+v$, DO computes $[C_{i,\Delta}, C_{i,0}, \widehat{C}_{i,0}] = [H(i \parallel v)^s, H(0\|i\|v)^{s'}, H(0\|i\|v)^{s-s''}]$. It computes a signature $\sigma = Sign(CT_0, K_v, K_s)$, where $CT_0 = \left(\widetilde{C}, C_\Delta, \widehat{C}_0, C_1, \widehat{C}_1, \{\{C_{i,t,\Delta}, C_{i,t,0}, \widehat{C}_{i,t,0}\}_{1\le t\le n_i}\}_{1\le i\le n}, \{C_{i,\Delta}, C_{i,0}, \widehat{C}_{i,0}\}_{n+1\le i\le n+v}\right)$. Finally, the ciphertext of M with respect to W is output as $CT_W = (CT_0, K_v, \sigma)$.

AnonDecrypt (PK, CT_W, SK_L): The ciphertext CT_W is tested and decrypted by user with secret key SK_L as follows:

1. Matching: user checks whether $L \vDash W$ in terms of the following equation:
$$e(\prod_{i=1}^{n} C_{i,t,\Delta} \prod_{i=n+1}^{n+v} C_{i,\Delta}, D_{\Delta,0}) \overset{?}{=} e(\widehat{C}_0, \widehat{D}_{\Delta,0} \prod_{i=1}^{n+v} D_{\Delta,i})$$
2. *Decryption*: If the indexes satisfy $L_i = v_{i,t}$, user can compute plaintext M as follows:
$$M = \widetilde{C}.\frac{e(\prod_{i=1}^n C_{i,t,0}, D_0).e(\prod_{i=1}^n \widehat{C}_{i,t,0}, \widehat{D}_0)}{e(C_1, \prod_{i=n+1}^{n+v} D_{i,1})} \cdot \frac{e(\prod_{i=n+1}^{n+v} C_{i,0}, D_{0,v}).e(\prod_{i=n+1}^{n+v} C_{i,0}, \widehat{D}_{0,v})}{e(\widehat{C}_1, \prod_{i=n+1}^{n+v} \widehat{D}_{i,1})}$$

Our analysis shows that the proposed scheme suffers from the following problem:

User Anonymity Attacks: In the proposed scheme, the authors assumed that there are n attributes $\omega_i(1 \le i \le n)$, each ω_i has n_i different values $S_i = \{v_{i,1}, \ldots, v_{i,n_i}\}$. Now, let us assume that DO encrypts message M under access policy $W = \left[W_1, \ldots, W_n\right]$, where W_i is a policy corresponding to attribute ω_i. Then, DO uploads ciphertext $CT = (C_\Delta, \widehat{C}_0, \widetilde{C}, C_1, \widehat{C}_1, \{\{C_{i,t,\Delta}, C_{i,t,0}, \widehat{C}_{i,t,0}\}_{1\le t\le n_i}\}_{1\le i\le n}, \{C_{i,\Delta}, C_{i,0}, \widehat{C}_{i,0}\}_{n+1\le i\le n+v})$ to the server. More specifically, for all $v_{i,t}$, if $v_{i,t} \in W_i$, then $C_{i,t,\Delta}$ will be equal to $\sigma_{i,\Delta} H(i \parallel v_{i,t})^{s'}$, where s' is a secret selected by DO and $\sigma_{i,\Delta}(1 \le i \le n)$ are random elements in $\mathbb{G}$ such that $\prod_{i=1}^n \sigma_{i,\Delta} = 1_{\mathbb{G}}$; otherwise, $C_{i,t,\Delta}$ will be a random element. For each policy W_i embedded in ciphertext CT, the adversary may select different values $v_{i,\bar{t}(1\le \bar{t}\le n_i)}$, compute $H(i \parallel v_{i,\bar{t}})$ and check whether $e(\prod_{i=1}^n C_{i,t,\Delta}, g_1) \overset{?}{=} e(\hat{C}_0, \prod_{i=1}^n H(i \parallel v_{i,\bar{t}}))$, where $\hat{C}_0$ is equal to $g_1^{s'}$. If the answer is yes, it can infer that the selected tuple $[v_{1,\bar{t}}, \ldots, v_{n,\bar{t}}]$ is a valid attribute tuple which is included in the ciphertext. Thus, the hidden policy $W = \left[W_1, \ldots, W_n\right]$ will be disclosed and the adversary can guess the attributes (and consequently identity) of any user who requests to access the ciphertext.

In another work, Zhang et al. [18] proposed *match-then-re-encrypt* scheme to provide a light-weight matching phase before re-encryption and updating access policy. The scheme consists of four main algorithms *Setup*, *KeyGeneration*, *Encryption* and *Decryption*, which are described as follows:

Setup(λ): This algorithm which is run by attribute authority AA sets authority's master secret key MK and publishes system public key PK. It chooses two cyclic multiplicative groups $\mathbb{G}$, $\mathbb{G}_T$ of prime order p, bilinear map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ from $\mathbb{G} \times \mathbb{G}$ to $\mathbb{G}_T$, hash function $H : \{0,1\}^* \to \mathbb{G}$ which maps any attribute value to a random element in $\mathbb{G}$, generator g for group $\mathbb{G}$, random elements $y \in_R \mathbb{Z}_p, g_2, g_3 \in_R \mathbb{G}$ and encoding function $E : \mathbb{G} \to \mathbb{G}_T$ as an encoding between $\mathbb{G}$ and $\mathbb{G}_T$. Assumed that there are n attributes $\omega_i(1 \le i \le n)$ in attribute universe $\mathcal{U} = \{\omega_1, \ldots, \omega_n\}$, each of them can take its value from n_i different values $S_i = \{v_{i,1}, \ldots, v_{i,n_i}\}$. For each attribute $\omega_i(1 \le i \le n)$, the algorithm chooses $\{T_{i,t} \in_R \mathbb{G}, a_{i,t} \in_R \mathbb{Z}_p, b_{i,t} \in_R \mathbb{Z}_p\}_{1\le t\le n_i}$ and computes $\{\{A_{i,t} = T_{i,t}^{a_{i,t}}, B_{i,t} = T_{i,t}^{b_{i,t}}\}_{1\le t\le n_i}\}_{1\le i\le n}$. It also computes $g_1 = g^y, Y = e(g, g_1, g_2)$ and publishes $PK = (g, g_1, g_2, g_3, Y, \{\{T_{i,t}, A_{i,t}, B_{i,t}\}_{1\le t\le n}\}_{1\le i\le n})$ as system public key and keeps $MK = (y, \{\{a_{i,t}, b_{i,t}\}_{1\le t\le n}\}_{1\le i\le n})$ secret as AA's own master secret key.

KeyGeneration(PK, MK, L): Let $L = \left[L_1, \ldots, L_n\right]$ be the attribute list for the user who wants to obtain the corresponding attribute secret key SK_L. AA chooses random elements r', $\bar{d}$, $\{r_i \widehat{r}_i, \lambda_i\}_{1\le i\le n} \in_R \mathbb{Z}_p$, sets $r = y - \sum_{i=1}^n r_i \mod p$ and $\widehat{r} = \sum_{i=1}^n \widehat{r}_i \mod p$, and computes $[D_0, \hat{D}, \bar{D}, D_{\Delta,0}] = [g_2^{y-r}, g_2^{y-\widehat{r}}, g_3^d, g^{r'}]$. For $1 \le i \le n$, suppose that v_{i,k_i} be the attribute value whose index satisfies $L_i = v_{i,k_i}$. Then, AA computes $[D_{\Delta,i}, D_{i,0}, D_{i,1}, D_{i,2}] = [g_2^{\widehat{r}_i} T_{i,k_i}^{r'}, g_2^{r_i} B_{i,k_i}^{\lambda_i a_{i,k_i}}, g^{\lambda_i a_{i,k_i}}, g^{\lambda_i b_{i,k_i}}]$ for all $1 \le i \le n$. The attribute secret key SK_L would be $\Big(D_0, \hat{D}_0, \bar{D}_0, D_{\Delta,0}, \{[D_{\Delta,i}, D_{i,0}, D_{i,1}, D_{i,2}]\}_{1\le i\le n} \Big)$,

Encrypt (PK, M, W): Let us assume that data owner DO encrypts message $M \in \mathbb{G}_T$ under access policy $W = \left[W_1, \ldots, W_n\right]$, where W_i is a policy corresponding to attribute ω_i. Then, DO chooses random elements $s, s', s'' \in_R \mathbb{Z}_p$ and sets $\widetilde{C} = MY^s$, $C_0 = g^s$, $C_{RE} = g_3^s$, $C_\Delta = Y^{s'}$, $C'_0 = g^{s'}$. DO also selects random elements $\sigma_{i,\Delta}(1 \le i \le n) \in \mathbb{G}$, such that $\prod_{i=1}^n \sigma_{i,\Delta} = 1_{\mathbb{G}}$, and computes $\{\{C_{i,t,\Delta}, C_{i,t,1}, C_{i,t,2}\}_{1\le t\le n_i}\}_{1\le i\le n}$ as follows:

1. If $v_{i,t} \in W_i$, then $[C_{i,t,\Delta}, C_{i,t,1}, C_{i,t,2}] = [\sigma_{i,\Delta} T_{i,t}^{s'}, B_{i,t}^{s-s''}, A_{i,t}^{s''}]$,
2. If $v_{i,t} \notin W_i$, then $[C_{i,t,\Delta}, C_{i,t,1}, C_{i,t,2}]$ will be random elements in $\mathbb{G}$.

Finally, the ciphertext of M with respect to W is $CT_W = (C_\Delta, C'_0, \widetilde{C}, C_0, C_{RE}, \{\{C_{i,t,\Delta}, C_{i,t,1}, C_{i,t,2}\}_{1\le t\le n_i}\}_{1\le i\le n})$.

Decrypt (PK, CT_W, SK_L): The ciphertext CT_W is tested and decrypted by user with secret key SK_L as follows:

1. Matching: user checks whether $L \vDash W$ $(\exists k_1, \ldots, k_n : L_1 = v_{1,k_1}, \ldots, L_n = v_{n,k_n})$ in terms of the following equation:
$$C_\Delta = \frac{e(C'_0, \hat{D}_0 \prod_{i=1}^n D_{\Delta,i})}{e(\prod_{i=1}^n C_{i,k_i,\Delta}, D_{\Delta,0})}$$
2. *Decryption*: If the indexes satisfy $L_i = v_{i,k_i}$, user can computes plaintext M as follows:
$$M = \frac{\widetilde{C}.e(\prod_{i=1}^n C_{i,k_i,1}, D_{i,1}).C_{i,k_i,2}, D_{i,2}}{e(C_0, D_0). \prod_{i=1}^n e(C_0, D_{i,0})}$$

Zhang et al. [18] also introduced two more algorithms (i.e. *RKGen*, and *Reencrypt*) and extended *Decrypt* algorithm to support proxy re-encryption and access policy updating. However, their anonymous attribute-based encryption scheme suffers from the following problem:

User Anonymity Attacks: Similar to [19,22], the authors assumed n different attributes $\omega_i (1 \leq i \leq n)$, each of them has n_i multiple values $S_i = \{v_{i,1}, \dots, v_{i,n_i}\}$. Now, let us assume that *DO* encrypts message M under access policy $W = \left[W_1, \dots, W_n\right]$, where W_i is a policy that corresponds to attribute ω_i. Then, *DO* uploads ciphertext $CT = (C_{\Delta}$, C'_0, $\widetilde{C}$, C_0, C_{RE}, $\{\{C_{i,t,\Delta}, C_{i,t,1}, C_{i,t,2}\}_{1 \leq t \leq n_i}\}_{1 \leq i \leq n})$ to the server. In this scheme, if $v_{i,t} \in W_i$, then $C_{i,t,\Delta}$ will be equal to $\sigma_{i,\Delta} T_{i,t}^{s'}$, where s' is a secret element selected by *DO* and $\sigma_{i,\Delta} (1 \leq i \leq n)$ are random elements in $\mathbb{G}$ such that $\prod_{i=1}^{n} \sigma_{i,\Delta} = 1_{\mathbb{G}}$ and $T_{i,t}$ are parts of public parameter PK published earlier in *Setup* process. If $v_{i,t} \notin W_i$, then $C_{i,t,\Delta}$ will be random elements in $\mathbb{G}$. For each policy W_i embedded in ciphertext CT, the adversary may select different values $v_{i,\bar{t}(1 \leq \bar{t} \leq n_i)}$ and check whether $e(\prod_{i=1}^{n} C_{i,t,\Delta}, g) \overset{?}{=} e(C'_0, \prod_{i=1}^{n} T_{i,\bar{t}})$, where C'_0 is equal to $g^{s'}$. If the answer is yes, it can infer that the selected tuple $[v_{1,\bar{t}}, \dots, v_{n,\bar{t}}]$ is a valid attribute tuple which is included in the ciphertext. Thus, the hidden policy $W = \left[W_1, \dots, W_n\right]$ will be disclosed and the adversary can guess the attributes and consequently quasi-identifier of the user who has requested to access the ciphertext.

It is worth noting that the complexity of executing user anonymity attacks is similar to the complexity of executing the decryption process. To conclude, the schemes in [18,19,22], in addition to their vulnerability to user anonymity attacks, they do not make use of tree access structures but rather integrate all attribute constraints into one piece of ciphertext allowing for only restricted forms of policies (i.e. the policies in the form of conjunctions of atomic formulas).

Jung et al. [20] proposed a multi-authority access control scheme (an extension to the contribution in [28]) and reported that their work supports data privacy, user anonymity and data access privilege. They also introduced *1-Out-of-k Oblivious Transfer* crypto primitive to protect user's identity and achieve full anonymity. Their proposed scheme consists of the following four algorithms:

Setup$(\mathbb{A}, \Lambda)$: The algorithm, which is run by attribute authorities $AA_{j(j \in \{1,\dots,\mathcal{N}\})}$, takes as input the attribute universe $\mathbb{A}$ and an implicit security parameter Λ. In this algorithm, one of the authorities selects and publishes bilinear group $\mathbb{G}_0$, prime number p, and generator g. Moreover, each attribute authority AA_k picks random parameters $v_k, s_{kj(j \in \{1,\dots,\mathcal{N}\} \backslash \{k\})} \in \mathbb{Z}_p^*$, computes $Y_k = e(g,g)^{v_k}$ and $g^{s_{kj}}$, and shares them with all other authorities. Receiving $\mathcal{N} - 1$ pieces of $g^{s_{jk}}$ and $Y_j = e(g,g)^{v_j}$ generated by other authorities $AA_{j(j \in \{1,\dots,\mathcal{N}\} \backslash \{k\})}$, it computes $Y = \prod_{k=1}^{\mathcal{N}} e(g,g)^{v_k} = e(g,g)^{\sum_{k=1}^{\mathcal{N}} v_k}$ and $x_k = \left(\prod_{j \in \{1,\dots,\mathcal{N}\} \backslash \{k\}} g^{s_{kj}}\right) / \left(\prod_{j \in \{1,\dots,\mathcal{N}\} \backslash \{k\}} g^{s_{jk}}\right) = g^{\left(\sum_{j \in \{1,\dots,\mathcal{N}\} \backslash \{k\}} g^{s_{kj}} - \sum_{j \in \{1,\dots,\mathcal{N}\} \backslash \{k\}} g^{s_{jk}}\right)}$. Then, the master secret key for AA_k is $\{v_k, x_k\}$ and the public parameter of the whole system is $PK = (\mathbb{G}_0, g, Y)$.

KeyGeneration$(PK, MK_{a_k(1 \leq k \leq \mathcal{N})}, \mathbb{A}_u)$: The key generation algorithm, which is run by attributes authorities $AA_{k(1 \leq k \leq \mathcal{N})}$, takes as input public parameters PK, master secret keys $MK_{a_k(1 \leq i \leq \mathcal{N})}$ and user's set of attributes $\mathbb{A}_u$. To perform *KeyGeneration*, each authority AA_k selects a random parameter $d_k \in \mathbb{Z}_p^*$, computes $x_k.g^{(v_k + d_k)}$ and shares it with other authorities. Receiving $\mathcal{N} - 1$ pieces $x_j.g^{(v_j + d_j)}$ generated by other authorities $AA_{j(j \in \{1,\dots,\mathcal{N}\} \backslash \{k\})}$, it computes $D = \prod_{\mathcal{N}}^{i=1} x_i.g^{v_i + d_i} = g^{\sum_{i=1}^{\mathcal{N}} v_i + d_i}$ and sends it to user U. It also privately sends $x_k.g^{d_k}$ to the user. For each attribute $att(i) \in \mathbb{A}_u$, each authority AA_k selects a random number $r_i \in_r \mathbb{Z}_p^*$, computes $H(att(i))^{r_i}$ and $D'_i = g^{r_i}$, and sends them to U. Then, user U computes $D_i = H(att(i))^{r_i}. \prod_{k=1}^{\mathcal{N}} x_k.g^{d_k} = H(att(i))^{r_i}.g^{\sum_{k=1}^{\mathcal{N}} d_k}$ and constructs his own secret key $SK_U = \{D = g^{\sum_{i=1}^{\mathcal{N}} v_i + d_i}, \forall i \in \mathbb{A}_u : (D_i = H(att(i))^{r_i}.g^{\sum_{k=1}^{\mathcal{N}} d_k}, D'_i = g^{r_i})\}$.

Encryption$(PK, \{\mathcal{T}_l\}_{l \in \{0,\dots,r-1\}}, M)$: The encryption algorithm, which is run by *DO*, takes as input public parameters PK, a set of privilege trees $\{\mathcal{T}_l\}_{l \in \{0,\dots,r-1\}}$, and plaintext M. It is assumed that user U can execute specific operation on the ciphertext if and only if his attributes satisfy the corresponding privilege tree $\mathcal{T}_l$. More specifically, $\mathcal{T}_0$ stands for the privilege to read M. In this algorithm, *DO* encrypts M using a random symmetric key K_e and an existing symmetric encryption scheme. For each $\mathcal{T}_l$, *DO* selects random number s_l in $\mathbb{Z}_p^*$ and shares it in privilege tree $\mathcal{T}_l$ with root R_l (In a top-down manner, similar to the way which has done in [29]). It also selects random number $h \in \mathbb{Z}_p^*$ and calculates $g^{h.s_l}$ and $D^{h^{-1}}$. Then, it calculates ciphertext $CT = \left(\{\mathcal{T}_l\}_{l \in \{0,\dots,r-1\}}, E_0 = K_e.Y^{s_0}, C = g^{hs_l}, \widehat{C} = D^{h^{-1}}, \{C_i = g^{q_i(0)}, C'_i = H(att(i))^{q_i(0)}\}_{\forall i \in \mathbb{A}^{\mathcal{T}_l} \forall l \in \{1,\dots,r-1\}} \right)$, where $\mathbb{A}^{\mathcal{T}_l}$ is the set of attributes included in $\mathcal{T}_l$. At the end, *DO* sends CT as well as secret $VR = \left(\{E_l = Y^{s_l}\}_{l \in \{1,\dots,r-1\}}\right)$ to cloud service provider *CSP*.

Decryption(PK, SK_U, CT): The decryption algorithm, which is run by U, takes as input public parameters PK, user's secret key SK_U and ciphertext CT. To have access to plaintext M, U computes $DecNode(CT, SK_U, x) = \frac{e(D_j, C_y)}{e(D'_j, C'_y)} = e(g,g)^{(\sum_{i=1}^{\mathcal{N}} d_i.q_y(0))/b}$ for each leaf node y in the privilege tree $\mathcal{T}_p$ and corresponding attribute $att(j) \in \mathbb{A}_u$. Then, it recursively computes A as $DecNode(R)$ for root node R of $\mathcal{T}_p$ as $DecNode(CT, SK_U, R)$. If U is eligible to access M (i.e. the privilege tree $\mathcal{T}_p$ is satisfied by $\mathbb{A}_u$), then A will be equal to $e(g,g)^{\sum_{i=1}^{\mathcal{N}} d_i.q_R(0)} = e(g,g)^{\sum_{i=1}^{\mathcal{N}} d_i.s_p}$. If U wants to read M, the decryption key K_e can be recovered by $K_e = \frac{E_0.A}{e(C,\widehat{C})} = \frac{\left(K_e.Y^{s_0}\right).\left(e(g,g)^{\sum_{i=1}^{\mathcal{N}} d_i.s_0}\right)}{e(g,g)^{\sum_{i=1}^{\mathcal{N}} (v_i + d_i)s_0}}$. For any other operation, U decrypts the corresponding privilege tree $\mathcal{T}_j$, recovers Y^{s_j} and sends it to the cloud server. The cloud server checks whether $Y^{s_j} = E_j$ and proceeds if the response is yes.

By analyzing the underlying algorithms in [20], we found that the scheme suffers from user authorization attacks and coarse-grained access control. In the following, we explain our findings.

User Authorization Attacks: Assume that malicious user U has been authorized and received $D = g^{\Sigma v_k + d_k}$ and $x_k.g^{d_k}$ to generate his own secret key $SK_u = \{D = g^{\Sigma v_k + d_k}, \forall i \in \mathbb{A}_u : (D_i = H(att(i))^{r_i}.g^{\Sigma d_k}, D'_i = g^{r_i})\}$ in *Key Generation* phase. He can calculate $\hat{D} = \prod x_k.g^{d_k}$, select random number $r_{i'}$ and forge secret key for any attribute $att(i')$ (i.e. $D_{i'} = H(att(i'))^{r_{i'}}.g^{\Sigma d_k}, D'_i = g^{r_{i'}}$) and thus be authorized to access any file he wants.

Jung et al. also introduce *AnonyControl-F* approach, the second approach presented in [20], to achieve full anonymity. Since in the former approach, described above, in *KeyGeneration* algorithm, each attribute authority AA_K is responsible for computing and sending $H(att(i))^{r_i}$ to user U, it has access to his identity information which can be used to authorize him, when he requests to have access to a ciphertext. To achieve full anonymity and make their scheme immune against this leakage, the authors assumed that each attribute belongs to a category containing k possible attribute values. All the attribute values belonging to the same category are managed by the same authority and each user has the right to select at most one attribute value in one category. Upon receiving the key request, the attribute authority picks a random number r_u for the user and generates $\mathrm{H}(att(i))^{r_u}$ for all $i \in \{1, \dots, k\}$. Using *1-Out-of-k Oblivious Transfer* crypto primitive (See Algorithm 2) user U can request only one attribute value (and not more) from one set he wants to be issued by the authority.

User Authorization Attacks: the authors assumed k possible attribute values for each attribute requested by user U. Then, using *1-Out-of-k Oblivious Transfer* crypto primitive, U asks for just one attribute value he wants to be issued by the authority. In this approach, a malicious user may request to have any other attribute value from the same set, which may not necessarily belong to him. In this way, a non-eligible user may be authorized by an authority responsible for issuing attributes anonymously.

Coarse-Grained Access Control: In key generation phase, attribute authorities collaborate to issue a secret parameter D as part of secret

Algorithm 1 1-out-of-2 Oblivious Transfer

```
1: Bob randomly picks a secret $s$ and publishes $g^s$ to Alice.
2: Alice creates an encryption/decryption key pair:$\{g^r, r\}$.
3: Alice chooses i and calculates $EK_i = g^r$ , $EK_{i-1} = \frac{g^s}{g^r}$ and sends $EK_0$ to Bob.
4: Bob calculates $EK_1 = \frac{g^s}{EK_0}$ and encrypts $M_0$ using $EK_0$ and $M_1$ using $EK_1$ and sends
   two ciphertexts $E_{EK_0}(M_0)$ and $E_{EK_1}(M_1)$ to Alice.
5: Alice can use $r$ to decrypt the desired ciphertext $E_{EK_i}(M_i)$, but she cannot decrypt the
   other one. Meanwhile, Bob does not know which ciphertext is decrypted.
```

Algorithm 2 1-out-of-k Oblivious Transfer

```
1: Bob randomly picks $k$ secrets $s_1, \dots, s_k$ and calculates $t_i$ as follows:
       $\forall i \in \{1, \dots, n\} : t_i = s_1 \oplus \dots \oplus s_{i-1} \oplus M_i$
2: For each $i \in \{1, \dots, k\}$, Bob and Alice are engaged in a 1-out-of-2 Oblivious Transfer (See
   Algorithm 1), where Bob's first message is $t_i$ and the second message is $s_i$. Alice picks
   $t_i$, to receive, if she wants $M_i$ and $s_i$ otherwise.
3: After Alice receives $n$ components, she has $t_i = s_1 \oplus \dots \oplus s_{i-1} \oplus M_i$ for the $i$ she wants
   and $s_j$ for $j < i$. Thus, she can recover $M_i$ by $M_i = ti \oplus s_{i-1} \oplus s_{i-2} \oplus \dots \oplus s_1$.
```

key for each user. In encryption phase, D is used to compute $\hat{C} = D^{h^{-1}}$. This means that data owner should use different Ds for different users to encrypt files restricting the "fine-grained" property of the scheme.

In [21], Jung et al. extended their contribution in [28] and made it immune against the leakage of master secret key [34]. Similar to [20], the scheme presented in [21] consists of four algorithms *Setup*, *Key-Generation*, *Encryption*, and *Decryption*, in which *Setup*, *Encryption*, and *Decryption* are similar to the corresponding ones in [20]. *KeyGeneration* algorithm is re-defined as follows:

KeyGeneration$(PK, MK_{a_k(1\le k\le \mathcal{N})}, \mathbb{A}_u)$: In this algorithm, each authority AA_k selects a random parameter $d_k \in \mathbb{Z}_p^*$, computes $x_k.g^{(v_k+d_k)}$ and shares it with other authorities. Receiving $\mathcal{N}-1$ pieces $x_j.g^{(v_j+d_j)}$ generated by other authorities $AA_{j(j\in\{1,\dots,\mathcal{N}\}\setminus\{k\})}$, it computes $D = \prod_{\mathcal{N}}^{i=1} x_i.g^{v_i+d_i} = g^{\sum_{i=1}^{\mathcal{N}} v_i+d_i}$ and sends it to user U. It also privately sends $x_k.g^{d_k}$ to U. For each attribute $att(i) \in \mathbb{A}_u$, AA_k is responsible for, it selects a random number $r_i \in_r \mathbb{Z}_p^*$, computes $\widehat{D}_i = H(att(i))^{r_i}.x_k.g^{d_k}$ and $D'_i = g^{r_i}$, and sends them to U. Then, U computes $D_i = \widehat{D}_i.\prod_{k=1,k\neq i}^{\mathcal{N}} x_k.g^{d_k} = H(att(i))^{r_i}.g^{\sum_{k=1}^{\mathcal{N}} d_k}$ and constructs his own secret key as $SK_U = \{D = g^{\sum_{i=1}^{\mathcal{N}} v_i+d_i}, \forall i \in \mathbb{A}_u : (D_i = H(att(i))^{r_i}.g^{\sum_{k=1}^{\mathcal{N}} d_k}, D'_i = g^{r_i})\}$.

By analyzing the underlying algorithms, we found that the scheme [21] suffers from the following shortcomings:

User and Authority Collusion Attacks: To generate secret key for each user, attribute authority AA_k, for each attribute $att(i) \in \mathbb{A}_u$ it is responsible for, selects random number $r_i \in_r \mathbb{Z}_q^*$, computes $\hat{D}_i = H(att(i))^{r_i}.x_k.g^{d_k}$ and $D'_i = g^{r_i}$, and sends them to that user. Let us assume that attribute $att(j)$ is issued by malicious authority $AA_{iss(j)}$. Moreover, let us assume that malicious user U is authorized to receive secret key $SK_u = \{D = g^{\Sigma v_k+d_k}, \forall i \in \mathbb{A}_u : D_i = H(att(i))^{r_i}.g^{\Sigma d_k}, D'_i = g^{r_i}\}$, where $j \in \mathbb{A}_u$. In collusion between malicious user U and malicious authority $AA_{iss(j)}$, instead of $\hat{D}_j = H(att(j))^{r_j}.x_{iss(j)}.g^{d_{iss(j)}}$, $AA_{iss(j)}$ can send $x_{iss(j)}.g^{d_{iss(j)}}$ to U. In this case, U is able to compute $\prod_{i=1}^{\mathcal{K}} x_i.g^{d_i} = g^{\Sigma d_i}$, and forge any attribute he wants.

Coarse-Grained Access Control: Similarly to [20,28], the scheme in [21] suffers from coarse-grained access control.

6. System and security models

In this section, first, we present the system model and its architecture. Second, we describe the threat model and security assumptions about the entities in that architecture. Third, we present the assumptions used to prove the security of the proposed schemes. Next, we describe the framework of *PACS* and its functional model. Finally, we define the security models used for security analysis in Section 8.

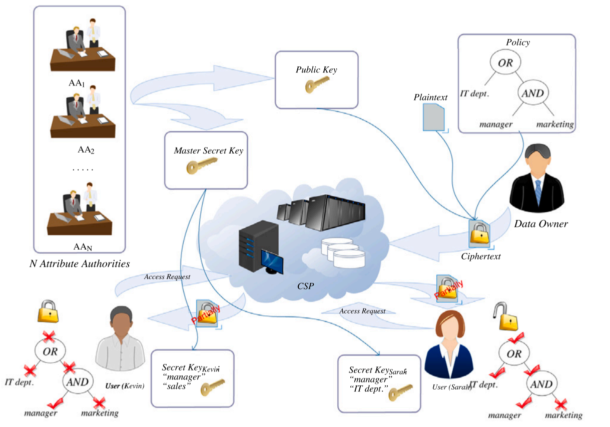


Fig. 2. Entities in *PACS*.

6.1. System model

In the architecture considered for *PACS*, there are four entities (see Fig. 2): $\mathcal{N}$ *Attribute Authorities* (AA_i), *User* (U), *Cloud Service Provider* (*CSP*), and *Data Owner* (*DO*). Attribute authorities $AA_{i(1\le i\le \mathcal{N})}$ play a central role, issuing public parameters and generating parts of user secret keys. Each authority contributes a key component based on the attributes it manages. A user's complete secret key is formed by combining these keys from all relevant authorities. Users U trigger access requests by transmitting their attributes to the Cloud Service Provider (*CSP*). The Data Owner (*DO*) defines access policies, encrypts data accordingly, and delegates storage to the cloud. Upon receipt of a user's access query, *CSP* partially decrypts the data based on the embedded attributes and associated access policies, utilizing CSP-assisted decryption for improved efficiency and scalability. If user U is authorized, they can decrypt the received data using their private keys, ensuring robust data confidentiality and access control. This architectural model not only safeguards data comprehensively but also optimizes performance and scalability in cloud environments, thereby addressing research objectives focused on enhancing privacy, security, and efficiency in cloud-based data access control. Additionally, distributing key generation among multiple authorities mitigates single points of failure and prevents any single authority from possessing the entire key, making it significantly harder for malicious authorities to collude and compromise user credentials.

6.2. Threat model

Each attribute authority AA_i is in charge of a subset of the users' attributes and for each attribute, it is responsible for, it knows the exact information of the key requester (i.e. the user who wants to access the data). It is assumed that AA_i is untrusted in the sense that it will follow the scheme, but, it may try to collude with users or other authorities and forge a new key without the involvement of some authorities. U might be malicious, collude with other users or authorities and forge his attributes to escalate his rights and get access to services and information he is not eligible to access. In our architecture, we assume that *CSP* is honest but curious. That means that *CSP* will faithfully follow the proposed scheme, but can launch passive attacks to get as much secret information as possible. Hence, the data stored in the cloud should remain encrypted all the time and any required transformation should not reveal the plaintext in the process. We assume that the policy, by which the data is encrypted, is publicly known and verifiable by everyone. We also assume that the communication channels are secure and packets are untraceable when queries and information are transmitted on these channels. This assumption can be realized using *Secure Socket Layer* (*SSL*) or some other techniques [35].

6.3. Security assumptions

Assumption 1 (*Computational Diffie–Hellman (CDH) Assumption*)**.** Given $(g,\ g^a,\ g^b) \in \mathbb{G}^3$, where a and b are two random numbers in $\mathbb{Z}_p^*$, there is no *Probabilistic Polynomial Time (PPT)* algorithm $\mathcal{A}$ with non-negligible advantage ε to compute g^{ab}.

Assumption 2 (*Decisional Bilinear Diffie–Hellman (DBDH) Assumption*)**.** Given $(g,\ g^a,\ g^b,\ g^c) \in \mathbb{G}^4$ and $T \in \mathbb{G}_T$, where a, b, c are three random numbers in $\mathbb{Z}_p^*$, there is no *PPT* algorithm $\mathcal{A}$ with non-negligible advantage ε to determine whether T is equal to $e(g,g)^{abc}$ or is a random number.

6.4. Framework

The framework of *PACS* is composed of the following six algorithms:

Setup$(\mathbb{A}, \Lambda) \longrightarrow (PK, MK_{a_{i(1\le i\le \mathcal{N})}})$: The setup algorithm, which is run by attribute authorities, takes as input the attribute universe $\mathbb{A}$ and an implicit security parameter Λ. It outputs public parameters PK as well as master secret key MK_{a_i} for each attribute authority AA_i.

KeyGeneration$(PK, MK_{a_{i(1\le i\le \mathcal{N})}}, \mathbb{A}_u) \longrightarrow (SK_U)$: The key generation algorithm, which is run by attributes authorities, takes as input public parameters PK, master secret keys $MK_{a_{i(1\le i\le \mathcal{N})}}$ and user's set of attributes $\mathbb{A}_u$. It generates secret key SK_U corresponding to the input attribute set $\mathbb{A}_u$ and returns it as output.

KeyGenOut$(PK, SK_U, b) \longrightarrow (SK_U^{1/b})$: This algorithm, which is run by *U*, takes as input public parameters PK, user's secret key SK_U and a secret retrieving key b known only to *U*. It returns as output a blinded secret key $SK_U^{1/b}$, which will be sent to *CSP*.

Encryption$(PK, \mathcal{T}, M) \longrightarrow CT$: The encryption algorithm, which is run by *DO*, takes as input public parameters PK, tree access structure $\mathcal{T}$, that determines access policy, and message M; it generates ciphertext CT as output.

DecryptionOut$(PK, SK_U^{1/b}, \mathbb{A}_u, CT) \longrightarrow CT'$: The algorithm, which is run by *CSP*, takes as input public parameters PK, blinded secret key $SK_U^{1/b}$ and ciphertext CT. If attributes $\mathbb{A}_u$, associated with *U*, satisfies the policy associated with CT, it computes and returns transformed ciphertext CT'.

Decryption$(PK, CT', b) \longrightarrow M$: The decryption algorithm, which is run by *U*, takes as input public parameters PK, transformed ciphertext CT' and secret retrieving key b, and returns message M as output.

6.5. Security models

In this section, we define security models for immunity of *PACS* against chosen plaintext attacks, users collusion attacks and authorities collusion attacks. We also provide security definition for anonymizing user's identity in the proposed scheme.

6.5.1. Chosen plaintext attacks

The selective security and indistinguishability against *Chosen Plaintext Attacks* (*IND-CPA*) for *PACS* is defined by the following game between challenger $\mathcal{C}$ and adversary $\mathcal{A}$:

Initialization. $\mathcal{A}$ declares the set of at most $\mathcal{N}-1$ compromised authorities that are under its control. The remaining authorities are controlled by $\mathcal{C}$. $\mathcal{A}$ also commits tree access structure $\mathcal{T}$ for the game.

Setup. $\mathcal{C}$ and $\mathcal{A}$ jointly run *Setup* algorithm to obtain the valid parameters.

Learning 1. $\mathcal{A}$ is able to query for an arbitrary number (i.e. p) of secret keys corresponding to chosen attribute sets $\mathbb{A}_{u_1}, \dots, \mathbb{A}_{u_p}$ for selected number of users $\{U_1, \dots, U_p\}$. These attribute sets are disjointly issued by all authorities, but none of them satisfies tree access structure $\mathcal{T}$. $\mathcal{A}$ is also allowed to query for an arbitrary number (i.e. $q-p$) of blinded secret keys corresponding to attribute sets $\mathbb{A}_{u_{p+1}}, \dots, \mathbb{A}_{u_q}$ disjointly issued by all authorities for selected number of users $\{U_{p+1}, \dots, U_q\}$. Furthermore, $\mathcal{A}$ can conduct an arbitrary number of computations, using its own (or compromised) secret (or blinded secret) keys, and all public parameters.

Challenge. $\mathcal{A}$ sends two distinct chosen plaintexts M_0 and M_1 to $\mathcal{C}$. $\mathcal{C}$ randomly selects bit $\varpi \in \{0,1\}$, encrypts M_ϖ with respect to tree access structure $\mathcal{T}$ and returns the result to $\mathcal{A}$.

Learning 2. $\mathcal{A}$ continues to repeat learning 1 adaptively.

Response. $\mathcal{A}$ outputs guess ϖ' of ϖ.

Definition 11. *PACS* is selective secure and indistinguishable against chosen plaintext attacks (*IND-CPA*), if all probabilistic polynomial time adversaries have only a negligible advantage in the above game. An adversary is said to have the advantage ε, if it wins the game with probability $Pr[\varpi' = \varpi] = \left(\frac{1}{2} + \varepsilon\right)$

6.5.2. Users collusion attacks

Malicious users may try to collude, combine their secret keys and obtain a new secret key to increase their privileges [36]. The security of *PACS* against *Users Collusion Attacks (UCA)* can be defined by the following game:

Initialization. All the authorities are controlled by $\mathcal{C}$. $\mathcal{A}$ commits tree access structure $\mathcal{T}$ for the game.

Setup. $\mathcal{C}$ and $\mathcal{A}$ jointly run *Setup* algorithm to obtain the valid parameters.

Learning 1. $\mathcal{A}$ is able to query for two secret keys SK_U and $SK_{U'}$, corresponding to two attribute sets $\mathbb{A}_u$ and $\mathbb{A}_{u'}$ belonging to two users U and U'. These attribute sets do not satisfy $\mathcal{T}$ individually, whereas their union does. More specifically, there is at least one attribute $att(k)$, issued by an uncompromised authority, in $\mathbb{A}_u$ but not in $\mathbb{A}_{u'}$. $\mathcal{A}$ is also allowed to conduct an arbitrary number of computations, using its own (or compromised) secret keys, and all public parameters.

Challenge. $\mathcal{A}$ sends two distinct chosen plaintexts M_0 and M_1 to $\mathcal{C}$. $\mathcal{C}$ randomly selects bit $\varpi \in \{0,1\}$, encrypts M_ϖ with respect to $\mathcal{T}$ and returns the result to $\mathcal{A}$.

Learning 2. $\mathcal{A}$ continues to repeat its computations adaptively, using its own (or compromised) secret keys, and all public parameters, similar to the ones in learning 1.

Response. $\mathcal{A}$ outputs guess ϖ' of ϖ.

Definition 12. *PACS* is secure against *Users Collusion Attacks (UCA)*, if all probabilistic polynomial time adversaries have only a negligible advantage in the above game.

6.5.3. Authorities collusion attacks

Malicious authorities may try to collude, combine their master secret parameters and generate valid master secret parameters for an uncompromised authority responsible for issuing part of user's secret key. To describe the security of *PACS* against *Authorities Collusion Attacks (ACA)*, we define the following game:

Initialization. Adversary $\mathcal{A}$ declares the set of compromised authorities that are under its control. The remaining authorities are controlled by challenger $\mathcal{C}$. $\mathcal{A}$ also commits tree access structure $\mathcal{T}$, which includes at least one attribute issued by an uncompromised authority.

Setup. $\mathcal{C}$ and $\mathcal{A}$ jointly run *Setup* algorithm to obtain the valid parameters.

Learning 1. $\mathcal{A}$ is able to query with the authorities under its control for an arbitrary number (i.e. p) of secret keys corresponding to chosen attribute sets $\mathbb{A}_1, \ldots, \mathbb{A}_p$ for selected number of users. $\mathcal{A}$ also can conduct an arbitrary number of computations, using its own secret keys (or its own blinded secret keys, which can be computed from its own secret keys by itself), and all public parameters.

Challenge. $\mathcal{A}$ sends two distinct chosen plaintexts M_0 and M_1 to $\mathcal{C}$. $\mathcal{C}$ randomly selects bit $\varpi \in \{0,1\}$, encrypts M_ϖ with respect to $\mathcal{T}$ (which includes at least one attribute issued by an uncompromised authority) and returns the result to $\mathcal{A}$.

Learning 2. $\mathcal{A}$ continues to repeat learning 1 adaptively.

Response. $\mathcal{A}$ outputs guess ϖ' of ϖ.

Definition 13. *PACS* is secure against *Authorities Collusion Attacks (ACA)*, if all probabilistic polynomial time adversaries have only a negligible advantage in the above game.

6.5.4. User's identity anonymization

Assume that the identity of user U is revealed to an attribute authority, when it issues a part of his secret key. To describe anonymity of U against honest but curious *CSP*, while requesting to have access to a ciphertext, we define the following game between a challenger and an adversary. The game is constructed to build an identifying blackbox that can extract identity of user whose blinded secret key has been used to decrypt ciphertext.

Initialization. adversary $\mathcal{A}$ declares the set of at most $\mathcal{N}-1$ compromised authorities that are under its control. The remaining authorities are controlled by challenger $\mathcal{C}$. $\mathcal{A}$ also commits tree access structure $\mathcal{T}$ for the game.

Setup. $\mathcal{C}$ and $\mathcal{A}$ jointly run *Setup* algorithm to obtain the valid parameters.

Learning 1. $\mathcal{A}$ is able to query for an arbitrary number (i.e. p) of secret keys corresponding to chosen attribute sets $\mathbb{A}_{u_1}, \ldots, \mathbb{A}_{u_p}$ for selected number of users $\{U_1, \ldots, U_p\}$. These attribute sets are disjointly issued by all authorities, but none of them satisfies tree access structure $\mathcal{T}$. $\mathcal{A}$ is also allowed to query for an arbitrary number (i.e. $q-p$) of blinded secret keys corresponding to attribute sets $\mathbb{A}_{u_{p+1}}, \ldots, \mathbb{A}_{u_q}$ disjointly issued by all authorities for selected number of users $\{U_{p+1}, \ldots, U_q\}$. Furthermore, $\mathcal{A}$ can conduct an arbitrary number of computations, using its own (or compromised) secret (or blinded secret) keys, and all public parameters.

Challenge $\mathcal{A}$ sends (all or identical parts of) secret keys SK_{U_0} and SK_{U_1} of two users U_0 and U_1 to $\mathcal{C}$. $\mathcal{C}$ randomly selects bit $\varpi \in \{0,1\}$, computes corresponding components of blinded secret key $SK_{U_\varpi}^{1/b}$ and returns them to $\mathcal{A}$.

Learning 2. $\mathcal{A}$ continues to repeat learning 1 adaptively.

Response. $\mathcal{A}$ outputs guess ϖ' of ϖ.

Definition 14. *PACS* anonymizes U's identity against *CSP*, if all probabilistic polynomial time adversaries have only a negligible advantage in the above game.

7. Construction of Privacy-Preserving Access Control Scheme (PACS)

In this section, we describe the construction of the proposed scheme. *PACS* contains six algorithms which are performed in four phases: setup, key generation, encryption, and decryption.

7.1. Setup phase

In the setup phase, attribute authorities $AA_{i(1\le i\le\mathcal{N})}$, perform *Setup* algorithm to generate public parameters as well as master secret keys. In this phase, each attribute authority AA_k

- Agrees with other authorities on a large prime number p, where the size of p is dictated by the security parameter Λ, a generator g for the bilinear group $\mathbb{G}$ of order p, and a public hash function $H : \{0,1\}^* \to \mathbb{G}$ which maps each binary attribute string into a group element in $\mathbb{G}$. This setup can be processed by one authority and shared with the others.
- Chooses random secret parameter $v_k \in \mathbb{Z}_p^*$ and computes $e(g,g)^{v_k}$ and shares it with all other authorities.
- Selects random integer $s_{kj} \in \mathbb{Z}_p^*(j \in \{1,\ldots,\mathcal{N}\} \setminus \{k\})$, computes $g^{s_{kj}}$ and shares it with authority AA_j $(\forall j \in \{1,\ldots,\mathcal{N}\} \setminus \{k\})$.
- Receives $\mathcal{N}-1$ pieces of $g^{s_{jk}}$ and $e(g,g)^{v_j}$ generated by other authorities $AA_j(j \in \{1,\ldots,\mathcal{N}\} \setminus \{k\})$. It selects random exponent $\tau_k \in \mathbb{Z}_p^*$, computes $\vartheta_k = g^{\tau_k}$ and $\zeta = \prod_{i\in\{1,\ldots,\mathcal{N}\}} e(g,g)^{v_i} = e(g,g)^{\sum_{i=1}^{\mathcal{N}} v_i}$ and publishes ϑ_k and ζ as authorities' public parameters. Aggregating authorities' public parameters constructs public parameters $PK = (\mathbb{G}, g, \zeta = e(g,g)^{\sum_{i=1}^{\mathcal{N}} v_i}, \vartheta_i = g^{\tau_i}(1 \le i \le \mathcal{N}), H(.))$.
- Computes authority secret key $x_k \in \mathbb{Z}_p^*$ as follows:

$$x_k = \Big(\prod_{j\in\{1,\ldots,\mathcal{N}\}\setminus\{k\}} g^{s_{kj}}\Big) / \Big(\prod_{j\in\{1,\ldots,\mathcal{N}\}\setminus\{k\}} g^{s_{jk}}\Big)$$
$$= g^{\left(\sum_{j\in\{1,\ldots,\mathcal{N}\}\setminus\{k\}} g^{s_{kj}} - \sum_{j\in\{1,\ldots,\mathcal{N}\}\setminus\{k\}} g^{s_{jk}}\right)}$$

  Then, the master secret key for attribute authority AA_k would be $MK_{a_k} = (v_k, \tau_k, x_k)$. It can be observed that randomly produced integers x_k, where $k \in \{1,\ldots,\mathcal{N}\}$, satisfy the condition $\prod_{k\in\{1,\ldots,\mathcal{N}\}} x_k = 1 \ mod \ p$.

7.2. Key generation phase

In the key generation phase, attribute authorities $AA_{i(1\le i\le\mathcal{N})}$ collaborate to perform *KeyGeneration* algorithm and issue secret key for each user. Then, a typical user U performs *KeyGenOut* to anonymize his secret key and sends his access request to *CSP*. It is worth noting that for *PACS*, similar to the traditional *ABE* approach, we only consider attributes (i.e., user either possesses an attribute or not) and not the values of these attributes. To perform *KeyGeneration*, each attribute authority AA_k

- Selects random number $d_k \in \mathbb{Z}_p^*$, computes $x_k.g^{d_k}$ and $x_k.g^{(v_k+d_k)}$, and shares them with all other authorities $AA_i(i \in \{1,\ldots,\mathcal{N}\} \setminus \{k\})$.
- Receives $\mathcal{N}-1$ pieces of $x_k.g^{d_k}$ and $x_i.g^{(v_i+d_i)}$ generated by $AA_i(i \in \{1,\ldots,\mathcal{N}\} \setminus \{k\})$.
- Computes $D_{\Sigma d_i} = \prod_{\mathcal{N}}^{i=1} x_i.g^{d_i} = g^{\sum_{i=1}^{\mathcal{N}} d_i}$ and $D = \prod_{i=1}^{\mathcal{N}} x_i.g^{(v_i+d_i)} = g^{\sum_{i=1}^{\mathcal{N}}(v_i+d_i)}$, and sends D to user U.
- Chooses random number $r_j \in_r \mathbb{Z}_q^*$ for each attribute $att(j) \in \mathbb{A}_u$, computes $D_j = D_{\Sigma d_i}^{(1/\tau_k)}(H(att(j)))^{r_j} = g^{(\sum_{i=1}^{\mathcal{N}} d_i/\tau_k)}(H(att(j)))^{r_j}$ and $D'_j = g^{r_j}$ and sends them to user U.

To perform $KeyGenOut$, user U

- Chooses random number $b \in \mathbb{Z}_p^*$ $(b > 1)$ as secret retrieving key RK_U and transforms secret key $SK_U = \left(D,\ \forall j \in \mathbb{A}_u : (D_j, D'_j)\right)$ to its blinded version. This is performed by raising its components to the power of $1/b$, denoted by $SK_U^{1/b} = \left(D^{1/b},\ \forall j \in \mathbb{A}_u : (D_j^{1/b}, D_j'^{1/b})\right)$.

7.3. Encryption phase

In the encryption phase, data owner DO performs $Encryption$ algorithm to define access policy and encrypt data based on that policy. In this process, DO

- Defines access control policy for all attributes.
- Selects random number s in $\mathbb{Z}_p^*$ and computes $E = M.e(g,g)^{\sum_{i=1}^{\mathcal{N}} v_i s}$.
- Shares secret s in tree access structure $\mathcal{T}$ with root R. In a top-down approach, it chooses polynomial q_x for each node x in tree access structure $\mathcal{T}$ as follows: For each node x, the algorithm sets degree d_x of polynomial q_x to $k_x - 1$ where k_x is the threshold value of node x. Starting from root node R, the algorithm sets $q_R(0) = s$ and randomly chooses d_R other coefficients to define q_R completely. For any other node x, it sets $q_x(0) = q_{parent(x)}(index(x))$[1] and chooses d_x other coefficients randomly to completely define q_x. Note that each leaf node y is associated with atomic attribute $att(y)$ in the set of attributes.
- Uploads ciphertext $CT = \Big(E = M.e(g,g)^{\sum_{i=1}^{\mathcal{N}} v_i s}, C = g^s, \forall y \in \mathbb{A}_{\mathcal{T}} : (C_y = \vartheta_{iss(y)}^{q_y(0)} = g^{(\tau_{iss(y)}\ q_y(0))}$,[2] $C'_y = H(att(y))^{q_y(0)}) \Big)$ to CSP.

In order to adopt the key encapsulation mechanism and reduce the size of ciphertext without sacrificing security [37], instead of computing $M.e(g,g)^{\sum_{i=1}^{\mathcal{N}} v_i s}$ for plaintext M, DO selects random content key CK and computes $CK.e(g,g)^{\sum_{i=1}^{\mathcal{N}} v_i s}$. It uses the content key CK to encrypt the target file (i.e., the file we want to encrypt and for which we define access control) with symmetric encryption $ENC_{CK}(.)$.

7.4. Decryption phase

During the decryption phase, user U initiates the process by sending an access request to the CSP. This request includes the user's blinded secret key $SK_U^{1/b} = (D^{1/b}, \forall j \in \mathbb{A}_u : (D_j^{1/b}, D_j'^{1/b}))$. Upon receiving the access request, CSP performs $DecryptionOut$ algorithm to check the eligibility of the user, outsource the computation cost and partially decrypt the ciphertext. Then, U performs $Decryption$ algorithm to have access to the corresponding plaintext.

To perform $DecryptionOut$, CSP

- Computes $DecNode(y) = \frac{e(D_j^{1/b}, C_y)}{e(D_j'^{1/b}, C'_y)} = e(g,g)^{(\sum_{i=1}^{\mathcal{N}} d_i).q_y(0)/b}$ for each leaf node y in tree access structure $\mathcal{T}$ and corresponding attribute $att(j) \in \mathbb{A}_u$.
- Recursively, computes A as $DecNode(R)$ for the root node R of $\mathcal{T}$ as follows: For each non-leaf node x, the algorithm calls $DecNode(z)$ for all child nodes z of x and stores the output as F_z. Let S_x be an arbitrary k_x-sized set of child nodes z. The computation of $DecNode(x)$ is as follows:

$$\begin{aligned} F_x &= \Pi_{z \in S_x} F_z^{\Delta_{\iota, S'_x}(0)}, \ where \ \begin{cases} \iota &= index(z) \\ S'_x &= \{\iota : z \in S_x\} \end{cases} \\ &= \Pi_{z \in S_x} \left(e(g,g)^{(\sum_{i=1}^{\mathcal{N}} d_i).q_z(0)/b} \right)^{\Delta_{\iota, S'_x}(0)} \\ &= \Pi_{z \in S_x} \left(e(g,g)^{(\sum_{i=1}^{\mathcal{N}} d_i).q_z(0)} \right)^{\Delta_{\iota, S'_x}(0)/b} \\ &= \Pi_{z \in S_x} \left(e(g,g)^{(\sum_{i=1}^{\mathcal{N}} d_i).q_{parent(z)}(\iota)} \right)^{\Delta_{\iota, S'_x}(0)/b} \ (via\ construction) \\ &= \Pi_{z \in S_x} \ e(g,g)^{(\sum_{i=1}^{\mathcal{N}} d_i).q_x(\iota).\Delta_{\iota, S'_x}(0)/b} \\ &= e(g,g)^{(\sum_{i=1}^{\mathcal{N}} d_i).q_x(0)/b} (using\ polynomial\ interpolation) \end{aligned}$$

where $\Delta_{\iota, S'_x}(0)$ is the Lagrange coefficient, which is defined as $\Delta_{\iota, S'_x}(x) = \Pi_{j \in S'_x, j \neq \iota} \frac{x-j}{\iota-j}$ [29]. The starts by calling the function $DecNode(.)$ for the root node R of the tree access structure $\mathcal{T}$. If the tree access structure is satisfied by $\mathbb{A}_u$, then A, which is set to $DecNode(R)$, becomes equal to $e(g,g)^{\sum_{i=1}^{\mathcal{N}} d_i.q_R(0)/b} = e(g,g)^{\sum_{i=1}^{\mathcal{N}} d_i.s/b}$.

- Computes the expression $\frac{e(C, D^{1/b})}{A} = \frac{e(g,g)^{\sum_{i=1}^{\mathcal{N}} (v_i + d_i)s/b}}{e(g,g)^{\sum_{i=1}^{\mathcal{N}} d_i.s/b}} = e(g,g)^{\sum_{i=1}^{\mathcal{N}} v_i s/b}$. Simplifying this expression results in the blinded plaintext $M^{1/b}$, since $e(g,g)^{\sum_{i=1}^{\mathcal{N}} v_i s/b} = \left(e(g,g)^{\sum_{i=1}^{\mathcal{N}} v_i s} \right)^{1/b} = M^{1/b}$. It is important to note that in this calculation, we do not need to compute the actual value of M. Instead, we directly obtain $M^{1/b}$. This approach not only achieves the desired result but also protects the value of M from disclosure to the CSP. Finally, the CSP sends $M^{1/b}$ to the user.

To perform $Decryption$, user U

- Computes plaintext M by raising received $M^{1/b}$ to the power b selected in key generation phase.

8. Security discussion

In this section, we analyze the security of the proposed scheme. The analysis includes considering immunity of *PACS* against authorities collusion attacks, users collusion attacks and chosen plaintext attacks. It also discusses how *PACS* anonymizes user's identity.

Theorem 1 (*Security Against Authorities Collusion Attacks*). *PACS is secure against compromised authorities.*

Proof. An adversary, using the help of malicious authorities, may try to compromise, generate valid master secret parameters and obtain a new secret key applicable in decryption process. Let ε be the probability of compromising each authority. Associating different secret random elements $\tau_{k(1 \le k \le \mathcal{N})}$ with different authorities $AA_{k(1 \le k \le \mathcal{N})}$ and constructing separate $g^{(\sum_{i=1}^{\mathcal{N}} d_i/\tau_k)}$ for each authority prevent applying one authority's shares of secrets (i.e. v_k and d_k) to the others. Thus, the probability of compromising authorities and breaking the security of *PACS* (via authorities collusion attacks) is

$$\sum_{i=\mathcal{N}}^{\mathcal{N}} \binom{\mathcal{N}}{i} \varepsilon^i (1-\varepsilon)^{\mathcal{N}-i} = \varepsilon^{\mathcal{N}},$$

which is negligible. □

Malicious users may try to collude, combine their secret keys and obtain a new secret key to increase their privileges. The security of *PACS* against *Users Collusion Attacks (UCA)* can be proved based on the following lemma:

Lemma 1. *Under Computational Diffie–Hellman (CDH) assumption, there is no PPT algorithm $\mathcal{A}$ with non-negligible advantage to derive $g^{\sum_{i=1}^{\mathcal{N}} d_{i,u}/\tau_{iss(k)}}$ or $(H(att(k)))^{r_{k,u}}$ (for any $att(k)$) from user's secret key $SK_U = \left(g^{\sum_{i=1}^{\mathcal{N}} (v_i + d_{i,u})}, \forall j \in \mathbb{A}_u : \ (g^{\sum_{i=1}^{\mathcal{N}} d_{i,u}/\tau_{iss(j)}} (H(att(j)))^{r_{j,u}}, \ g^{r_{j,u}}) \right)$.*

Proof. Let $\mathcal{A}$ be a PPT algorithm that solves the problem of deriving $g^{\sum_{i=1}^{\mathcal{N}} d_{i,u}/\tau_{iss(k)}}$ or $(H(att(k)))^{r_{k,u}}$ for some $att(k)$ from user's secret key $SK_U = \left(g^{\sum_{i=1}^{\mathcal{N}} (v_i + d_{i,u})}, \forall j \in \mathbb{A}_u : \ (g^{\sum_{i=1}^{\mathcal{N}} d_{i,u}/\tau_{iss(j)}} (H(att(j)))^{r_{j,u}}, \ g^{r_{j,u}}) \right)$.

[1] $parent(x)$ is the node x's parent node and $index(x)$ is a number associated with each node x ranging from 1 to $num_{parent(x)}$.

[2] $iss(y)$ is the authority responsible for issuing $att(y)$.

Let $g^{\mathfrak{d}} = g^{\sum_{i=1}^{\mathcal{N}}(v_i+d_{i,u})}$, $g^{\mathfrak{t}_{iss(k)}} = g^{\sum_{i=1}^{\mathcal{N}} d_{i,u}/\tau_{iss(k)}}, \forall j \in \mathbb{A}_u : g^{\mathfrak{h}_j} = H(att(j))$. Thus, the problem is equivalent to extract $(g^{\mathfrak{t}_{iss(k)}}, g^{\mathfrak{h}_k r_{k,u}})$ from $SK_U = \big(g^{\mathfrak{d}}, \forall j \in \mathbb{A}_u : (g^{\mathfrak{t}_{iss(j)}+\mathfrak{h}_j r_{j,u}}, g^{r_{j,u}}) \big)$ for some $att(k)$. Since unknown parameters $\mathfrak{d}$ and $r_{j,u}(\forall j \neq k)$ are not related to the problem and since g and $H(att(k)) = g^{\mathfrak{h}_k}$ are publicly known, the above problem can be reduced to finding $(g^{\mathfrak{t}_{iss(k)}}, g^{\mathfrak{h}_k r_{k,u}})$ from $(g, g^{\mathfrak{h}_k}, g^{\mathfrak{t}_{iss(k)}+\mathfrak{h}_k r_{k,u}}, g^{r_{k,u}})$.

Let us assume that there exists *PPT* algorithm $\mathcal{A}'$ with non-negligible advantage to extract $(g^{\mathfrak{t}_{iss(k)}}, g^{\mathfrak{h}_k r_{k,u}})$ from $(g, g^{\mathfrak{h}_k}, g^{\mathfrak{t}_{iss(k)}+\mathfrak{h}_k r_{k,u}}, g^{r_{k,u}})$. We show in the following that we can construct *PPT* algorithm $\mathcal{B}$ with non-negligible advantage to solve *CDH* problem $(g_1, g_1^a, g_1^b) \to g_1^{ab}$.

$\mathcal{B}$ selects random integer c and sends $(g = g_1, g^{\mathfrak{h}_k} = g_1^b, g^{\mathfrak{t}_{iss(k)}+\mathfrak{h}_k r_{k,u}} = g_1^c, g^{r_{k,u}} = g_1^a)$ as input to $\mathcal{A}'$. Upon receiving the corresponding output (R_1, R_2), $\mathcal{B}$ checks whether $R_1.R_2 = g_1^c$ and $e(g_1, R_2) = e(g_1^a, g_1^b)$. If the response is yes, it returns R_2. Otherwise, it selects another random c and invokes $\mathcal{A}$ using the new input $(g_1, g_2^b, g_1^c, g_1^a)$. Since $a = r_{k,u}$, $g^{\mathfrak{h}_k r_{k,u}} = (g_1^b)^a = R_2$, $e(g_1, R_2) = e(g_1, g_1^{ab}) = e(g_1^a, g_1^b)$ and $g_1^c = g^{\mathfrak{t}_{iss(k)}+\mathfrak{h}_k r_{k,u}} = R_1 R_2$, $\mathcal{B}$ returns a valid answer. Thus, we are able to construct *PPT* algorithm $\mathcal{B}$ that can solve the *CDH* problem with non-negligible advantage. □

Theorem 2 (*Security Against Users Collusion Attacks*). *PACS is secure against users collusion attacks under CDH assumption.*

Proof. Assume that malicious users U' and U collude, attempting to combine their secret keys, transfer U 's attributes to U' and obtain more privileges. Based on prior knowledge of

$$\begin{aligned} SK_U &= (D_U, \forall j \in \mathbb{A}_u : (D_{j,U}, D'_{j,U})) \\ &= (g^{\sum_{i=1}^{\mathcal{N}}(v_i+d_{i,u})}, \forall j : (g^{\sum_{i=1}^{\mathcal{N}} d_{i,u}/\tau_{iss(j)}}(H(att(j)))^{r_{j,u}}, g^{r_{j,u}})) \end{aligned}$$

and

$$\begin{aligned} SK_{U'} &= (D_{U'}, \forall j \in \mathbb{A}_{u'} : (D_{j,U'}, D'_{j,U'})) \\ &= (g^{\sum_{i=1}^{\mathcal{N}}(v_i+d_{i,u'})}, \forall j : (g^{\sum_{i=1}^{\mathcal{N}} d_{i,u'}/\tau_{iss(j)}}(H(att(j)))^{r_{j,u}'}, g^{r_{j,u'}})), \end{aligned}$$

the only possible approach for U' to construct a new secret key is to exchange $g^{\sum_{i=1}^{\mathcal{N}} d_{i,u'}/\tau_{iss(j)}}$ or $(H(att(j)))^{r_{j,u'}}$ with $g^{\sum_{i=1}^{\mathcal{N}} d_{i,u}/\tau_{iss(j)}}$ or $(H(att(j)))^{r_{j,u}}$ over some attribute $att(j)$; according to Lemma 1, under *CDH* assumption, user U' cannot extract $g^{\sum_{i=1}^{\mathcal{N}} d_{i,u'}/\tau_{iss(j)}}$ or $(H(att(j)))^{r_{j,u'}}$ components from his secret key and exchange it with the corresponding component of U 's secret key. □

Theorem 3 (*Security Against Chosen Plaintext Attacks*). *Under Decisional Bilinear Diffie–Hellman (DBDH) assumption, PACS is selective secure and indistinguishable against chosen plaintext attacks (IND-CPA).*

Proof. Let $\mathcal{A}$ be a *PPT* algorithm that can break the security of *PACS* in selective secure *CPA* model. Let $(G, G^a, G^b, G^c) \to e(G,G)^{abc}$ be a *DBDH* problem. We show that if adversary $\mathcal{A}$ has non-negligible advantage ε in *IND-CPA* game (see Section 6.5), then, we can construct *PPT* algorithm $\mathcal{B}$ to solve *DBDH* problem with non-negligible advantage $\frac{\varepsilon}{2}$ as follows:

Let C be a challenger corresponding to *DBDH* problem. C flips coin Θ. if $\Theta = 0$, it sets $(g, A, B, C, Z) = (g, g^a, g^b, g^c, e(g,g)^{abc})$, where a, b, c are chosen randomly; otherwise, it sets Z randomly. Next, C gives (g, A, B, C, Z) to simulator $\mathcal{B}$. $\mathcal{B}$ plays the role of challenger in the following game:

Initialization. $\mathcal{A}$ controls the set of compromised authorities AA_{i_k} $(k \in \{1, \dots, \bar{\mathcal{N}}\})$ containing at most $\mathcal{N} - 1$ authorities. The remaining authorities AA_{i_k} $(k \in \{\bar{\mathcal{N}}+1, \dots, \mathcal{N}\})$ are controlled by $\mathcal{B}$. $\mathcal{A}$ also commits tree access structure $\mathcal{T}$ wanted to be challenged in which some attributes are issued by $\mathcal{A}$'s authorities and the remaining by $\mathcal{B}$.

Setup. Without knowing the actual value of $\sum_{i=1}^{\mathcal{N}} v_i$ and $\sum_{i=1}^{\mathcal{N}} d_i$, $\mathcal{B}$ sets $a = \sum_{i=1}^{\mathcal{N}} d_i$, $b = \sum_{i=1}^{\mathcal{N}} v_i / \sum_{i=1}^{\mathcal{N}} d_i$, $c = s$, where $\sum_{i=1}^{\mathcal{N}} v_i$, $\sum_{i=1}^{\mathcal{N}} d_i$ and s are chosen randomly.[3] Then, it computes $\zeta = e(A, B) = e(g,g)^{\sum_{i=1}^{\mathcal{N}} v_i}$. It selects random elements $\tau_i(\bar{\mathcal{N}}+1 \le i \le \mathcal{N})$ and sets $\vartheta_i = g^{\tau_i}(\bar{\mathcal{N}}+1 \le i \le \mathcal{N})$. Meanwhile, $\mathcal{A}$ selects random elements $\tau_i(1 \le i \le \bar{\mathcal{N}})$ and sets $\vartheta_i = g^{\tau_i}(1 \le i \le \bar{\mathcal{N}})$. Then, they publish public parameters $PK = (\mathbf{G}, g, \zeta, \vartheta_i(1 \le i \le \mathcal{N}), H(.))$.

Learning 1. $\mathcal{A}$ is allowed to have access to an arbitrary number (i.e. p) of secret keys corresponding to chosen attribute sets $\mathbb{A}_{u_1}, \dots, \mathbb{A}_{u_p}$ for a selected number of users $\{U_1, \dots, U_p\}$. These attribute sets are disjointly issued by all authorities $AA_i(i \in \{1, \dots, \mathcal{N}\})$, but none of them satisfies tree access structure $\mathcal{T}$. For each attribute j, issued by an uncompromised authority AA_{i_k} $(k \in \{\bar{\mathcal{N}}+1, \dots, \mathcal{N}\})$, $\mathcal{B}$ randomly picks r_j and computes $D_j = A^{(1/\tau_{iss(j)})}(H(att(j)))^{r_j}$ and $D'_j = g^{r_j}$. Then, it returns its own part of secret key to $\mathcal{A}$. $\mathcal{A}$ is also allowed to compute its own part of secret key (i.e. $\forall j$ *issued by* $AA_{i_k}(k \in \{1, \dots, \bar{\mathcal{N}}\}) : (D_j, D'_j)$), conduct an arbitrary number of computations, using its own (or compromised) secret keys and all public parameters. $\mathcal{A}$ is also allowed to query for an arbitrary number (i.e. $q - p$) of blinded secret keys corresponding to attribute sets $\mathbb{A}_{u_{p+1}}, \dots, \mathbb{A}_{u_q}$ for selected number of users $\{U_{p+1}, \dots, U_q\}$. These attribute sets are also disjointly issued by all authorities $AA_i(i \in \{1, \dots, \mathcal{N}\})$. In response to a query for blinded secret key for attribute set $\mathbb{A}_{u_i}$, $\mathcal{B}$ chooses random components α, r, τ_i $\forall i \in \{1, \dots, \mathcal{N}\}$, $\hat{r}_j$ $\forall j \in$ *attributes*, and sets $SK_{U_i}^{1/b} = \big(D^{1/b} = g^{\alpha} g^{r}, \forall j \in \mathbb{A}_{u_i} : D_j^{1/b} = g^{r/\tau_{iss(j)}}(H(att(j)))^{\hat{r}_j}, D_j'^{1/b} = g^{\hat{r}_j} \big)$.[4] If $\mathcal{B}$ has already received the same query for the same attribute set, it does not compute blinded secret key repeatedly; but, instead it uses the previous computation. Then, it returns the corresponding blinded secret key to $\mathcal{A}$.

Challenge. $\mathcal{A}$ sends two distinct chosen plaintexts M_0 and M_1 to the challenger. The challenger randomly selects bit $\varpi \in \{0,1\}$ and returns ciphertext $CT^* = (M_{\varpi}.Z,\ C, \forall y \in \mathbb{A}_u : C_y,\ C'_y)$ to $\mathcal{A}$. If $\Theta = 0$, $Z = e(g,g)^{abc}$, where $a = \sum_{i=1}^{\mathcal{N}} d_i$, $b = \sum_{i=1}^{\mathcal{N}} v_i / \sum_{i=1}^{\mathcal{N}} d_i$ and $c = s$. Therefore, $Z = e(g,g)^{\alpha s}$ and CT^* is a valid ciphertext of M_{ϖ}. Otherwise, if $\Theta = 1$, Z is a random element and $M_{\varpi}.Z$ does not contain any information about M_{ϖ}.

Learning 2. $\mathcal{A}$ continues to repeat learning 1 adaptively.

Response. $\mathcal{A}$ submits guess ϖ' of ϖ. If $\varpi' = \varpi$, $\mathcal{B}$ outputs $\Theta' = 0$ indicating that $\mathcal{A}$ returned back a valid *DBDH*-tuple $(g, A, B, C, Z) = (g, g^{\sum_{i=1}^{\mathcal{N}} d_i}, g^{\sum_{i=1}^{\mathcal{N}} v_i / \sum_{i=1}^{\mathcal{N}} d_i}, g^s, e(g,g)^{\sum_{i=1}^{\mathcal{N}} v_i s})$. Otherwise, $\mathcal{B}$ outputs $\Theta' = 1$ indicating that $\mathcal{A}$ returned back a random tuple (g, A, B, C, Z).

When $\Theta = 0$, $\mathcal{A}$ gets a valid ciphertext of M_{ϖ}. Since the advantage of adversary $\mathcal{A}$ is ε by definition, the probability of correctly guessing ϖ' of ϖ is $Pr[\varpi' = \varpi | \Theta = 0] = \frac{1}{2} + \varepsilon$. If $\varpi' = \varpi$, $\mathcal{B}$ outputs $\Theta' = 0$. Thus, we have $Pr[\Theta' = \Theta | \Theta = 0] = Pr[\varpi' = \varpi | \Theta = 0] = \frac{1}{2} + \varepsilon$. When $\Theta = 1$, $\mathcal{A}$ has no information about ϖ. In this case, we have $Pr[\varpi' \neq \varpi | \Theta = 1] = Pr[\varpi' = \varpi | \Theta = 1] = \frac{1}{2}$. If $\varpi' \neq \varpi$, $\mathcal{B}$ outputs $\Theta' = 1$. So, we have $Pr[\Theta' = \Theta | \Theta = 1] = Pr[\varpi' \neq \varpi | \Theta = 1] = \frac{1}{2}$. Hence, the overall advantage in this game is:

$$\begin{aligned} Pr[\Theta' = \Theta] - \frac{1}{2} &= Pr[\Theta' = \Theta | \Theta = 1] + Pr[\Theta' = \Theta | \Theta = 0] - \frac{1}{2} \\ &= \frac{1}{2} \times (\frac{1}{2} + \varepsilon) + \frac{1}{2} \times \frac{1}{2} - \frac{1}{2} = \frac{\varepsilon}{2} \end{aligned}$$

[3] Since $\sum_{i=1}^{\mathcal{N}} v_i$ and $\sum_{i=1}^{\mathcal{N}} d_i$ are partially selected by uncompromised authorities AA_{i_k} $(k \in \{\bar{\mathcal{N}}+1, \dots, \mathcal{N}\})$, their randomly assigning does not affect the duty of compromised authorities AA_{i_k} $(k \in \{1, \dots, \bar{\mathcal{N}}\})$, to assign v_{i_k} and d_{i_k} $(k \in \{1, \dots, \bar{\mathcal{N}}\})$, which are under the control of $\mathcal{A}$.

[4] Note that $\mathcal{B}$ does not know the actual retrieving key $b = \sum_{i=1}^{\mathcal{N}} v_i / \alpha$. Consequently, it cannot deduce $r = \sum_{i=1}^{\mathcal{N}} d_i / b$ and $\forall j \in \mathbb{A}_{u_i} : g^{(r/\tau_{iss(j)})} = g^{(\sum_{i=1}^{\mathcal{N}} d_i / b\tau_{iss(j)})}$, $g^{r_j} = g^{\hat{r}_j / b}$.

Thus, we are able to construct *PPT* algorithm $\mathcal{B}$ that can solve the *DBDH* problem with advantage $\frac{\epsilon}{2}$. $\square$

Theorem 4 (*Anonymizing User Identity*). *Assume that the identity of user U is revealed to an attribute authority while it issues part of his secret key (i.e. SK_U). Under CDH, PACS anonymizes U's identity, even if the adversary compromises all authorities.*

Proof. Let $\mathcal{A}$ be a *PPT* algorithm that can derive the identity of user U while requesting to access a file. More specifically, without knowing secret retrieving key b, $\mathcal{A}$ would be able to extract some parameter P in user U's secret key $SK_U = \Big(g^{\sum_{i=1}^{\mathcal{N}}(v_i+d_{i,u})}, \forall j \in \mathbb{A}_u : (g^{\sum_{i=1}^{\mathcal{N}} d_{i,u}/\tau_{iss(j)}}(H(att(j)))^{r_{j,u}}, g^{r_{j,u}}) \Big)$ from corresponding $P^{1/b}$ in $SK_U^{1/b} = \Big(g^{\sum_{i=1}^{\mathcal{N}}(v_i+d_{i,u}/b)}, \forall j \in \mathbb{A}_u : (g^{\sum_{i=1}^{\mathcal{N}} d_{i,u}/\tau_{iss(j)} b}(H(att(j)))^{r_{j,u}/b}, g^{r_{j,u}/b}) \Big)$. We show that if $\mathcal{A}$ has non-negligible advantage to extract P from $P^{1/b}$, we can construct *PPT* algorithm $\mathcal{B}$ with non-negligible advantage to solve *CDH* problem $(g_1, g_1^{\hat{\mathfrak{a}}}, g_1^{\hat{\mathfrak{b}}}) \rightarrow g_1^{\hat{\mathfrak{a}}\hat{\mathfrak{b}}}$.

To achieve this goal, $\mathcal{B}$ sets $g = g_1$ and $P^{1/b} = g_1^{\hat{\mathfrak{a}}}$, where $b = \hat{\mathfrak{b}}$. Then, it sends $P^{1/b}$ as input to $\mathcal{A}$. Note that $\mathcal{B}$ does not know the actual value of $\hat{\mathfrak{b}}$. Upon receiving the corresponding output P, without changing anything, $\mathcal{B}$ transfers it as its own output. $\square$

9. SPACS: Statistical Privacy-Preserving Access Control Scheme

As described in Section 4, the statistical analysis of the combination of attributes belonging to the quasi-identifier of users can be used to re-identify them. In *PACS*, when a user sends his request to have access to a ciphertext encrypted and stored in the cloud, the adversary, with the help of malicious *CSP*, can access the set of attributes belonging to the quasi-identifier of the user and re-identify him. In this section, aiming to provide anonymity of users against statistical analysis of their attributes, we propose an extension of *PACS*, called *SPACS*, which supports $\mathcal{K}$-anonymity of users without trusting authorities or providers. Since supporting $\mathcal{K}$-anonymity of users against statistical analysis of attributes requires issuing at least $\mathcal{K}$ possible values for the combination of attributes belonging to quasi-identifier of users, we assume that each attribute $att(i)$ included in quasi-identifier of users (e.g. $att(i) = Gender$) takes its value $v_{att}(i)$ from a set (an equivalence class of anonymity) of at least $\mathcal{K}_i$ possible attribute values $v_{att}(i) \in \{v_j(i)\}_{1 \le j \le \mathcal{K}_i, \mathcal{K} \le \mathcal{K}_i}$ (e.g. $v_{att}(i) \in \{male, female\}$). Each attribute authority issues just one attribute value for each user attribute it is responsible for. Later on, users send their queries including their attributes (not attribute values), authorized by authorities, to *CSP*. In encryption process, based on attribute values *DO* considers for each attribute, he encrypts plaintext M and uploads it to the cloud server. Ideally, different attribute values would be encrypted using separate tree access structures. However, to minimize computational overhead, *DO* constructs a single tree access structure in which different attribute values $v_j(x)$ are assigned to a leaf node associated with attribute $att(x)$. This allows *DO* to compute different $C'i_j$ corresponding to the various attribute values $v_j(x)$, streamlining the encryption process. We assume that the policy along with different attribute values used to encrypt M are publicly accessible; this allows user U to check whether he is eligible to access M or not. Based on the defined policy embedded in ciphertext, U sends his blinded request to *CSP*. *CSP* receives the list of attributes (not the values) and blinded secret key of U. It calculates all possible blinded ciphertexts CT's and sends them back to U. U verifies the outsourced computations of blinded ciphertexts; if the verification passes for one CT', U would be able to un-blind it and have access to plaintext M embedded in ciphertext. This verification can be done using commitment techniques similar to the techniques described in [38,39]. We can describe the framework of *SPACS* by the following six algorithms. Similar to *PACS*, these algorithms are performed in four phases: setup, key generation, encryption, and decryption.

Setup $(\mathbb{A}, \Lambda) \longrightarrow (PK, MK_{a_i(1 \le i \le \mathcal{N})})$: Similar to *PACS*, attribute authorities collaborate to construct AA_k's master key $MK_{a_k} = (v_k, \tau_k, x_k)$ and public parameters $PK = (S, g, \zeta = e(g,g)^{\sum_{i=1}^{\mathcal{N}} v_i}, \vartheta_i = g^{\tau_i}(1 \le i \le \mathcal{N}), H(.))$.

KeyGeneration $(PK, MK_{a_i(1 \le i \le \mathcal{N})}, \mathbb{A}_u) \longrightarrow SK_U$: In a similar way to *PACS*, each authority AA_k computes $D = g^{\sum_{j=1}^{\mathcal{N}}(v_j+d_j)}$ and sends it to user U. Then, for each attribute $att(j) \in \mathbb{A}_u$ authorized by AA_k, it selects one attribute value $v_{att}(j)$ among all possible values, computes $D_j = g^{(\sum_{i=1}^{\mathcal{N}} d_i/\tau_k)}(H(v_{att}(j)))^{r_j}$ and $D'_j = g^{r_j}$, and sends D_j, D'_j along with $v_{att}(j)$ to user U. The user's secret key would be $SK_U = (D, \forall j \in \mathbb{A}_u : (D_j, D'_j))$.

KeyGenOut $(PK, SK_U, b) \longrightarrow SK_U^{1/b}$: Similar to *PACS*, user U chooses random secret retrieving key $b \in \mathbb{Z}_p^*$ and transforms secret key SK_U to its blinded version (i.e. $SK_U^{1/b} = (D^{1/b}, \forall j \in \mathbb{A}_u : (D_j^{1/b}, {D'_j}^{1/b}))$).

Encrypt $(PK, \mathcal{T}, M) \longrightarrow CT$: Similar to *PACS*, data owner *DO* selects random number $s, \bar{s} \in_r \mathbb{Z}_p^*$ and shares it in tree access structures $\mathcal{T}$ with root R. Each leaf node y in tree access structure $\mathcal{T}$ is associated with attribute value $v_{att}(y)$ assigned for attribute $att(y)$. It selects two generators $\mathfrak{g}$ and $\mathfrak{h}$ for $\mathbb{Z}_p^*$, chooses random message $\bar{M}$ and computes $Commit(H(M), H(\bar{M})) = (\mathfrak{g}^{H(M)}.\mathfrak{h}^{H(\bar{M})})$. Then, it uploads the ciphertext $CT = \Big(\mathfrak{g}, \mathfrak{h}, \hat{C} = Commit(H(M), H(\bar{M})), E = M.\zeta^s, \bar{E} = \bar{M}.\zeta^s, C = g^s, \forall i \in \mathbb{A}_{\mathcal{T}} : (C_i = \vartheta_{iss(i)}^{q_i(0)} = g^{(\tau_{iss(i)}\ q_i(0))}, C'_i = H(v_{att}(y))^{q_i(0)})\Big)$ to *CSP*.

As we mentioned earlier, different attribute values $v_j(x)$s, considered by *DO* to be associated with attribute $att(x)$, should be embedded in different tree access structures. However, to reduce the computation cost of encryption, *DO* can construct just one tree access structure $\mathcal{T}$, assign different attribute values $v_j(x)$s to a leaf node with attribute $att(x)$ in $\mathcal{T}$ and compute different C'_{i_j} corresponding to different attribute values $v_j(x)$s (i.e. computing different $\{C'_{i_j}\}_j$ for one tree access structure instead of constructing different tree access structures and computing C'_i for each of them).

DecryptionOut $(PK, SK_U^{1/b}, \mathbb{A}_u, CT) \longrightarrow CT'$: To have access to plaintext M embedded in CT, user U checks if the values of his attributes satisfy the policy defined for that ciphertext. If yes, he sends his blinded secret key $SK_U^{1/b}$, and his attribute set $\mathbb{A}_u$ (not the values) to *CSP*. For each leaf node x in tree access structure $\mathcal{T}$ and corresponding attribute $att(i) \in \mathbb{A}_u$, *CSP* computes $DecNode(CT, SK_U^{1/b}, x) = \frac{e(D_i^{1/b}, C_x)}{e(D'_i, C'_x)}$. If the user's attribute value, embedded in D_i, is equal to the value defined for attribute of node x, $DecNode(CT, D_i^{1/b}, x)$ will be equal to $e(g,g)^{q_x(0)\Sigma d_k/b}$. *CSP* recursively computes A as $DecNode(CT, SK_U^{1/b}, R)$ for root node R of tree access structure $\mathcal{T}$. If the user is eligible to access plaintext M and user's attributes values ($\forall att(i) \in \mathbb{A}_u$) are equal to the corresponding values embedded in $\mathcal{T}$, then A will be equal to $e(g,g)^{q_R(0)\Sigma d_k/b} = e(g,g)^{s\Sigma d_k/b}$ and $\zeta^{s/b} = \frac{e(C, D^{1/b})}{A}$ would be equal to $e(g,g)^{\sum_{i=1}^{\mathcal{N}} v_i s/b}$. Then, *CSP* sends all possible $CT' = (\hat{C}, E, \bar{E}, \zeta^{s/b})$ to U.

Decrypt $(PK, b, CT') \longrightarrow M$: User U computes ζ^s using $\zeta^{s/b}$ and his retrieving key b. He extracts M and $\bar{M}$ as follows: $M = E/\zeta^s$ and $\bar{M} = \bar{E}/\zeta^s$. . Then, it checks commitment $\hat{C} \overset{?}{=} Commit(H(M), H(\bar{M}))$. If it holds, U has access to valid plaintext M.

Providing user U with privacy protection and anonymity against statistical analysis requires anonymizing his attributes included in quasi-identifier of users. Thus, to reduce the computation overhead, we need to provide anonymity, and define $\mathcal{K}$ possible values for a subset of attributes which are part of quasi-identifier (and not all of them).

Theorem 5. *If PACS is selectively secure and indistinguishable against chosen plaintext attacks (IND-CPA), then SPACS is selectively secure and indistinguishable against chosen plaintext attacks (IND-CPA).*

Proof. First, we assume that there is just one possible attribute value $v_{att}(i)$ for each attribute $att(i)$ (i.e. $\mathcal{K} = 1$). In this case, according to the construction of *SPACS*, compared to *PACS*, it has additional parts

Table 2
Performance comparison of *PACS*, *SPACS*, [18–21]: Computation complexity and security characteristics (*FGA: Fine-Grained Access, MA: Multi Authority, UA: User Anonymity, USA: User Statistical Anonymity, UAU: User Authorization, CA: Collusion Attacks, CPA: Chosen Plaintext Attacks*).

Scheme	Computation overhead in phase				Supporting					Immunity to	
	Key Gen. (U)	Key Gen. (AA_i)	Enc. (DO)	Dec. (U)	*FGA*	*MA*	*UA*	*USA*	*UAU*	*CA*	*CPA*
[19]	0	$(6\vert\mathbb{A}\vert+4)T_{E_G}$	$(3\sum_{i=1}^{\vert\mathcal{T}\vert} N_i+3)T_{E_G}+2T_{E_{G_T}}$	$(2\sum_{i=1}^{\vert\mathcal{T}\vert} N_i+4)T_P$	✓	×	✓	×	✓	✓	✓
[18]	0	$(6\vert\mathbb{A}\vert+4)T_{E_G}$	$(3\sum_{i=1}^{\vert\mathcal{T}\vert} N_i+3)T_{E_G}+2T_{E_{G_T}}$	$(2\sum_{i=1}^{\vert\mathcal{T}\vert} N_i+2\vert\mathcal{T}\vert+2)T_P$	✓	×	✓	×	✓	✓	✓
[20]	0	$(2\vert\mathbb{A}_u\vert/(\mathcal{N}+2))T_{E_G}$	$(2\vert\mathcal{T}\vert+2)T_{E_G}+T_{E_{G_T}}$	$(4\vert\mathbb{A}_u\vert+1)T_P$	×	✓	✓	×	×	×	×
[21]	0	$(\vert\mathbb{A}_u\vert(\mathcal{N}-1)+2\vert\mathbb{A}_u\vert/(\mathcal{N}+2))T_{E_G}$	$(2\vert\mathcal{T}\vert+2)T_{E_G}+T_{E_{G_T}}$	$(4\vert\mathbb{A}_u\vert+1)T_P$	×	✓	✓	×	×	×	×
PACS	$(2\vert\mathbb{A}_u\vert+1)T_{E_G}$	$(2\vert\mathbb{A}_u\vert/(\mathcal{N}+2))T_{E_G}$	$(2\vert\mathcal{T}\vert+1)T_{E_G}+T_{E_{G_T}}$	$T_{E_{G_T}}$	✓	✓	✓	×	✓	✓	✓
SPACS	$(2\vert\mathbb{A}_u\vert+1)T_{E_G}$	$(2\vert\mathbb{A}_u\vert/(\mathcal{N}+2))T_{E_G}$	$(2\vert\mathcal{T}\vert+2+\vert v_{att}\vert)T_{E_G}+T_{E_{G_T}}$	$(3\vert v_{att}\vert)T_{E_{G_T}}$	✓	✓	✓	✓	✓	✓	✓

$\mathfrak{g}$, $\mathfrak{h}$, $\hat{C} = Commit(H(M), H(\bar{M}))$, and $\bar{E} = \bar{M}.\zeta^s$ in a ciphertext. Thus, to prove the selective security of *SPACS* against *CPA*, we need to show that the additional parts do not disclose any more information about the chosen plaintext, selected by challenger $\mathcal{C}$, in *CPA* model defined for *PACS*. To do that, we define the following two games:

- $Game_0$: The original game defined for selective security and indistinguishability against chosen plaintext attacks (*IND-CPA*) of *PACS*.
- $Game_1$: Same as $Game_0$ except the way challenger $\mathcal{C}$ generates challenge ciphertext $CT = \left(\mathfrak{g}, \mathfrak{h}, \hat{C}, E, \bar{E}, C, \forall i \in \mathbb{A}_{\mathcal{T}} : (C_i, C'_i)\right)$ in which challenger $\mathcal{C}$ selects $\mathfrak{g}$, $\mathfrak{h}$, $\hat{C}$ and $\bar{E}$ randomly.

Since $\mathfrak{g}$, $\mathfrak{h}$ are selected randomly and since *Pedersen Commitment Scheme* is perfectly hiding [31], by proving the negligibility of the advantage of the adversary in $Game_1$, we can deduce that, for $\mathcal{K} = 1$, the construction of *SPACS*, which is *PACS* equipped with commitment part, is also selective secure and indistinguishable against chosen plaintext attacks. To do that, let us assume that $\mathcal{A}$ be a *PPT* adversary in selective secure *CPA* model defined by $Game_1$. We show that if adversary $\mathcal{A}$ has non-negligible advantage in $Game_1$, we can construct *PPT* algorithm $\mathcal{B}$ that can break the security of *PACS* in selective secure *CPA* model (i.e. $Game_0$) with non-negligible advantage. Let $\mathcal{C}$ be a challenger corresponding to $\mathcal{B}$. The simulator $\mathcal{B}$, as an intermediary layer, simultaneously plays the role of challenger for $\mathcal{A}$, collaborates with $\mathcal{A}$ to run initialization, setup, and first round learning phases in both $Game_0$ and $Game_1$. As a challenger in $Game_1$, $\mathcal{B}$ takes the items selected by $\mathcal{A}$ in initialization, setup, and first round learning phases, transfers them to $\mathcal{C}$ in $Game_0$ and returns back its responses to $\mathcal{A}$. In challenge phase, $\mathcal{A}$ selects two distinct chosen plaintexts M_0 and M_1 and sends them to $\mathcal{C}$ via $\mathcal{B}$. $\mathcal{C}$ selects bit $\varpi \in \{0,1\}$ uniformly at random, and returns the ciphertext $(E, C, \forall i \in \mathbb{A}_{\mathcal{T}} : (C_i, C'_i))$ to $\mathcal{B}$. $\mathcal{B}$ selects $\mathfrak{g}$, $\mathfrak{h}$, $\hat{C}$ and $\bar{E}$ randomly, and sends its challenge ciphertext $CT = (\mathfrak{g}, \mathfrak{h}, \hat{C}, E, \bar{E}, C, \forall i \in \mathbb{A}_{\mathcal{T}} : (C_i, C'_i))$ to $\mathcal{A}$. After performing second round of learning, similar to the first one, $\mathcal{A}$ selects its own guess ϖ' of ϖ and sends it to $\mathcal{B}$. Then, $\mathcal{B}$ transfers guess ϖ' as its own output.

Thus, for $\mathcal{K} = 1$ if there exists a *PPT* algorithm with non-negligible advantage to break security of *SPACS* in selective secure *CPA* model, we are able to construct a *PPT* algorithm with non-negligible advantage to break security of *PACS* in selective secure *CPA* model. Moreover, since for the case that $\mathcal{K} \geq 1$, the adversary has at most the same knowledge as that for $\mathcal{K} = 1$, *SPACS* will remain selective secure and indistinguishable against chosen plaintext attacks. □

Theorem 6. *Assume that the identity of each user U can be recognized from his set of attributes $\mathbb{A}_u$. If attribute values $v_{att}(i)$s considered for each attribute att(i) belong to a set of at least $\mathcal{K}$ possible attribute values $v_{att}(i) \in \{v_j(i)\}_{1\le j\le\mathcal{K}_i, \mathcal{K}\le\mathcal{K}_i}$, then SPACS satisfies $\mathcal{K}$-anonymity for all users.*

Proof. Based on the construction of *SPACS*, since each user's attribute *att*(i) takes its value $v_{att}(i)$ from a set of at least $\mathcal{K}$ possible attribute values $\{v_j(i)\}_{1\le j\le\mathcal{K}_i, \mathcal{K}\le\mathcal{K}_i}$, then any sequence of attribute values $\left\langle v_{att}(i'_1), \dots, v_{att}(i'_{|\mathbb{A}_u|})\right\rangle$ associated with user U (respectively, $\left\langle v_{att}(i'_1), \dots, v_{att}(i'_{|\mathbb{A}_u^+|})\right\rangle$ associated with the quasi-identifier of U) belongs to a set of at least $\mathcal{K}$ possible indistinguishable sequences of attribute values $\left\langle \{v_j(i'_1)\}_{1\le j\le\mathcal{K}_{i'_1}}, \dots, \{v_j(i'_{|\mathbb{A}_u|})\}_{1\le j\le\mathcal{K}_{i'_{|\mathbb{A}_u|}}} \right\rangle$ (respectively, $\left\langle \{v_j(i'_1)\}_{1\le j\le\mathcal{K}_{i'_1}}, \dots, \{v_j(i'_{|\mathbb{A}_u^+|})\}_{1\le j\le\mathcal{K}_{i'_{|\mathbb{A}_u^+|}}} \right\rangle$). Thus, according to $\mathcal{K}$-anonymity definition for individual users, *SPACS* is $\mathcal{K}$-anonymous for all users. □

10. Performance evaluation

To evaluate the performance of the proposed schemes, we compare computation overhead of *PACS* and *SPACS* against state of the art including [18–21]. Let $|\mathcal{T}|$ be the number of leaves in the tree access structure, $E_{\mathbb{G}}$ and $E_{\mathbb{G}_T}$ denote the exponentiation operations in groups $\mathbb{G}$ and $\mathbb{G}_T$ respectively, P indicates the bilinear pairing operation in $\mathbb{G}/\mathbb{G}_T$, $T_{\mathcal{O}}$ represents the time taken to perform one operation $\mathcal{O}$, $\mathbb{A}_u$ denotes the user's set of attributes involved in encryption and decryption, $|v_{att}|$ be the number of attribute values considered for combination of all attributes that belong to quasi-identifier, $\mathcal{N}$ be the number of authorities, $l_{\mathbb{G}}$ and $l_{\mathbb{G}_T}$ denote the length of elements in groups $\mathbb{G}$ and $\mathbb{G}_T$ respectively. Table 2 shows a comparison for computation complexity of *PACS*, *SPACS* and the schemes presented in [18–21] in terms of *Key Generation*, *Encryption* and *Decryption*. The computation complexity concerns the most significant operations, namely exponentiation operations $E_{\mathbb{G}}$ and $E_{\mathbb{G}_T}$, and pairing operation P.

To perform the evaluation, we make use of *MNT159* curves type D with the embedding degree $k = 6$. Our evaluation is based on *Java* realization of *CP-ABE* toolkit [40] and uses *Java Pairing-Based Cryptography* (*jPBC*) library [41] (i.e. a port of the *Pairing-Based Cryptography* (*PBC*) library [42] to perform the mathematical operations underlying pairing-based cryptosystems directly in *Java*). The analysis is conducted based on a platform consisting of (a) a large instance Amazon EC2 with the configuration of 7.5 GB RAM, 4 EC2 compute units, 850 GB instance storage and 64-bit platform as *CSP*, and (b) an Apple iMac with Intel Core 2 Duo at 2.66 GHz, 4 GB RAM as data owner, user and attribute authorities. We evaluate the time overhead caused by authorities and users interactions in *Setup* and *Key Generation* phases using *OMNET++* and *INET* framework. We compare the performance of *PACS* and *SPACS* with other existing anonymous attribute-based solutions [18–21]. For our comparison, we assume that $\mathcal{K}$ is equal to three and the number of attributes involved in quasi-identifier is equal to five (i.e. $|\mathbb{A}_u^+| = 5$).

Figs. 3 shows the computation overhead of *PACS*, *SPACS* and [18–21]. More specifically, Figs. 3(a) and 3(b) show the impact of the number of authorities and number of attributes associated with user, in *Setup* and *Key Generation* phases. We observe that by varying either the number of authorities (see Fig. 3(a)) or the number of attributes (see Fig. 3(b)), *PACS* and *SPACS* incur almost the same computation times for authorities in *Setup* and *Key Generation* phases compared to those incurred by [20] (and less in comparison to those incurred by [21]). Moreover, to outsource the computation overhead in decryption process to *CSP*, *PACS* and *SPACS* impose a small computation overhead on user (which is caused by *KeyGenOut* algorithm, see Section 7.2). This results in reducing the computation overhead of user in decryption process to almost constant irrespective of the number of attributes

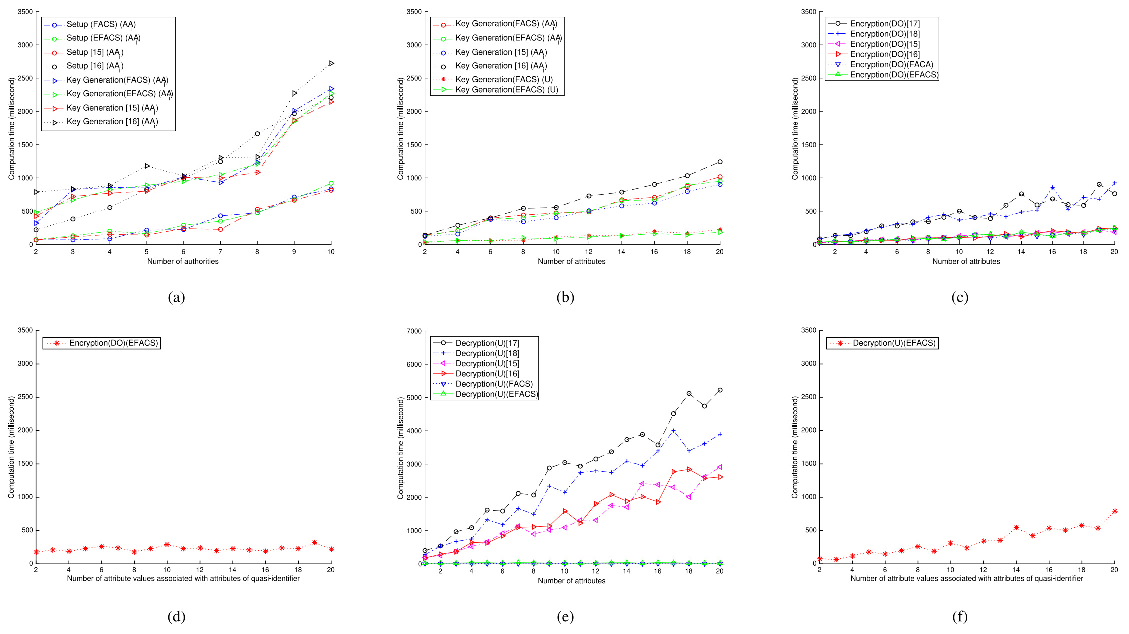


Fig. 3. Computation time analysis of the proposed schemes: (a) The impact of number of authorities in *Setup* and *Key Generation* overhead while each authority issues five attributes for each user, (b) The impact of number of attributes for each user in *Setup* and *Key Generation* overhead ($\mathcal{N} = 5$), (c) The impact of number of attributes in *Encryption* overhead, (d) The impact of number of attribute values, associated with attributes included in quasi-identifier, in *Encryption* overhead ($number\ of\ attributes = 20$), (e) The impact of number of attributes in *Decryption* overhead, (f) The impact of number of attribute values, associated with the combination of attributes included in quasi-identifier, in *Decryption* overhead ($number\ of\ attributes = 20$).

(see Fig. 3(e)). Indeed, compared to [18–21], *PACS* and *SPACS* impose negligible computation overhead on user side in decryption process. Fig. 3(c) shows almost similar computation time overhead for [20,21], *PACS* and *SPACS* in *Encryption* process, while the number of attributes increases from 1 to 20 (and less compared to [18,19]). Here, since none of [18–21] satisfies user statistical anonymity (See Table 2), to get a fair comparison, we assume one attribute value for each attribute in *Encryption* and *Decryption* processes of *SPACS* (See Figs. 3(c) and 3(e)).

Figs. 3(d) and 3(f) illustrate the impact of number of attribute values associated with the combination of attributes included in quasi-identifier, on *Encryption* and *Decryption* processes while using *SPACS*. We observe that *SPACS* imposes a small computation overhead for *DO* in *Encryption* process, irrespective of the number of attribute values (See Fig. 3(d)). Moreover, it introduces a small computation overhead (increases linearly) for user in *Decryption* process, while the combination of attribute values in quasi-identifier varies from 1 to 20 different values (See Fig. 3(f)).

11. Conclusion

In this paper, we discussed how to preserve user privacy in the *ABE* systems. Our investigation shows that supporting anonymity of users requires providing anonymity for their individual-specific attributes as well as their identities. Thus, we developed a new model of anonymity to include users' individual-specific attributes as well as their identities for attribute-based encryption. We reviewed existing contributions in anonymous attribute-based encryption and did show their weaknesses in users and authorities collusion, user authorization and user anonymity protection. Furthermore, we proposed *PACS*, a fine-grained access control for multi-authority supporting and user anonymization (without any trusted third-party). We showed its immunity against users collusion, authority collusion, and chosen plaintext attacks. We also proposed an extension of *PACS*, called *SPACS*, which supports statistical anonymity for attributes and individual users without trusting authorities or providers. The security analysis and performance evaluation did show that the proposed solution is a promising approach to use anonymous *ABE*.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Yaser Baseri reports that financial support was provided by Fonds de Recherche du Québec - Nature et Technologies (FRQNT) and by the Natural Sciences and Engineering Research Council of Canada (NSERC).

References

[1] Abdullayeva F. Cyber resilience and cyber security issues of intelligent cloud computing systems. Results Control Optim 2023;12:100268. http://dx.doi.org/10.1016/j.rico.2023.100268.

[2] Firoozjaei MD, Mahmoudyar N, Baseri Y, Ghorbani AA. An evaluation framework for industrial control system cyber incidents. Int J Crit Infrastruct Prot 2022;36:100487. http://dx.doi.org/10.1016/j.ijcip.2021.100487.

[3] Kim I, Susilo W, Baek J, Kim J. Harnessing policy authenticity for hidden ciphertext policy attribute-based encryption. IEEE Trans Dependable Secure Comput 2022;19(3):1856–70. http://dx.doi.org/10.1109/TDSC.2020.3040712.

[4] Chen N, Li J, Zhang Y, Guo Y. Efficient CP-ABE scheme with shared decryption in cloud storage. IEEE Trans Comput 2022;71(1):175–84. http://dx.doi.org/10.1109/TC.2020.3043950.

[5] Jiang R, Wu X, Bhargava B. SDSS-MAC: Secure data sharing scheme in multi-authority cloud storage systems. Comput Secur 2016;62:193–212. http://dx.doi.org/10.1016/j.cose.2016.07.007.

[6] Baseri Y, Hafid A, Cherkaoui S. Privacy preserving fine-grained location-based access control for mobile cloud. Comput Secur 2018;73:249–65. http://dx.doi.org/10.1016/j.cose.2017.10.014.

[7] Qian H, Li J, Zhang Y, Han J. Privacy-preserving personal health record using multi-authority attribute-based encryption with revocation. Int J Inf Secur 2015;14:487–97. http://dx.doi.org/10.1007/s10207-014-0270-9.

[8] Li J, Zhang R, Lu Y, Han J, Zhang Y, Zhang W, Dong X. Multiauthority attribute-based encryption for assuring data deletion. IEEE Syst J 2023;17(2):2029–38. http://dx.doi.org/10.1109/JSYST.2022.3208149.

[9] Ali M, Mohajeri J, Sadeghi M-R, Liu X. A fully distributed hierarchical attribute-based encryption scheme. Theoret Comput Sci 2020;815:25–46. http://dx.doi.org/10.1016/j.tcs.2020.02.030.

[10] Li J, Yao W, Zhang Y, Qian H, Han J. Flexible and fine-grained attribute-based data storage in cloud computing. IEEE Trans Serv Comput 2017;10(5):785–96. http://dx.doi.org/10.1109/TSC.2016.2520932.

[11] Li J, Zhang Y, Ning J, Huang X, Poh GS, Wang D. Attribute based encryption with privacy protection and accountability for CloudIoT. IEEE Trans Cloud Comput 2022;10(2):762–73. http://dx.doi.org/10.1109/TCC.2020.2975184.

[12] Sweeney L. K-anonymity: A model for protecting privacy. Internat J Uncertain Fuzziness Knowledge-Based Systems 2002;10(05):557–70. http://dx.doi.org/10.1142/S0218488502001648.

[13] Hirschprung RS, Leshman O. Privacy disclosure by de-anonymization using music preferences and selections. Telemat Inform 2021;59:101564. http://dx.doi.org/10.1016/j.tele.2021.101564.

[14] LeFevre K, DeWitt D, Ramakrishnan R. Mondrian multidimensional K-anonymity. In: 22nd international conference on data engineering. ICDE'06, 2006, p. 25. http://dx.doi.org/10.1109/ICDE.2006.101.

[15] Sei Y, Okumura H, Takenouchi T, Ohsuga A. Anonymization of sensitive quasi-identifiers for l-diversity and t-closeness. IEEE Trans Dependable Secure Comput 2019;16(4):580–93. http://dx.doi.org/10.1109/TDSC.2017.2698472.

[16] Rebollo-Monedero D, Forné J, Soriano M, Puiggalí Allepuz J. P-probabilistic k-anonymous microaggregation for the anonymization of surveys with uncertain participation. Inform Sci 2017;382–383:388–414. http://dx.doi.org/10.1016/j.ins.2016.12.002.

[17] Kabir ME, Mahmood AN, Wang H, Mustafa AK. Microaggregation sorting framework for K-Anonymity statistical disclosure control in cloud computing. IEEE Trans Cloud Comput 2020;8(2):408–17. http://dx.doi.org/10.1109/TCC.2015.2469649.

[18] Zhang Y, Li J, Chen X, Li H. Anonymous attribute-based proxy re-encryption for access control in cloud computing. Secur Commun Netw 2016;9(14):2397–411. http://dx.doi.org/10.1002/sec.1509.

[19] Zhang Y, Chen X, Li J, Wong DS, Li H, You I. Ensuring attribute privacy protection and fast decryption for outsourced data security in mobile cloud computing. Inform Sci 2017;379:42–61. http://dx.doi.org/10.1016/j.ins.2016.04.015.

[20] Jung T, Li X-Y, Wan Z, Wan M. Control cloud data access privilege and anonymity with fully anonymous attribute-based encryption. IEEE Trans Inf Forensics Secur 2015;10(1):190–9. http://dx.doi.org/10.1109/TIFS.2014.2368352.

[21] Jung T, Li XY, Wan Z, Wan M. Rebuttal to comments on control cloud data access privilege and anonymity with fully anonymous attribute-based encryption. IEEE Trans Inf Forensics Secur 2016;11(4):868. http://dx.doi.org/10.1109/TIFS.2015.2509946.

[22] Zhang Y, Chen X, Li J, Wong DS, Li H. Anonymous attribute-based encryption supporting efficient decryption test. In: Proceedings of the 8th ACM SIGSAC symposium on information, computer and communications security. p. 511–6. http://dx.doi.org/10.1145/2484313.2484381.

[23] Nasiraee H, Ashouri-Talouki M. Privacy-preserving distributed data access control for CloudIoT. IEEE Trans Dependable Secure Comput 2022;19(4):2476–87. http://dx.doi.org/10.1109/TDSC.2021.3060337.

[24] Ramu G. A secure cloud framework to share EHRs using modified CP-ABE and the attribute bloom filter. Educ Inf Technol 2018;23(5):2213–33. http://dx.doi.org/10.1007/s10639-018-9713-7.

[25] Zhang M, Zhou J, Zhang G, Cui L, Gao T, Yu S. APDP: Attribute-based personalized differential privacy data publishing scheme for social networks. IEEE Trans Netw Sci Eng 2023;10(2):922–33. http://dx.doi.org/10.1109/TNSE.2022.3224731.

[26] Michalevsky Y, Joye M. Decentralized policy-hiding ABE with receiver privacy. In: Lopez J, Zhou J, Soriano M, editors. Computer security. Cham: Springer International Publishing; 2018, p. 548–67. http://dx.doi.org/10.1007/978-3-319-98989-1_27.

[27] Wu A, Zhang Y, Zheng X, Guo R, Zhao Q, Zheng D. Efficient and privacy-preserving traceable attribute-based encryption in blockchain. Ann Telecommun 2019;74:401–11. http://dx.doi.org/10.1007/s12243-018-00699-y.

[28] Jung T, Li X-Y, Wan Z, Wan M. Privacy preserving cloud data access with multi-authorities. In: 2013 proceedings IEEE INFOCOm. 2013, p. 2625–33. http://dx.doi.org/10.1109/INFCOM.2013.6567070.

[29] Bethencourt J, Sahai A, Waters B. Ciphertext-policy attribute-based encryption. In: 2007 IEEE symposium on security and privacy. SP'07, 2007, p. 321–34. http://dx.doi.org/10.1109/SP.2007.11.

[30] Butler D, Lochbihler A, Aspinall D, Gascón A. Formalising Σ-protocols and commitment schemes using CryptHOL. J Automat Reason 2021;65(4):521–67. http://dx.doi.org/10.1007/s10817-020-09581-w.

[31] Pedersen TP. Non-interactive and information-theoretic secure verifiable secret sharing. In: Feigenbaum J, editor. Advances in cryptology — CRYPTO '91. Berlin, Heidelberg: Springer Berlin Heidelberg; 1992, p. 129–40. http://dx.doi.org/10.1007/3-540-46766-1_9.

[32] Sweeney L. Computational disclosure control: A primer on data privacy protection (Ph.D. thesis), Massachusetts Institute of Technology; 2001, URL https://dspace.mit.edu/handle/1721.1/8589.

[33] Samarati P. Protecting respondents identities in microdata release. IEEE Trans Knowl Data Eng 2001;13(6):1010–27. http://dx.doi.org/10.1109/69.971193.

[34] Ma H, Zhang R, Yuan W. Comments on "Control cloud data access privilege and anonymity with fully anonymous attribute-based encryption". IEEE Trans Inf Forensics Secur 2016;11(4):866–7. http://dx.doi.org/10.1109/TIFS.2015.2509865.

[35] Chen S, Jero S, Jagielski M, Boldyreva A, Nita-Rotaru C. Secure communication channel establishment: TLS 1.3 (over TCP fast open) versus QUIC. J Cryptology 2021;34(3):26. http://dx.doi.org/10.1007/s00145-021-09389-w.

[36] Li J, Yao W, Han J, Zhang Y, Shen J. User collusion avoidance CP-ABE with efficient attribute revocation for cloud storage. IEEE Syst J 2018;12(2):1767–77. http://dx.doi.org/10.1109/JSYST.2017.2667679.

[37] Lai J, Deng R, Guan C, Weng J. Attribute-based encryption with verifiable outsourced decryption. IEEE Trans Inf Forensics Secur 2013;8(8):1343–54. http://dx.doi.org/10.1109/TIFS.2013.2271848.

[38] Qin B, Deng R, Liu S, Ma S. Attribute-based encryption with efficient verifiable outsourced decryption. IEEE Trans Inf Forensics Secur 2015;10(7):1384–93. http://dx.doi.org/10.1109/TIFS.2015.2410137.

[39] Mao X, Lai J, Mei Q, Chen K, Weng J. Generic and efficient constructions of attribute-based encryption with verifiable outsourced decryption. IEEE Trans Dependable Secure Comput 2016;13(5):533–46. http://dx.doi.org/10.1109/TDSC.2015.2423669.

[40] Junwei W. Java realization for ciphertext-policy attribute based encryption (CP-ABE) http://hotfixs.com/cpabe/.

[41] De Caro A, Iovino V. jPBC library–the Java realization for pairing-based cryptography. http://gas.dia.unisa.it/projects/jpbc/.

[42] Lynn B. Pairing-based cryptography library (PBC). 2024, https://crypto.stanford.edu/pbc/. [Accessed 14 April 2024].