Accountable Privacy Preserving Attribute based Framework for Authenticated Encrypted Access in Clouds

Sana Belguith[1], Nesrine Kaaniche[2], Maryline Laurent[3], Abderrazak Jemai[4], Rabah Attia[5]
[1] School of Computing, Science and Engineering, University of Salford, Manchester, UK
[2]Department of Computer Science, University of Sheffield, UK
[3]SAMOVAR, CNRS, Télécom SudParis, Institut Polytechnique de Paris, France
Member of the Chair Values and Policies of Personal Information
[4]Université de Carthage, Ecole Polytechnique de Tunisie, Laboratoire SERCOM, INSAT, 1080, Tunis, Tunisie
[5]SERCom Lab, Ecole Polytechnique de Tunisie, Université de Carthage, Tunisie

Abstract—**In this paper, we propose an accountable privacy preserving attribute-based framework, called Ins-PAbAC, that combines attribute based encryption and attribute based signature techniques for securely sharing outsourced data contents via public cloud servers. The proposed framework presents several advantages. First, it provides an encrypted access control feature, enforced at the data owner's side, while providing the desired expressiveness of access control policies. Second, Ins-PAbAC preserves users' privacy, relying on an anonymous authentication mechanism, derived from a privacy preserving attribute based signature scheme that hides the users' identifying information. Furthermore, our proposal introduces an accountable attribute based signature that enables an inspection authority to reveal the identity of the anonymously-authenticated user if needed. Third, Ins-PAbAC is provably secure, as it is resistant to both curious cloud providers and malicious users adversaries. Finally, experimental results, built upon OpenStack Swift testbed, point out the applicability of the proposed scheme in real world scenarios.**

Index Terms—**Cloud computing, cloud data sharing, privacy, attribute based encryption, attribute based signature, accountability, encrypted access control**

I. Introduction

Nowadays, the widespread adoption of cloud data storage applications raises several security and privacy issues. That is, ensuring the secrecy of outsourced data is considered as a major challenge for cloud clients, due to the loss of data control [1], [2], [3]. Thus, several solutions have proposed to apply data encryption at the data owner side, such that the decrypting keys are preserved out of reach of the cloud provider [4], [5]. Although data encryption ensures the confidentiality of data against malicious entities, the use of traditional encryption mechanisms *i.e.,* symmetric and asymmetric encryption is not sufficient to support fine-grained access control to outsourced data. On the one hand, access control policies need to be enforced to ensure flexible and distinguishable data sharing among users with different privileges. In addition, sharing data among dynamic groups of users requires an efficient distribution of deciphering keys between different authorized users. On the other hand, the involvement of the cloud provider makes the leakage of access patterns a major issue as it may disclose users' private information and even disclose confidential information about the outsourced data itself [6], [7]. Consequently, data confidentiality and users' privacy can not be ensured if these sensitive data are not protected.

The increasing need for fine-grained access control over outsourced data while preserving their secrecy led to the emergence of several encrypted access control schemes. Among these techniques, Attribute based Encryption (ABE) has appeared as a promising cryptographic technique which provides both confidentiality and fine grained access control to outsourced data [8]. ABE consists on encrypting data by the data owner w.r.t. a defined access policy over a set of attributes. Consequently, an authorised user who is able to decrypt and access data is an entity holding a set of attributes satisfying the access policy.

In this paper, we present Ins-PAbAC, an accountable privacy preserving attribute-based framework, for an authenticated encrypted access to data outsourced to cloud servers. The proposed framework combines Attribute Based Encryption (ABE) and Attribute Based Signature (ABS) mechanisms, while considering a two-level access control model. That is, it introduces (i) *fine-grained access control* supporting comprehensive granularity for access rules, and *anonymous data access*, allowing the storage server to manage access requests without any need to learn the user's identity neither his attributes.

On one hand, the use of ABE techniques allows the data owner to restrict access to his outsourced encrypted data to users that possess the required attributes. Accordingly, the data owner encrypts the data content w.r.t. an access structure, such that only users that have a set of attributes, satisfying the defined access policy, can access to data. Therefore, the data owner protects his data from unauthorised accesses including the cloud service provider (CSP) without need to share secret keys with users.

On the other hand, to avoid non-authorised users from downloading the data stored in the cloud, we apply an ABS scheme to verify the users' access rights. Indeed, the CSP asks the user to provide a signature w.r.t. the encryption access policy defined by the data owner. By verifying this signature, the CSP ensures that the ciphertext is only downloaded by an authorised user who is able to decrypt it.

The originality of Ins-PAbAC is multifold. First, the Ins-PAbAC framework proposes a privacy preserving authentication feature. That is, based on a novel design of a multi-authority attribute-based signatures, we ensure anonymous authentication for requesting data users where the user's identity remains protected against the cloud service provider. In addition, as the attribute based signature scheme is used to authenticate requesting users, the cloud provider is able to control the bandwidth consumption, thus, maintaining the system's availability. Obviously, the authentication of requesting users permits to mitigate *flooding attacks* [9] as only authenticated users can download encrypted data contents.

Second, as a decentralised multi-authority attribute based framework, Ins-PAbAC ensures that users' attributes are issued and managed by multiple authorities. Therefore, it reduces the bottleneck of considering one single central authority for managing secret parameters of all the system's users. In addition, unlike other multi-authority attribute based schemes which only support the issuance of one attribute per authority, Ins-PAbAC enables issuing a set of attributes from every attribute authority.

Third, thanks to the use of common public parameters and one single access policy for both encryption and signature algorithms, Ins-PAbAC is considered as highly scalable, providing an effective key management and offering interesting performances such as low storage and computation costs, at both the client and the cloud provider side.

Our Ins-PAbAC solution presented in this paper is an extension of the work that we published in [10]. As explained below, this extension details formal threat models and security analysis, emphasizes the support of multi-authority cloud storage systems and introduces a *proof-of-concept* of the proposed framework. The contributions of this work are as follows:

1) we extend the attribute-based signature scheme by adding an accountability feature. In fact, the accountability consists in allowing an inspection authority to reveal the identity of the signing user if needed. This added feature is suitable in real-life scenarios, where accountability and resources' misuse prevention are required.
2) the use of a multi-authority framework leads us to provide a mechanism for tying the user's key components together to prevent collusion attacks between users. As each key component may come from an attribute authority while there is no coordination between the different attribute authorities. Although its benefits in reducing key-escrow attacks, the use of multiple not-coordinated attribute authorities raises collusion attacks between users where they can combine their attributes to access outsourced data. To overcome these attacks, we extended the proposed scheme by applying Lewko et al.'s [11] technique for tying the user's key components together and preventing collusion attacks by applying a user identifier Id.
3) we provide formal system and security models for Ins-PAbAC framework. We discuss the resistance of Ins-PAbAC against two adversaries, relying on two different threat models. We prove that our proposed scheme satisfies the confidentiality, the unforgeability, the privacy and the anonymity removal requirements.
4) we evaluate the computational performances of our proposal, based on the OpenStack Storage system (Swift) testbed [12]. We conduct a number of experiments to measure the processing and communication costs as well as the impact of cryptographic algorithms' execution at both the user and cloud sides.

Paper Organisation – Section II presents security considerations and design goals. Then, Section III reviews related work and introduces both attribute based encryption and signature mechanisms. In Section IV, we describe Ins-PAbAC architecture, its system model as well as the related security models. Afterwards, we detail the Ins-PAbAC framework design and describe its different procedures in Section V. In Section VI, rigorous security discussions are given. Then, theoretical performance analysis is provided in Section VII. Finally, implementation results are introduced in Section VIII, before concluding in Section IX.

II. Problem Statement

Nowadays, cloud services are becoming the main architectural model for health organisations, thanks to the various advantages they provide to all the involved actors [13], [14]. For instance, cloud applications guarantee accessibility to an expanded range of access devices such as PCs, network of computers, smart-phones and network-enabled medical devices. Consequently, collaboration is made easier between health-care staff in order to provide accurate health-care services and avoid unnecessary redundant tasks. However, these distributed applications and decentralised environments raise data security and users' privacy preservation challenges, mainly with the myriad of laws and regulations aiming at protecting users' privacy and sensitive personal data, such as Health Insurance Portability and Accountability Act (HIPAA) [15] or the European General Data Protection Regulation (GDPR) [16]. For instance, HIPAA states that access policies have to be finely defined, such that authorized users may belong to several groups with different access privileges to outsourced data contents.

Let us consider a medical organisation that relies on cloud based services to collect and share Electronic Health Records (EHRs) among the medical staff, belonging to different organisations such as hospitals, research laboratories as well as health ministry. A health-care information system based on cloud services has to protect medical records from unauthorized access. For example, doctors need to share patients' health information and collaborate with the involved hospital employees to properly prescript treatments. As such, they usually form dynamic sharing groups with different granted privileges.

In addition, the system should protect the privacy of the users. Hence, the private identifying information of the involved cloud users, such as doctors and patients, must not be revealed to the cloud provider. For instance, the disclosure of access patterns may reveal privacy-sensitive information about patients and/or doctors and consequently may leak confidential information about the medical data content.

Furthermore, although ensuring anonymous access to medical data, designated authorities need to be able to reveal the identity of a user when required. For instance, if a unauthorised access or even modification of a medical record occurs, the user should be identified.

Thus, our proposed Ins-PAbAC framework has to fulfill the following properties:

- ***data confidentiality*** **–** Ins-PAbAC should ensure the secrecy of outsourced data contents against both curious cloud service providers and malicious users.
- ***fine-grained access control*** **–** Ins-PAbAC should ensure flexible security policies among several dynamic groups of users with different access rights, belonging to diverse groups.
- ***privacy*** **–** while requesting access to outsourced data contents, the proposed Ins-PAbAC framework has to protect users' access patterns. The cloud service provider must be able to grant access without knowing additional identifying information about the requesting users.
- ***accountability*** **–** preserving user's privacy should not lead to resources' misuse. This implies, for security purposes, that a designated authority should be able to recover the identity of the requesting entity.
- ***low processing cost*** **–** the design of Ins-PAbAC algorithms has to consider scalability features and devices' processing capabilities.

III. Attribute Based Cryptography

Several security solutions have been proposed in the literature to provide secure data sharing in cloud [1], [17], [18]. Indeed, cryptographic mechanisms have been usually applied to achieve fine grained access control for remote storage systems [19], [20], [21].

More specifically, encryption is applied at the client side in order to prevent unauthorized data disclosure to untrusted servers and unauthorized users. Hence, the decryption keys are only disclosed to the authorized users [1]. However, these techniques ensure confidentiality of outsourced data, key distribution remains complex, especially, while increasing the number of users. Indeed, relying on cloud environments, data owner may share data with users all over the world which makes distribution of the decryption keys very difficult and requires efficient and secure mechanisms.

Attribute Based Encryption (ABE) which was first introduced by Sahai and Waters [22] in 2005, provides an encrypted access control mechanisms over data. ABE relies on using attributes to define decryption keys which reduces the key management efforts.

In the following, we introduce Attribute Based Encryption mechanisms (ABE) and their applications in cloud environments in Section III-A. Then, we present Attribute Based Signature schemes (ABS) and we review ABS schemes use in cloud applications in Section III-B.

A. Attribute based Encryption (ABE)

The concept of Attribute Based Encryption (ABE) has been designed by Sahai and Waters in 2005, as a new technique to ensure encrypted access control to data [22]. ABE is differentiated from traditional encryption techniques as data are encrypted w.r.t. to an access policy over a set of attributes. A user is able to decrypt data if there is a match between his secret key, which is generated from his authenticated set of attributes, and the access policy associated with the ciphertext.

ABE mechanisms are categorised into two types, referred to as Key-Policy ABE (KP-ABE) [23] and Ciphertext-Policy ABE (CP-ABE) [8].

ABE schemes have been widely applied to secure outsourced data to distant cloud servers [24]. Although the proposed schemes achieves encrypted fine grained access control to data, they rely on a single authority to manage all the attributes used in the system and issue the related secret keys [25], [26], [27]. As such, the risk of key escrow attacks is arising, considering that all users' private keys are maintained by one single central entity.

Lewko and Waters [11] have proposed a decentralized attribute based encryption scheme, where users' secret keys are issued from different attribute authorities. Each attribute authority is in charge of deriving a private key associated to a user's attribute. This proposal do not require the use of a central trusted authority to issue secret keys which must remain active and uncorrupted throughout the lifetime of the system. Moreover, in order to prevent collusion in such a setting, this scheme requires that each user has a unique global identifier (*GID*). This identifier (*GID*) must be presented to each attribute authority to receive the attribute's secret key. However, in their proposal, Lewko and Waters [11] assume that each attribute authority is responsible for issuing only one attribute.

Huang et al. [28] introduced a hierarchical attribute-based encryption (HABE). In their proposal, a partial decryption and signing construction was introduced thanks to the application of a delegation mechanism consisting of delegating most of the computation overhead on the client side to the cloud service provider. Zhao et al. have proposed an attribute encryption scheme with non-monotonic access-structures [29]. This proposed scheme is designed to achieve fine grained access control in mobile health systems. However, [29] relies on a central trusted authority to manage the attributes and issue users' secret keys. Based on the decentralized attribute based encryption scheme proposed by Lewko *et al.* [11], Zhou et al. presented a multi-authority-attribute based encryption scheme for cloud data storage systems [30]. This scheme guarantees users' revocation relying on the attribute authority to re-issue secret keys to remaining non-revoked users. Wei *et al.* [31] have designed a decentralized cloud data sharing solution based on a multi-authority attribute based encryption scheme. Yang *et al.* [32] have proposed a similar data sharing solution for cloud computing which is based on multi-authority attribute based encryption. Recently, Sandor *et al.* [33] have proposed a multi-authority ABE scheme to secure data in mobile cloud data storage systems.

Finally, it is worth noticing that the reviewed techniques do not propose a mechanism to authenticate the requesting users. Moreover, most of the decentralized attribute based encryption schemes rely on the use of Lewko et al. decentralized ABE scheme [11] which assumes that each attribute authority is responsible for issuing only one attribute.

B. Attribute based Signature (ABS)

Attribute-Based Signature (ABS) has been derived from ABE mechanism to provide a fine grained and privacy preserving authentication of users. Similar to the concept of ABE, ABS consists in enabling a signer to sign a message only if his set of attributes satisfies a defined access policy. A verification entity can verify the signature's correctness without accessing the signer's identity neither knowing the exact attributes used to generate the signature [38]. In 2011, Maji et al. [38] introduced the first construction for a multi-authority attribute based signature scheme. This scheme consists of using multiple attribute authorities to manage attributes and issue related secret keys.

Several works rely on the attribute based signature to ensure data owners' authentication and fine grained access control over outsourced data to the cloud. Indeed, Zhao et al. [37] applied the ciphertext-policy attribute based encryption (CP-ABE) proposed by Bethencourt et al. [8] combined with the Maji et al. [38] attribute based signature to ensure fine grained access control to outsourced data in the cloud. This proposal is based on a centralized ABE and ABS schemes, thus it relies on a central trusted authority to issue secret keys to all the users.

Liu et al. [49] proposed a secure attribute based signature scheme where most computations required for signature generation procedure are outsourced to the CSP side. In this protocol, CSP is able to generate a half-signature using an outsourcing key received from the user. After receiving the half-signature, the user is able to transform it into a valid ABS signature. Rao et al. [54] have proposed a Key-Policy Attribute Based Signature (KP-ABS) scheme with a constant signature size. In their scheme, the signing key is associated with an access structure while the signature is generated using an attribute set satisfying the access structure. However, ciphertext-policy attribute based signature schemes (CP-ABS) are more appropriate to data sharing than KP-ABS since it enables the encrypting entity to generate an access tree over selected attributes. Ibrahim *et al.* [52] have proposed an attribute based authentication scheme. This scheme enables a cloud user to anonymously authenticate with the cloud server.

Most of aforementioned attribute based signature schemes have been proposed as an anonymous authentication mechanism in cloud sharing scenarios. Although anonymity is an important feature of attribute based signature mechanism, some malicious users may take advantage of this feature to escape from being traced. Hence, it is interesting to introduce the accountability feature in order to trace the signer's identity by an inspection authority when necessary.

Accountability in attribute based signatures was first introduced by Khader [34]. However, the proposed feature consists of revealing the attributes used to sign while only preserving the anonymity of the signer's identity.

El Kaafarani et al. [46] have proposed the first fully traceable decentralised attribute based signature. Subsequently, several traceable ABS schemes [55], [50] were proposed. In [55], Kaaniche and Laurent have proposed an anonymous certification (AC) mechanism built over a traceable attribute based signature (ABS). Their proposal provides a data minimization cryptographic scheme, permitting the user to reveal only required information to any service provider and ensures unlinkability between the different authentication sessions, while preserving the anonymity of the requesting user. Hong et al. [50] have introduced a key-policy attribute based signature (KP-ABS) scheme. This scheme deals with the assumption of including an untrusted authority. In this scheme, the signer's private key is composed of two parts: the first component is generated by attribute authority while the second part is chosen privately by the signer's. Moreover, this proposal introduces the accountability property. Recently, Cui *et al.* [53] applied ABS to secure communication in vehicular ad-hoc networks. This scheme ensures anonymous authentication in vehicular networks while supporting the accountability feature.

Table I summarizes the main differences between our proposed scheme Ins-PAbAC and the state of the art works. We compare several works based on different features such as multi-authority setting, accountability, security models, etc.

IV. Model Description

In this section, we first present our system architecture in Section IV-A and the system model in Section IV-B. Afterwards, we detail our security model in Section IV-C.

A. General Architecture

As depicted in Figure 1, our traceable Ins-PAbAC framework considers a cloud storage system that involves multiple authorities. The system model consists of the following five different entities:

- The Central Trusted Authority (CTA) is responsible for generating the global public parameters and the inspection authority's secret key. CTA is considered as a trusted entity in our model.
- The Attribute Authorities (AA) are a group of authorities responsible for managing attributes and issuing related secret keys. Unlike state of the art multi-authorities ABE schemes where each authority issues only one attribute, in Ins-PAbAC, each AA may manage a whole set of attributes. These authorities are considered as a trusted entities as they have access to users secret keys as well as their identities.
- The Inspection Authority (IA) is an independent authority able to revoke the anonymity of a malicious user when an attack occurs. IA is considered as a trusted entity.
- The Cloud Service Provider (CSP) is a remote cloud server who stores and shares data among authorised users. CSP is also responsible for authenticating users before downloading data. CSP is a honest but curious entity,

TABLE I: Features and Functionality Comparison of Attribute Based Encryption and Signature Schemes

Scheme	*Type*	*Access policy*	*Mutli-authority*	*Accountability*	*End User Authentication*	*Security Models*
Sahai et al. (2005) [22]	Fuzzy	Threshold	✗	✗	✗	IND-CPA
Bethencourt et al. (2007) [8]	CP-ABE	Monotone	✗	✗	✗	IND-CPA
Khader et al. (2007) [34]	KP-ABS	Monotone	✗	✗	✓	UF-CMA
Maji et al. (2008) [35]	CP-ABS	Monotone	✗	✗	✗	UF-CMA
Li et al. (2010) [36]	CP-ABS	Threshold	✓	✗	✓	IND-CPA
Zhao et al. (2011) [37]	CP-ABE + CP-ABS	Monotone	✗	✗	✗	IND-CPA + UF-CMA
Maji et al. (2011) [38]	CP-ABS	Monotone	✓	✗	✓	UF-CMA
Wang et al. (2010) [39]	CP-ABE	Monotone	Herarchical	✗	✗	CPA-Security
Yu et al. (2010) [40]	KP-ABE	Monotone	✓	✓	✓	CPA-Security
Waters (2011) [41]	CP-ABE	Monotone	✗	✗	✗	CPA-Security
Lewko et al. (2011) [11]	CP-ABE	Monotone	✓	✗	✗	CPA-Security
Ruj et al. (2011) [42]	CP-ABE	Monotone	✓	✗	✗	CPA-Security
Ruj et al. (2012) [43]	CP-ABE + CP-ABS	Monotone	✓	✗	✗	IND-CPA + UF-CMA
Li et al. (2013) [44]	KP-ABE	Monotone	✓	✗	✗	IND-CPA
Yang et al. (2014) [45]	CP-ABE	Monotone	✓	✗	✗	IND-CPA
El Kaafarani et al. (2014) [46]	CP-ABS	Monotone	✓	✓	✓	UF-CMA
Ruj et al. (2014) [47]	CP-ABS	Monotone	✓	✗	✗	IND-CPA + UF-CMA
Horvath et al.(2015) [48]	CP-ABE	Monotone	✓	✗	✗	IND-CPA
Liu et al. (2015) [49]	CP-ABS	Monotone	✗	✗	✗	UF-CMA
Rao et al. (2016) [49]	KP-ABS	Monotone	✗	✗	✗	UF-CMA
Kaaniche et al. (2016)	CP-ABS	Monotone	✗	✗	✓	UF-CMA
Hong et al. (2016) [50]	KP-ABS	Monotone	✗	✗	✓	UF-CMA
Belguith et al. [10] (2016)	CP-ABE + CP-ABS	Monotone	✓	✗	✓	IND-CPA + UF-CMA
Huang et al. (2017) [28]	CP-ABE + CP-ABS	Monotone	✗	✗	✓	IND-CPA + UF-CMA
Belguith et al. (2018) [51]	CP-ABE	Monotone	✓	✗	✗	IND-CPA
Wei et al. (2018) [31]	CP-ABE	Monotone	✓	✗	✗	IND-CPA
Yang et al. (2018) [32]	CP-ABE	Monotone	✓	✗	✗	IND-CPA
Ibrahim *et al.* (2018) [52]	CP-ABS	Monotone	✗	✗	✗	UF-CMA
Sandor *et al.* (2019) [33]	CP-ABE	Monotone	✓	✗	✗	IND-CPA
Cui *et al.* (2019) [53]	CP-ABS	Threshold	✗	✓	✓	UF-CMA
Ins-PAbAC	CP-ABE + CP-ABS	Monotone	✓	✓	✓	IND-CPA + UF-CMA

it executes the protocol properly but it tries to gain knowledge about data and their users.

- The data owner (O) is the data producer. He defines access rights and encrypts data with respect to them before outsourcing to the cloud.
- The data user (U) requests access to data and authenticates with the CSP. He decrypts the received data using his access rights. A user may be malicious if she tries to access data without authorisation.

B. System Model

Our Ins-PAbAC proposal is defined upon the following nine algorithms. It involves four procedures on the basis of three phases. During the first phase, the system initialisation procedure SYS_INIT is executed. The second phase occurs when the data owner wants to share data files with other cloud users, based on both the data storage procedure STORE and the data retrieval procedure BACKUP. While the first and second phases are mandatory for outsourcing data files to the cloud servers, the third phase occurs when there is a need for user's anonymity revocation, relying on the execution of the inspection procedure INSPEC.

The SYS_INIT procedure consists of three randomized algorithms for the generation of public parameters referred to as `setup`, the generation of the private and public parameters related to the involved attribute authorities denoted by $\texttt{setup}_{\texttt{auth}}$, and the `keygen` algorithm to generate the users' private keys. The STORE procedure consists of the `encdata` algorithm used to encrypt data files in order to perform the data storage scenario. For data retrieval, the BACKUP procedure deals with the user' authentication, namely `sign` and `verif` algorithms and the data decryption algorithm referred to as `decdata`. The inspection procedure occurs when there is a need to reveal the identity of the user. It consists of `trace` and `judge` algorithms.

- SYS_INIT procedure
  $\texttt{setup}(\lambda) \rightarrow \texttt{PP}$ – the setup algorithm is performed by the central trusted authority (CTA). This randomized algorithm takes as input the security parameter λ and outputs the global public parameters PP and the the inspection authority's secret key sk_{ins}.

  $\texttt{setup}_{\texttt{auth}}(\texttt{PP})$ – this randomized algorithm is executed by an attribute authority AA_j. It takes as inputs the public parameters PP and outputs the pair of the attribute authority's private and public keys (sk_{AA_j}, pk_{AA_j}).

  $\texttt{keygen}(\texttt{PP}, sk_{AA_j}, pk_{AA_j}, Id, S_j) \rightarrow sk_{S_j}$ – this algorithm is performed by an attribute authority AA_j in order to generate the user's secret key related to a set of attributes $S_j = \{a_{1_j}, \cdots, a_{n_j}\}$, where n_j is the number of attributes of S_j. It takes as input the global parameters PP, the pair of private and public attribute authority's keys (sk_{AA_j}, pk_{AA_j}) and the users' identity Id. It outputs the secret key sk_{S_j} related to the set of attributes S_j.

- STORE procedure
  $\texttt{encdata}(\texttt{PP}, \{pk_{AA_j}\}, D_F, (A, \rho)) \rightarrow E_D$ – the `encdata` algorithm is performed by the data owner O. It takes as input the global public parameters PP, the set of

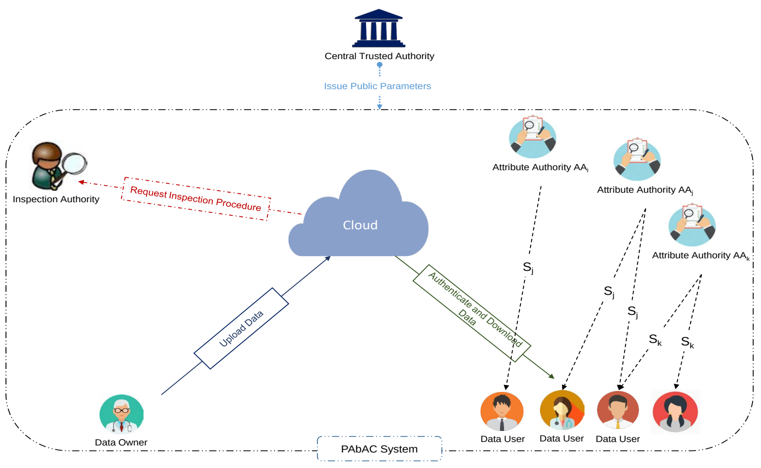
Central Trusted Authority
Issue Public Parameters
Inspection Authority
Request Inspection Procedure
Cloud
Attribute Authority AA$_i$
Attribute Authority AA$_j$
Attribute Authority AA$_k$
S_j
S_j
S_j
S_k
S_k
Upload Data
Authenticate and Download Data
Data Owner
Data User
Data User
Data User
PAbAC System

Fig. 1: Ins-PAbAC Network Model: Main Entities, Phases and Procedures

involved attribute authorities' public keys $\{pk_{AA_j}\}$, the data file D_F and the access policy (A,ρ). The encryption algorithm outputs a ciphertext denoted by E_D.

- BACKUP procedure
  $\mathtt{sign}(\mathtt{PP}, m, (A,\rho), sk_{S_j}) \rightarrow \Sigma$ – this algorithm is run by the requesting user U. Given the public parameters PP, a random token m, the encryption access policy (A,ρ) and the user's secret key sk_{S_j} related to the set of attributes S_j that satisfies the access policy, the signing algorithm outputs a signature Σ.

  $\mathtt{verif}(\mathtt{PP}, \{pk_{AA_j}\}, m, \Sigma, (A,\rho)) \rightarrow V$ – this deterministic algorithm fulfilled by the cloud provider CSP takes as input the global public parameters PP, a set of the involved attribute authorities' public keys $\{pk_{AA_j}\}$, a random token m, a signature Σ and the encryption access policy (A,ρ). It outputs a boolean value V, such that $V = \mathtt{true}$ if Σ is valid and $V = \mathtt{false}$ otherwise.

  $\mathtt{decdata}(\mathtt{PP}, \{sk_{S_j}\}, (A,\rho), E_D) \rightarrow D_F$ – the user U executes the decryption algorithm to retrieve the original data file D_F. This algorithm takes as input the public parameters PP, the user secret decryption key $\{sk_{S_j}\}$, the encryption access policy (A,ρ) and the ciphertext E_D and outputs the original data file D_F.

- INSPEC procedure
  $\mathtt{trace}(\mathtt{PP}, s_{ins}, \{pk_{AA_j}\}, \sigma, (A,\rho)) \rightarrow (sid, \pi)$ – this deterministic algorithm is performed by an inspection authority. It takes as input the global public parameters PP, a set of the involved attribute authorities' public keys $\{pk_{AA_j}\}$, the inspection authority's secret key sk_{ins}, a signature Σ and an access policy (A,ρ), and outputs the identity of the signer sid and a proof π attesting to this claim. If the algorithm is unable to trace the signature to a signer, it returns an error message.Note that sid is an identity of the user generated over the attributes authorities public keys and his global identity Id. This value is stored in a matching table at the inspection authority side to retrieve a malicious user's identity while executing the inspection procedure.

  $\mathtt{judge}(\mathtt{PP}, \{pk_{AA_j}\}, \Sigma, (A,\rho), sid, \pi) \rightarrow (0,1)$– the judge is a deterministic algorithm which takes as input a the global public parameters PP, a set of the involved attribute authorities' public keys $\{pk_{AA_j}\}$, a signature Σ, an access policy (A,ρ), a signer identity sid, and a tracing proof π , and outputs 1 if π is a valid proof that sid has produced σ or 0 otherwise.

Our traceable Ins-PAbAC scheme has to satisfy the correctness property. The correctness property requires that for all security parameter λ, all public parameters $\mathtt{PP} \in \mathtt{setup}(\lambda)$, all $(sk_{AA_j}, pk_{AA_j}) \in \mathtt{setup}_{\mathtt{auth}}(\mathtt{PP})$, all secret keys $sk_{S_j} \in \mathtt{keygen}(\mathtt{PP}, \{sk_{AA_j}\}, Id, S_j)$, all $(A,\rho) \in \mathcal{G}$ where $\mathcal{G}$ is the access structure space, If the user has a set of attributes S_j satisfying the access policy (A,ρ), for all generated signature $\Sigma \in \mathtt{sign}(\mathtt{PP}, m, (A,\rho), sk_{S_j})$, the verification algorithm $\mathtt{verif}(\mathtt{PP}, \{pk_{AA_j}\}, m, \Sigma, (A,\rho))$ outputs 1.

In addition, for all $E_D \in \mathtt{encdata}(\mathtt{PP}, \{pk_{AA_j}\}, D_F, (A,\rho))$, if the user has successfully obtained the secret key sk_{S_j} related to the required attributes for deciphering the encrypted file and has successfully signed the received message m from the CSP, the $\mathtt{decdata}(\mathtt{PP}, sk_{S_j}, (A,\rho), E_D)$ outputs D_F. Indeed, for all $(sid, \pi) \in \mathtt{trace}(\mathtt{PP}, s_{ins}, \{pk_{AA_j}\}, \sigma, (A,\rho))$, $\mathtt{judge}(\mathtt{PP}, \{pk_{AA_j}\}, \Sigma, (A,\rho), sid, \pi)$ outputs 1.

C. Security Model

In this section, we consider two adversaries for proving the security and privacy properties of our Ins-PAbAC scheme. First, we point out the case of a *honest but curious* cloud provider. In fact, the cloud server is considered as honest as it provides accurate inputs or outputs, while properly executing any calculations expected from it. However, it is curious as it attempts to gain extra knowledge from the protocol. Consequently, we consider the honest but curious adversary against both the data confidentiality requirement considering adaptive chosen plaintext attacks (IND-CPA) and the computational privacy requirements.

Second, we consider the case of malicious users trying to

override their rights. That is, malicious users may attempt to deviate from the protocol or to provide invalid inputs. As such, we consider the malicious adversary mainly against the unforgeability requirement considering adaptive chosen messages attacks (UF-CMA).
Third, we consider the case of a malicious user trying to forge a signature that is recognised by the tracing authority. In other words, this malicious user tries to produce a signature while the inspection authority traces another signing user. Therefore, we consider the adversary against the anonymity removal requirement.

1) Indistinguishability: In our security model, we assume that the adversary is allowed to query for any secret keys that cannot be used for decrypting the challenge ciphertext. Moreover, we consider the assumption introduced in Lewko et al. proposal [11] where the adversary can only corrupt authorities statically. As considered by all the multi authorities ABE schemes such as Chase et al. [56] and Lewko et al. [11], we consider the static corruption model as the model where the adversary can only corrupt authorities statically before the game starts and not after. For this purpose, we define Exp^{ind}, a security game between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$. The adversary tries to decipher a signcrypted message, i.e; to distinguish between two randomly generated encrypted messages, without having sufficient deciphering attributes.
In their standard definition, multi-authority ABE schemes are proved to be secure based on Chosen Plaintext Attack (CPA) security games [11], [56], [41]. Indeed, CPA is a security game that captures the resistance of a public key encryption scheme against adversaries. In CPA, the adversary has access to a key generation oracle, however, this access is limited until the challenge ciphertext is known. For some other ABE schemes, a stronger security game can be considered which is Chosen Ciphertext Attack (CCA). In CCA, the adversary has, in addition, access to a decryption oracle, however, this access is also limited until the challenge ciphertext is known. This requirement derives the non-malleability feature which consists that the adversary cannot output a second ciphertext so that the corresponding plaintexts are meaningfully related. As for Ins-PAbAC, we consider the CPA-security game as the scheme is based on Lewko et al. which is proved to be CPA-secure in the standard model. Further details about these security notions can be found in [57].

Let consider S_{AA} the set of all attribute authorities and S'_{AA} a set of corrupted attributes authorities.

Our Ins-PAbAC scheme is secure against static corruption of the attribute authorities if there is no probabilistic polynomial time (PPT) adversary that can win the Exp^{ind} security game defined below with a non-negligible advantage.

The Exp^{ind} security game is formally defined, between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$, as follows:

Initialisation – in this phase, the adversary $\mathcal{A}$ chooses a challenge access structure $\Psi^* = (A^*, \rho^*)$ and sends it to the challenger $\mathcal{C}$.

Setup – in this phase, the challenger $\mathcal{C}$ runs the `setup` algorithm to generate the public parameters.

Then, the adversary $\mathcal{A}$ selects a set of corrupted attributes authorities $S'_{AA} \subset S_{AA}$ and runs the $\texttt{setup}_{\texttt{auth}}$ algorithm to obtain their public and private keys.

Then, $\mathcal{C}$ queries the honest attribute authorities' public and private keys by running the $\texttt{setup}_{\texttt{auth}}$ algorithm. Afterwards, the challenger $\mathcal{C}$ publishes the public keys of the honest attributes authorities.

Queries phase – in this phase, for each session k, the adversary queries the secret keys related to a set of attributes $\{S_j\}_k$ belonging to a set of non-corrupted attributes authorities $a_i \in S_{AA} \setminus S'_{AA}$. Then, the challenger returns the corresponding secret keys to the adversary. Note that the set of attributes $\{S_j\}_k$ does not satisfy the access policy $\Psi^* = (A^*, \rho^*)$ i.e; $\Psi^*(\{S_j\}_k) \neq 1$.

Challenge – during the challenge phase, the adversary chooses two equal length plaintexts M_0 and M_1 and sends them to the challenger. The challenger $\mathcal{C}$ chooses a random bit b such that $b \in \{0,1\}$ and encrypts M_b under the access structure (A^*, ρ^*). The generated ciphertext E_b is then returned to the adversary.

Queries phase 2 – in this phase, the adversary $\mathcal{A}$ who has already received M_b, can query a polynomially bounded number of queries as in **Queries Phase 1**, except that the adversary $\mathcal{A}$ can not query secret keys related to a set of attributes which satisfy the access policy $\Psi^* = (A^*, \rho^*)$.

Guess – the adversary tries to guess which message $M_{b'}$ where $b' \in \{0,1\}$ corresponds to the challenge ciphertext E_b. The advantage of the adversary to win the game is defined as:

$$Adv_{\mathcal{A}}[Exp^{Conf}(1^{\xi})] = |Pr[b = b'] - \frac{1}{2}|$$

Definition 1. *Ins-PAbAC is CPA-secure (i.e., chosen plaintext attack (CPA) secure) against static corruption of the attribute authorities if the advantage $Adv_{\mathcal{A}}[Exp^{Conf}(1^{\xi})]$ is negligible for all PPT adversaries.*

2) Unforgeability: As considered by all the multi authorities ABS schemes such as Maji et al. [38] and El Kharafani et al. [46], we consider the static corruption model as the model where the adversary can only corrupt authorities statically before the game starts and not after. Let consider S_{AA} the set of all attribute authorities and S'_{AA} a set of corrupted attributes authorities.

Our Ins-PAbAC scheme is unforgeable against chosen message attack (UF-CMA) if there is no probabilistic polynomial time (PPT) adversary that can win the Exp^{unf} security game with non-negligible advantage. The Exp^{unf} security game is formally defined, between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$ as follows:

Initialisation – in this phase, the adversary $\mathcal{A}$ chooses a challenge access structure (A^*, ρ^*) and sends it to the challenger $\mathcal{C}$.

Setup – the challenger $\mathcal{C}$ chooses a security parameter λ and generates the public parameters by running the `setup` algorithm. Then, $\mathcal{C}$ sends the public parameters to the adversary $\mathcal{A}$. Moreover, the challenger runs the $\texttt{setup}_{\texttt{auth}}$ for non corrupted authorities and publishes the related public keys. Then, the adversary $\mathcal{A}$ selects a set of corrupted attributes authorities $S'_{AA} \subset S_{AA}$ and runs the $\texttt{setup}_{\texttt{auth}}$ algorithm to obtain their public and private keys.

Queries phase – in this phase, for each session k, the adversary $\mathcal{A}$ can query secret keys related to the set of attributes $\{S_j\}_k$ belonging to a set of non-corrupted authorities $S'_{AA} \setminus S_{AA}$. The challenger $\mathcal{C}$ returns the corresponding secret keys to the adversary $\mathcal{A}$.

Moreover, the adversary $\mathcal{A}$ can query the signature for any message m and for any access structure (A,ρ).

Note that the adversary $\mathcal{A}$ can not query secret keys related to a set of attributes which satisfy the challenge access structure (A^*,ρ^*).

Forgery phase – the adversary generates a signature Σ^* on a message m^* selected by the challenger, with respect to the challenge access structure (A^*,ρ^*). The adversary wins the Exp^{unf} game if Σ^* is a valid signature on the access structure (A^*,ρ^*) where this latter was not queried before and the combination of the adversary's attributes with the attributes related to the corrupted attribute authorities does not satisfy the access structure.

Hence, the adversary's advantage is defined as follows:

$$Adv_{\mathcal{A}}[Exp^{unf}(1^{\xi})] = |Pr[Exp^{unf}(1^{\xi})] = 1|$$

Definition 2. *Ins-PAbAC scheme is unforgeable against chosen-message attack (UF-CMA), if the advantage $Adv_{\mathcal{A}}[Exp^{unf}(1^{\xi})]$ is negligible for all PPT adversaries.*

This unforgeability property also includes the collusion among users trying to override their rights by combining their complementary attributes to generate a signature satisfying a given access structure. It also covers the non frameability case when a user also aims to override his rights but on his own.

3) Privacy: Ins-PAbAC is a privacy preserving mechanism if it relies on the computational private signature scheme. Indeed, a signature scheme is said computational private if there is no adversary that can pinpoint which set of attributes is used for the signature generation while given an unbounded computational power.

We require that all the attribute authorities are honest as knowing the $\mathcal{H}(Id)$ of a user. In this game, we are interested to prove that our scheme is private against an honest but curious CSP.

Our Ins-PAbAC scheme is said to be computationally private if any adversary $\mathcal{A}$ running in polynomial time cannot win the $Exp_{\mathcal{A}}{}^{Priv}$ security game with non-negligible advantage.

The $Exp_{\mathcal{A}}{}^{Priv}$ security game is formally defined, between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$ as follows:

Setup – the adversary $\mathcal{A}$ requests from the challenger $\mathcal{C}$ the global public parameters as well as the public authorities' keys. Then, the challenger $\mathcal{C}$ runs the `setup` algorithm and the $\texttt{setup}_{\texttt{auth}}$ algorithms. Afterwards, $\mathcal{C}$ returns the global public parameters PP and the set of the public authorities' keys $\{pk_{AA_j}\}$ to the adversary $\mathcal{A}$.

Challenge phase – the adversary $\mathcal{A}$ chooses an access structure $\psi = (A,\rho)$, two sets of attributes S_{j_1} and S_{j_2} satisfying the monotone access structure $\psi = (A,\rho)$ such that $\psi(S_{j_1}) = \psi(S_{j_2}) = 1$ and a message m and sends them to the challenger $\mathcal{C}$.

Afterwards, the challenger $\mathcal{C}$ picks a random bit $b \in \{1,2\}$ and executes the `KeyGen` algorithm such that $sk_{S_{j_b}} = \texttt{Keygen}(\texttt{PP}, \{sk_{AA_j}\}, Id_{\mathcal{C}}, S_{j_b})$ and outputs the signature $\Sigma_b = \texttt{sign}(\texttt{PP}, m, (A,\rho), sk_{S_{j_b}})$. The signature Σ_b is sent to the adversary $\mathcal{A}$ as a challenge signature.

Guess – the adversary $\mathcal{A}$ tries to guess which set of attributes S_{j_1} or S_{j_2} was used to generate the signature Σ_b. $\mathcal{A}$ outputs a bit b' and wins the game if $b' = b$.

The advantage of the adversary $\mathcal{A}$ in the above game is defined as follows:

$$Adv_{\mathcal{A}}[Exp^{Priv}(1^{\xi})] = |Pr[b = b'] - \frac{1}{2}|$$

Definition 3. *Our accountable Ins-PAbAC satisfies the computational privacy property if for any two attribute sets S_{j_1} and S_{j_2}, any message m, any signature Σ with respect to any access structure $\psi = (A,\rho)$ where $\psi(S_{j_1}) = \psi(S_{j_2}) = 1$, any adversary $\mathcal{A}$ cannot distinguish which attribute set S_{j_1} or S_{j_2} is used to generate the signature Σ, i.e; $Adv_{\mathcal{A}}[Exp^{Priv}(1^{\xi})]$ is negligible with respect to the security parameter λ.*

4) Anonymity Removal: The anonymity removal requirement ensures that there is no adversary that can produce a signature while the inspection authority traces another signing user. In addition, it also requires that there is no adversary that can generate a non-traceable signature which has been successfully verified by the service provider. Thus, the anonymity removal requirement covers the untraceable forgery attacks.

In the anonymity removal game, the adversary is allowed to corrupt the tracing authority and ask for the signing keys of any signer to be revealed. However, unlike the full unforgeability game presented in Section VI-C, we require that all the attribute authorities are honest as knowing the secret key of any attribute authority makes it easy to create untraceable signatures by malicious users.

Our Ins-PAbAC scheme is secure under anonymity removal attack if there is no probabilistic polynomial time (PPT) adversary that can win the Exp^{AnR} security game with a non-negligible advantage.

The Exp^{AnR} security game is formally defined, between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$ as follows:

Initialisation – in this phase, the adversary $\mathcal{A}$ chooses a challenge access structure (A^*,ρ^*) and sends it to the challenger $\mathcal{C}$.

Setup – the challenger chooses a security parameter λ and generates the public parameters by running the `setup` algorithm. The challenger sends the public parameters to the adversary. Moreover, the challenger runs the $\texttt{setup}_{\texttt{auth}}$ for non corrupted authorities and publishes the related public keys.

Queries phase – in this phase, for each session k, the adversary $\mathcal{A}$ can query secret keys related to the set of attributes $\{S_j\}_k$ belonging to non-corrupted authorities. The challenger $\mathcal{C}$ returns the corresponding secret keys to the adversary. Moreover, $\mathcal{A}$ can query the signature for any message m and for any access structure (A,ρ).

Output – the adversary generates a signature Σ^* on a message m^* selected by the challenger, with respect to the challenge access structure (A^*,ρ^*). The adversary wins the Exp^{AnR} game if Σ^* is untraceable by the challenger $\mathcal{C}$, relying on the `trace` algorithm.

The advantage of the adversary $\mathcal{A}$ in the above game is defined as:

$$Adv_{\mathcal{A}}[Exp^{AnR}(1^{\xi})] = |Pr[Exp^{AnR}(1^{\xi})] = 1|$$

Definition 4. *Ins-PAbAC scheme is secure against anonymity removal attacks, if the advantage $Adv_{\mathcal{A}}[Exp^{AnR}(1^{\xi})]$ is negligible for all PPT adversaries.*

V. Ins-PAbAC: an accountable Privacy preserving Attribute based framework for fine grained Access Control in clouds

A. Motivation

In order to achieve fine grained and privacy preserving access control to outsourced data in cloud storage, we combine two cryptographic techniques, ciphertext-policy attribute based encryption (CP-ABE) and attribute based signature (ABS). Moreover, ABS ensures the accountability feature which is necessary for imputing malicious acts to individuals. Attribute based encryption is appropriate for cloud data sharing thanks to its functional properties. First, attribute based techniques do not require certificates to certify the users' public keys as in the traditional public key cryptography, thus leading to an easier key management system. Second, the public keys are derived with no need for previous computation of corresponding private keys. For instance, in attribute based cryptography, there are no requirements to generate the private key before the public key unlike traditional public key derivation schemes. Third, CP-ABE encryption enables the enciphering entity to only exploit public parameters and generate the access structure to encrypt data before outsourcing the data files to cloud servers. Hence, this technique is much more appropriate for data outsourcing and for one-to-many communications than other traditional encryption techniques. Fourth, the enciphering entity does not know the entities who are able to access data, thus providing strong privacy feature. Fifth, CP-ABE specifies fine-grained access rights for each individual user through the assignment of a set of attributes. Only users who hold attributes satisfying the access tree of the encrypted data can decrypt them. Thereofore, CP-ABE is considered as one of the most appropriate public key primitive for one-to-many communications, supporting efficient key management, access control and encryption.

Furthermore, Attribute Based Signature (ABS) is of interest for supporting shared access control, as presented in our Ins-PAbAC framework. In ABS, messages are signed w.r.t. an access structure, and the CSP has to verify that the requesting user has a set of attributes that satisfies the access structure. As such, additionally to being certificate-less, having separate generation of public and private keys, and managing access rights based on attributes assignment, likely to CP-ABE, ABS authenticates the user without knowing his identity or the set of attributes used in the signing procedure.

In our accountable Ins-PAbAC framework, we design an extension of the centralized attribute based encryption scheme introduced by Waters [41] to achieve a multi-authority attribute based encryption scheme based on an adaptation of Lewko et al. multi-authority ABE technique. In addition, we design a multi-authority attribute based signature scheme based on an original use of Waters' identity based scheme [58]. Indeed, instead of defining a user by his identity, we change the scheme to define a user using a set of attributes. We ensure, by this design, an enhanced privacy preservation of the user's identifying information. Furthermore, we extend the designed scheme to a multi-authority setting. For instance, the attributes used in the signature are issued by different authorities to avoid key escrow attacks led against a central authority. Unlike other multi-authority attribute based techniques [47], [11], the proposed encryption and signature schemes presented by Ins-PAbAC support the issuance of a set of attributes obtained from the same authority. This feature enables saving the communication and computation costs of the proposed access control scheme.

Furthermore, the data owner defines only one access policy to be used in both the encryption and authentication phases. To authenticate data users when requesting access to an outsourced data file, CSP first asks the user to sign a random message w.r.t. the access policy associated to the requested ciphertext and uploaded by the data owner. This feature allows the CSP to restrict the ciphertext's retrieval to only users who are able to decrypt it. In other words, if a user was able to sign the message w.r.t. the encryption access policy, then he proved that he holds the set of attributes satisfying the encryption access policy and he will be able to decrypt the ciphertext.

To remove anonymity of users which is required by the CSP in case of malicious acts, we design a new accountability mechanism, working as follows. After an attack is detected, the CSP who has previously registered the user's signed token can forward to the inspection authority the user's signature along with the used access policy and requests for anonymity removal. Then, the inspection authority can use the user's signature to retrieve *sid* and then reveal the user's identity. Therefore, the Ins-PAbAC signature scheme is extended to support the accountability property, that is the capability for an inspection authority to trace the identity of anonymous users.

For ease of presentation, the different notations used in this paper are listed in Table II.

TABLE II: The different notations used in this paper

Notation	Description
PP	Public parameters
sk_{AA_j}	Private key of an attribute authority AA_j
pk_{AA_j}	Public key of an attribute authority AA_j
S_U	Set of users' attributes
S_j	Set of attributes certified by the attribute authority AA_j
Sk_{S_j}	Secret keys related to the set of attributes S_j obtained from the attribute authority AA_j
AA	Attribute Authority
D_F	Data file
O	Data Owner
U	User
E_D	Encrypted data file
m	Authentication message to be signed by a user
Σ	Signature over a message m
Ψ	Access policy
s_{ins}	Secret key of the inspection authority
Id	User's identity used to tie his secret keys
Ω	Number of attribute authorities
sid	An identity of user managed by the inspection authority
n	Number of attributes in the access policy

B. Overview

In Ins-PAbAC, the data owner O first defines an access structure ψ that points out who can access the outsourced data with respect to a set of attributes. Then, the data file is encrypted under the access structure ψ, based on an attribute based encryption algorithm. Subsequently, the data owner stores the encrypted data in the cloud. When a user U wants to access the outsourced data file, he has first to authenticate with the cloud. For this purpose, he has to sign a random message, obtained from the cloud, under the access structure ψ associated with the outsourced data file. Afterwards, the cloud verifies the correctness of the received signature in order to send the requested elements, namely the encrypted data file. Furthermore, if a malicious user forwards a forged signature to the cloud service provider, this latter can request that this user is traced and his identity is revealed by an independent trusted inspection authority. Based on the required attributes, specified in the access structure ψ, the requesting user uses his private keys in order to decrypt the encrypted data file.

Our Ins-PAbAC construction relies on the following complexity assumptions:

Definition 5. ***Computational Diffie Hellman problem (CDH)***

Given a generator g of a multiplicative cyclic group $\mathbb{G}$ of order N and given two group elements $g^a \in \mathbb{G}$ and $g^b \in \mathbb{G}$ where $a,b \in \mathbb{Z}_N$ are two secrets, the problem of calculating g^{ab} from g^a and g^b is called the Computational Diffie Hellman problem.

Definition 6. ***Decisional Bilinear Diffie-Hellman Assumption (DBDH)***

Given a generator g of a multiplicative cyclic group $\mathbb{G}$ of order N and given three group elements $g^a \in \mathbb{G}$, $g^b \in \mathbb{G}$ and $g^c \in \mathbb{G}$ where $a,b,c \in \mathbb{Z}_N^*$ are three secrets, the problem of distinguishing between tuples of the form $(g^a, g^b, g^c, \hat{e}(g,g)^{abc})$ and $(g^a, g^b, g^c, \hat{e}(g,g)^z)$ for some random integer z, is called the Decisional Bilinear Diffie-Hellman Assumption (DBDH).

C. Construction

In this section, we review the procedures and algorithms of our traceable Ins-PAbAC construction.

1) System Initialisation Procedure – SYS_INIT*:* The SYS_INIT procedure consists of three randomized algorithms, defined as follows:

- `setup` – the CTA defines two multiplicative groups $\mathbb{G}_1$ and $\mathbb{G}_T$ of order N, a bilinear map $\hat{e}: \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_T$ and a collision resistant hash function $\mathcal{H}: \{0,1\}^* \to \mathbb{Z}_N$. In addition, it chooses two generators g,h of $\mathbb{G}_1$ such that $h = g^{s_{ins}}$ where s_{ins} is the inspection authority's private key. Then, the CTA picks at random a set of random values $\{r_i\}_{i\in[1,q]}$, such that $r_i \in \mathbb{Z}_N$, $u_i = g^{r^i}$ and $\{u_0 = g, \cdots, u_q\}$ are generators of $\mathbb{G}_1$. Finally, it outputs the global public parameters PP defined as follows:

$$\mathsf{PP} = \{\mathbb{G}_1, \mathbb{G}_T, N, \mathcal{H}, h, \hat{e}, u_0, \cdots, u_q\}$$

- $\texttt{setup}_{\texttt{auth}}$ – each attribute authority AA_j generates the pair of its private and public authority's keys (sk_{AA_j}, pk_{AA_j}) defined as follows:

$$sk_{AA_j} = (\alpha_j, t_j)$$
$$pk_{AA_j} = (\hat{e}(g,g)^{\alpha_j}, g^{t_j}) = (\hat{pk_{AA_j}}, \check{pk_{AA_j}})$$

Where $\alpha_j, t_j \in \mathbb{Z}_N$ are random values.

- `keygen` – for any user U having a set of attributes $S_j = \{a_{1_j}, \cdots, a_{n_j}\}$ where n_j is the number of attributes of S_j, the attribute authority AA_j chooses a random value $f_{Uj} \in \mathbb{Z}_N$. Then, it computes the secret key sk_{S_j} as depicted by Algorithm 1.

Algorithm 1 `keygen` procedure

```
1: Input: the global parameters PP, the pair of private and public attribute authority's keys $(sk_{AA_j}, pk_{AA_j})$, a set of attributes $S_j$ and the users' identity $Id$.
2: Output: the secret key $sk_{S_j}$ related to the set of attributes $S_j$
3: $M_j \leftarrow g^{\alpha_j} h^{t_j} h^{f_{Uj}} h^{t_j \mathcal{H}(Id)}$;
4: $sk_{S_j} \leftarrow \{M_j\}$;
5: $L_j \leftarrow g^{t_j} g^{f_{Uj}}$;
6: $sk_{S_j} \leftarrow sk_{S_j} \cup \{L_j\}$;
7: for all $i \in [1 \ldots n_j]$ do
8:     $K_{i,j} \leftarrow {u_i}^{t_j} {u_i}^{f_{Uj}}$;
9:     $sk_{S_j} \leftarrow sk_{S_j} \cup \{K_{i,j}\}$;
10: end for
11: return $sk_{S_j}$
```

2) Data Storage Procedure – STORE*:* To outsource a data file (D_F) to the cloud, the data owner O performs the STORE procedure. For this purpose, he first defines an access policy ψ and obviously selects the attribute needed to satisfy it. We note that the access policy ψ is described in terms of a monotonic boolean formula. We represent the boolean formula as an access tree where the interior nodes are AND and OR gates, and the leaf nodes correspond to attributes [10]. Thus, the access policy corresponds to the couple (A, ρ) where A is an $n \times l$ access matrix and ρ is the function that maps the matrix rows to the required attributes. These attributes have to be obtained from a set of certified Attribute authority (AA_j) that is responsible of issuing the required attributes. After defining the access structure (A, ρ), the data owner encrypts the data file D_F, using the `encdata` algorithm. We note that our encryption algorithm relies on Lewko and Waters ABE scheme [11]. That is, we extend this proposal [11] to support a decentralised setting for deriving a set of private keys related to a set of attributes from each single attribute authority, while preserving users' privacy with respect to the involved attribute authorities.

The STORE procedure consists of the `encdata` algorithm, defined as follows:

- `encdata` – the data owner O first picks two random values $p,s \in \mathbb{Z}_N^2$. Then, the `encdata` algorithm computes λ_i and w_i such that $\lambda_i = \vec{A_i} \cdot \vec{v}$ where $\vec{v} = [s, v_1, \cdots, v_l] \in {\mathbb{Z}_N}^l$ is a random vector and $w_i = \vec{A_i} \cdot \vec{\tau}$ such as $\vec{\tau} = [0, \tau_1, \cdots, \tau_l] \in {\mathbb{Z}_N}^l$ is a random vector. The `encdata` algorithm outputs the ciphertext as a tuple $E_D = (C_0, C_{1,i}, C_{2,i}, C_{3,i})_{i\in[1,n]}$, where i presents a matrix row corresponding to an attribute i, defined as follows:

$$\begin{cases} C_0 = D_F \hat{e}\cdot(g,g)^s \\ C_{1,i} = \hat{e}(g,g)^{\lambda_i} \\ C_{2,i} = g^p {u_i}^p \\ C_{3,i} = g^p g^{w_i} {u_i}^p \end{cases}$$

3) Data Backup Procedure – BACKUP*:* For the data retrieval scenario, the BACKUP procedure starts with the user' authentication, with respect to the `sign` and `verif` algorithms. Once authenticated, the requesting user executes the data decryption algorithm referred to as `decdata`, to retrieve the data file D_F.

a) Anonymous User Authentication: When a user U wants to access to the encrypted data file (E_D) outsourced by the data owner O, the CSP has first to authenticate the user U, with respect to the access structure (A,ρ) associated with the encrypted data file. So that, the cloud provider sends a random value m which consists of the cloud provider identity concatenated with the current time (i.e. m is assumed to be different for each authentication session). The requesting user has then to sign the received value m with respect to the access structure (A,ρ) and sends his signature to the cloud provider. We note that if the verification fails, the user cannot access to data and the cloud provider does not send the encrypted data file. The anonymous authentication procedure consists of two algorithms, defined as follows:

- `sign` – the user first selects the sub-set of his attributes S_U that satisfies the access policy, such as: $\psi(S_U) = 1$ and signs the received value m. The user finally sends the signature Σ to the cloud provider who checks the resulting signature. Thus, the user first converts ψ to its corresponding monotone span program A which is an $n\times l$ access matrix, with respect to the row labeling function $\rho : [n] \to S_U$. In addition, he computes the vector $\vec{y}$ such as $\psi(S_U) = 1$ and $\vec{y}\cdot\vec{A} = [1,0,\cdots,0]$. In order to sign the random token m, the data owner first chooses two random values $r,t\in\mathbb{Z}_N$ and computes the signature $\Sigma = (x_{i\in[1,n]}, z_{j\in[1,\kappa]}, \sigma_1, \sigma_2, \sigma_3, \sigma_4)$ as follows:

$$\begin{cases} \sigma_1 = \prod_{i=1}^n \prod_{j=1}^\kappa [(K_{\{\rho(i),j\}} {u_i}^t)]^{y_i} h^{mt} g^{mr} h^t \\ \sigma_2 = \prod_{j=1}^\kappa M_j h^t \\ \sigma_3 = \prod_{j=1}^\kappa L_j g^t \\ \sigma_4 = \hat{e}(g, g^r h^t) \\ x_i = [\prod_{j=1}^\kappa L_j g^t]^{y_i} \\ z_j = \hat{e}(\check{pk_{AA_j}}, h^{\mathcal{H}(id)})\hat{e}(g,h)^t \end{cases}$$

for all $i\in[1,n]$ and $j\in[1,\kappa]$ Where κ is the number of involved attribute authorities.
Finally, the signature for the message m generated by the user with respect to the signing policy (A,ρ) is set as follows:

$$\Sigma = (x_{i\in[1,n]}, z_{j\in[1,\kappa]}, \sigma_1, \sigma_2, \sigma_3, \sigma_4) \tag{1}$$

- `verif` – the access policy ψ is first converted to its corresponding monotone span program A. Then, the CSP computes the vector $\vec{\beta} = [1,\beta_2,\cdots,\beta_n]$, such that $\{\beta_i\}_{i\in[2,n]}$ are randomly chosen and computes $\mu_i = \sum_{j=1}{}^l \beta_j A_{i,j}$. The cloud server accepts the signature if both Equation 2 and Equation 3 hold.
First, the CSP verifies the following equation:

$$\hat{e}(g,\sigma_1.\sigma_2) \overset{?}{=} \prod_{j=1}^{\kappa}[\hat{pk_{AA_j}}\cdot z_j\cdot\sigma_4^m]\prod_{i=1}^{n}\hat{e}(h^{\mu_i}u_{\rho(i)}, x_i) \tag{2}$$

Then, he computes the vector $\vec{y}$ such as $\vec{y}\cdot\vec{A} = [1,0,\cdots,0]$ and verifies that the following equation holds for all $i\in[1,n]$:

$$x_i \overset{?}{=} \sigma_3^{y_i} \tag{3}$$

The correctness of the signature verification algorithm is detailed in Section VI-A.

b) Data Retrieval: The data retrieval procedure consists of the `decdata` algorithm, defined as follows:

- `decdata` – for each matrix row i, the user computes:

$$\begin{aligned} ® &= \frac{C_{1,i}.\hat{e}(\prod_{j=1}^\kappa L_j\cdot K_{i,j}, C_{3,i})}{\hat{e}(\prod_{j=1}^\kappa L_j\cdot K_{i,j}, C_{2,i})} \\ &= \hat{e}(g,g)^{\lambda_i}\hat{e}(g,g)^{w_i\sum_{j=1}^\kappa t_j}\hat{e}(u_i,g)^{w_i\sum_{j=1}^\kappa t_j}\hat{e}(u_i,g)^{w_i f_{Uj}} \end{aligned}$$

Afterwards, the user chooses a set of constants $\{c_i\}_{i\in[1,n]}\in\mathbb{Z}_N$ such that $\sum_i c_i\vec{A_i} = [1,0,\cdots,0]$. Then, he performs:

$$\begin{aligned} \prod_{i=1}^n ®^{c_i} &= \prod_{i=1}^n(\hat{e}(g,g)^{\lambda_i}\hat{e}(g,g)^{w_i\sum_{j=1}^\kappa t_j}\hat{e}(u_i,g)^{w_i\sum_{j=1}^\kappa t_j}\hat{e}(u_i,g)^{w_i f_{Uj}})^{c_i} \\ &= \prod_{i=1}^n \hat{e}(g,g)^{\lambda_i c_i}\hat{e}(g,g)^{\sum_{j=1}^\kappa t_j w_i c_i}\hat{e}(u_i,g)^{\sum_{j=1}^\kappa t_j w_i c_i} \\ &\quad \hat{e}(u_i,g)^{f_{Uj} w_i c_i} \\ &= \hat{e}(g,g)^{\sum_{i=1}^n \lambda_i c_i}\hat{e}(g,g)^{\sum_{j=1}^\kappa t_j \sum_{i=1}^n w_i c_i}\hat{e}(u_i,g)^{\sum_{j=1}^\kappa t_j \sum_{i=1}^n w_i c_i} \\ &\quad \hat{e}(u_i,g)^{f_{Uj}\sum_{i=1}^n w_i c_i} \end{aligned}$$

We note that $\lambda_i = \vec{A}.\vec{v}$ and $w_i = \vec{A}.\vec{\tau}$, where $\vec{v}.[1,0,\cdots,0] = \sum_{i=1}^n \lambda_i c_i = s$ and $\vec{w}.[1,0,\cdots,0] = \sum_{i=1}^n w_i c_i = 0$.
In the sequel, the user gets the following result:

$$\prod_{i=1}^n ®^{c_i} = \hat{e}(g,g)^s \tag{4}$$

The data file D_F is then obtained such as:

$$\begin{aligned} D_F &= \frac{C_0}{\prod_{i=1}^n ®^{c_i}} \\ &= \frac{C_0}{\hat{e}(g,g)^s} \end{aligned}$$

The proof of correctness of the decryption algorithm is detailed in Section VI-A.

D. System Inspection : INSPEC

The inspection procedure INSPEC carries out two deterministic algorithms. These algorithms are denoted by `trace` and `judge` respectively and are defined as follows:

- `trace` – based on the signature Σ, the inspection authority IA uses his secret key s_{ins} to extract the identity of the related signer, based on Equation 5.

$$\left(\frac{\hat{e}(g, \sigma_2 \cdot \sigma_3^{-s_{ins}})}{\prod_{j=1}^{\kappa} \hat{pk_{AA_j}}}\right)^{\frac{1}{s_{ins}}} = \hat{e}(g, \prod_{j=1}^{\kappa} \check{pk_{AA_j}})^{\mathcal{H}(Id)} \tag{5}$$

  Then, the inspection authority uses a matching table in order to retrieve an entry equal to $\hat{e}(g, \prod_{j=1}^{\kappa} \check{pk_{AA_j}})^{\mathcal{H}(Id)}$. In the sequel, IA returns the signer identity $sid = \hat{e}(g, \prod_{j=1}^{\kappa} \check{pk_{AA_j}})^{\mathcal{H}(id)}$ and a proof π attesting to this claim. π is generated for the `judge` algorithm to check that the tracing procedure is done correctly. In other words, this algorithm double checks whether the user identity *sid* has been produced properly with respect to the received signature Σ.
- `judge` – the proof of validity of such an inspection procedure is done by proving that the decryption is correctly performed, using the knowledge of the inspection secret key s_{ins}. It outputs 1 if π is a valid proof that *sid* has produced Σ or 0 otherwise. In other words, this algorithm double checks whether the user identity *sid* has been produced properly with respect to the received signature Σ.

VI. SECURITY DISCUSSION

In this section, we prove the correctness of our Ins-PAbAC construction (c.f. Section VI-A) and we discuss the resistance of Ins-PAbAC against two adversaries, based on two realistic threat models. We prove the security of our proposed scheme with respect to the security model as defined in Section IV-C.

A. Correctness

We detail the correctness of our construction, with respect to Theorem 1.

Theorem 1. Authorized users can successfully authenticate and decrypt enciphered data files.

We recall that cloud users have to collect their certified attributes and the related secret keys from attribute authorities AAs. As such, in Ins-PAbAC, only users, having valid private keys related to their attributes, are able to access data stored in the cloud, once successfully authenticated with the cloud server. This is due to both the correctness of our decryption and signature verification algorithms, asserted by Lemma 1 and Lemma 2 respectively and the compliance of the unforgeability property of the Ins-PAbAC signature scheme, inherited from [58] and emphasized by Lemma 3.

In the following, we assume that there are κ involved attribute authorities. Each attribute authority is responsible of generate the secret key sk_{S_j} related to the set of attributes S_j, such as $S_j \subset S_U$, $\sum_{j \in [1,\kappa]} S_j = S_U$ and $\psi(S_U) = 1$.

Lemma 1. Data Decryption Correctness.

Proof. After receiving his attributes' secret key sk_{S_j} from each involved attribute authority AA_j, the authorized user first computes:

$$\begin{aligned}
\text{®} &= \frac{C_{1,i}.\hat{e}(\prod_{j=1}^{\kappa} L_j \cdot K_{i,j}, C_{3,i})}{\hat{e}(\prod_{j=1}^{\kappa} L_j \cdot K_{i,j}, C_{2,i})} \\
&= \frac{\hat{e}(g,g)^{\lambda_i}.\hat{e}(\prod_{j=1}^{\kappa} g^{t_j} g^{f_{Uj}} {u_i}^{t_j} {u_i}^{f_{Uj}}, g^p g^{w_i} {u_i}^p)}{\hat{e}(\prod_{j=1}^{\kappa} g^{t_j} g^{f_{Uj}} {u_i}^{t_j} {u_i}^{f_{Uj}}, g^p {u_i}^p)} \\
&= \hat{e}(g,g)^{\lambda_i} \hat{e}(g,g)^{w_i \sum_{j=1}^{\kappa} t_j} \hat{e}(u_i,g)^{w_i \sum_{j=1}^{\kappa} t_j} \hat{e}(u_i,g)^{w_i f_{Uj}}
\end{aligned}$$

Then, he computes the constants $c_i \in \mathbb{Z}_N$ such that $\sum_i c_i \cdot \vec{A_i} = [1, 0, \cdots, 0]$.

Note that $\lambda_i = \vec{A_i} \cdot \vec{v}$ where $\vec{v} = [s, v_2, \cdots, v_n]$ and $w_i = \vec{A_i} \cdot \vec{\tau}$ such as $\vec{\tau} = [0, \tau_2, \cdots, \tau_n]$. Hence, we deduce that $\sum_{i=1}^{n} \lambda_i c_i = s$ and $\sum_i w_i c_i = 0$. Consequently, the value $\hat{e}(g,g)^s$ is derived as follows:

$$\begin{aligned}
\prod_{i=1}^{n} \text{®}^{c_i} &= \prod_{i=1}^{n} (\hat{e}(g,g)^{\lambda_i} \hat{e}(g,g)^{w_i \sum_{j=1}^{\kappa} t_j} \hat{e}(u_i,g)^{w_i \sum_{j=1}^{\kappa} t_j} \hat{e}(u_i,g)^{w_i f_{Uj}})^{c_i} \\
&= \prod_{i=1}^{n} \hat{e}(g,g)^{\lambda_i c_i} \hat{e}(g,g)^{\sum_{j=1}^{\kappa} t_j w_i c_i} \hat{e}(u_i,g)^{\sum_{j=1}^{\kappa} t_j w_i c_i} \\
&\quad \hat{e}(u_i,g)^{f_{Uj} w_i c_i} \\
&= \hat{e}(g,g)^{\sum_{i=1}^{n} \lambda_i c_i} \hat{e}(g,g)^{\sum_{j=1}^{\kappa} t_j \sum_{i=1}^{n} w_i c_i} \hat{e}(u_i,g)^{\sum_{j=1}^{\kappa} t_j \sum_{i=1}^{n} w_i c_i} \\
&\quad \hat{e}(u_i,g)^{f_{Uj} \sum_{i=1}^{n} w_i c_i} \\
&= \hat{e}(g,g)^s \hat{e}(g,g)^0 \hat{e}(u_i,g)^0 \hat{e}(u_i,g)^0 \\
&= \hat{e}(g,g)^s
\end{aligned}$$

Therefore, the data file D_F can be obtained as follows:

$$\begin{aligned}
D_F &= \frac{C_0}{\prod_{i=1}^{n} \text{®}^{c_i}} \\
&= \frac{C_0}{\hat{e}(g,g)^s}
\end{aligned}$$

□

Lemma 2. Authorized users can successfully authenticate to the Cloud Service Provider.

Proof. When an authorized user wants to access outsourced data, he has to provide a correct signature Σ, with respect to the access policy ψ defined by the data owner, that has to be verified by the CSP in an anonymous way.

If the generated signature $\Sigma = (x_{i \in \{1,n\}}, z_{j \in \{1,\kappa\}}, \sigma_1, \sigma_2, \sigma_3, \sigma_4)$ is a valid signature of the message m for the predicate ψ, then both Equation 2 and Equation 3 hold. Subsequently, the CSP first verifies that $x_i = \sigma_3^{y_i}$ where $\vec{y} \cdot \vec{A} = [1, 0, \cdots, 0]$ and $i \in [1, n]$. Afterwards, the CSP proceeds in verifying that Equation 2 holds as follows:

$$\begin{aligned}
Ⓢ &= \hat{e}(g, \sigma_1 \cdot \sigma_2) \\
&= \hat{e}(g, \prod_{i=1}^{n}\prod_{j=1}^{\kappa}({u_i}^{t_j}{u_i}^{f_{Uj}}{u_i}^{t})^{y_i} g^{mr} h^{mt} h^t \prod_{j=1}^{\kappa} g^{\alpha_j} h^{t_j} \cdot h^{t_j \mathcal{H}(Id)} h^t h^{f_{Uj}}) \\
&= \hat{e}(g, \prod_{j=1}^{\kappa} g^{\alpha_j} \prod_{j=1}^{\kappa} h^{t_j \mathcal{H}(Id)+t} (g^r h^t)^m \prod_{j=1}^{\kappa}\prod_{i=1}^{n} {u_i}^{(t_j+t+f_{Uj})y_i} h^{t_j+t+f_{Uj}}) \\
&= \hat{e}(g, \prod_{j=1}^{\kappa} g^{\alpha_j}) \hat{e}(g, \prod_{j=1}^{\kappa} h^{t_j \mathcal{H}(Id)+t}) \hat{e}(g, (g^r h^t))^m \\
&\quad \hat{e}(g, \prod_{j=1}^{\kappa}\prod_{i=1}^{n} {u_i}^{(t_j+t+f_{Uj})y_i} h^{t_j+t+f_{Uj}}) \\
&= \prod_{j=1}^{\kappa} \hat{e}(g,g)^{\alpha_j} \hat{e}(g,h)^{t_j \mathcal{H}(Id)+t} \hat{e}(g, g^r h^t)^m \\
&\quad \prod_{j=1}^{\kappa}\prod_{i=1}^{n} \hat{e}(g, {u_i}^{(t_j+t+f_{Uj})y_i} h^{t_j+t+f_{Uj}}) \\
&= \prod_{j=1}^{\kappa} \hat{e}(g,g)^{\alpha_j} \hat{e}(g,h)^{t_j \mathcal{H}(Id)+t} \hat{e}(g, g^r h^t)^m \\
&\quad \prod_{j=1}^{\kappa}\prod_{i=1}^{n} \hat{e}(g, {u_i}^{(t_j+t+f_{Uj})y_i} h^{(t_j+t+f_{Uj})\sum_{i=1}^{n}\mu_i y_i}) \\
&= \prod_{j=1}^{\kappa} \hat{e}(g,g)^{\alpha_j} \hat{e}(g,h)^{t_j \mathcal{H}(Id)+t} \hat{e}(g, g^r h^t)^m \\
&\quad \prod_{j=1}^{\kappa}\prod_{i=1}^{n} \hat{e}(g, {u_i}^{(t_j+t+f_{Uj})y_i} h^{(t_j+t+f_{Uj})\mu_i y_i}) \\
&= \prod_{j=1}^{\kappa} \hat{e}(g,g)^{\alpha_j} \hat{e}(g^{t_j}, h^{\mathcal{H}(Id)}) \hat{e}(g,h)^t \hat{e}(g, g^r h^t)^m \\
&\quad \prod_{j=1}^{\kappa}\prod_{i=1}^{n} \hat{e}(g^{(t_j+t+f_{Uj})y_i}, u_i h^{\mu_i}) \\
&= \prod_{j=1}^{\kappa} \hat{pk}_{AA_j} z_j {\sigma_4}^m \prod_{i=1}^{n} \hat{e}(x_i, h^{\mu_i} u_i)
\end{aligned}$$

Note that $\mu_i = \sum_{j=1}^{l} \beta_j A_{i,j}$, the last equality is obtained by:

$$\begin{aligned}
\sum_{i=1}^{n} \mu_i y_i (\sum_{j=1}^{\kappa} t_j + t + f_{Uj}) &= (\sum_{j=1}^{\kappa} t_j + t + f_{Uj}) \sum_{i=1}^{n} \mu_i y_i \\
&= \sum_{j=1}^{\kappa} (t_j + t + f_{Uj}).1 + 0 \\
&= \sum_{j=1}^{\kappa} t_j + t + f_{Uj}
\end{aligned}$$

□

Lemma 3. Inspection Correctness.

Proof. The inspection authority uses his secret key s_{ins} to deduce the signer identity of a given signature Σ. First, IA uses the user's signature to compute the equation 5 as follows:

$$\begin{aligned}
Ⓡ &= \hat{e}(g, \sigma_2 \cdot (\sigma_3)^{-s_{ins}}) \\
&= \hat{e}(g, \prod_{j=1}^{\kappa} M_j h^t \cdot \prod_{j=1}^{\kappa} (L_j g^t)^{-s_{ins}}) \\
&= \hat{e}(g, \prod_{j=1}^{\kappa} g^{\alpha_j} h^{t_j} h^{t_j \mathcal{H}(Id)} h^t h^{f_{Uj}} \prod_{j=1}^{\kappa} (g^{t_j} g^t g^{f_{Uj}})^{-s_{ins}}) \\
&= \prod_{j=1}^{\kappa} \hat{e}(g,g)^{\alpha_j} \prod_{j=1}^{\kappa} \hat{e}(g, g^{t_j s_{ins}} g^{f_{Uj} s_{ins}} g^{t_j \mathcal{H}(Id) s_{ins}} g^{t s_{ins}} g^{-t_j s_{ins}} g^{-f_{Uj} s_{ins}} \\
&\quad g^{-t s_{ins}}) \\
&= \prod_{j=1}^{\kappa} \hat{e}(g,g)^{\alpha_j} \cdot \prod_{j=1}^{\kappa} \hat{e}(g, g^{t_j \mathcal{H}(Id) s_{ins}}) \\
&= \prod_{j=1}^{\kappa} \hat{pk}_{AA_j} \cdot \prod_{j=1}^{\kappa} \hat{e}(g,g)^{t_j \mathcal{H}(Id) s_{ins}} \\
&= \prod_{j=1}^{\kappa} \hat{pk}_{AA_j} \cdot \prod_{j=1}^{\kappa} \hat{e}(g, g^{t_j})^{s_{ins} \mathcal{H}(Id)}
\end{aligned}$$

Subsequently, the inspector computes:

$$\begin{aligned}
\frac{Ⓢ^{\frac{1}{s_{ins}}}}{\prod_{j=1}^{\kappa} \hat{pk}_{AA_j}} &= \hat{e}(g, \prod_{j=1}^{\kappa} g^{t_j})^{\mathcal{H}(Id)}) \\
&= \hat{e}(g, \prod_{j=1}^{\kappa} \check{pk}_{AA_j})^{\mathcal{H}(id)}
\end{aligned}$$

Finally, the inspection authority uses the matching table in order to retrieve an entry equal to $\hat{e}(g, \prod_{j=1}^{\kappa} \check{pk}_{AA_j})^{\mathcal{H}(Id)}$. □

B. Indistinguishability

In the following proof, we prove that our Ins-PAbAC scheme is CPA-Secure against static corruption of the attribute authorities with respect to Theorem 1.

Theorem 1. *If Lewko et al. decentralized CP-ABE scheme [11] is CPA-secure, then, our Ins-PAbAC scheme is selectively CPA-secure such that $Adv_{\mathcal{A}}[Exp^{conf}] \leq Adv_{\mathcal{A}}[Exp^{Lewko}]$, according to Definition 1 with respect to the hardness of the CDH assumption (Definition 6).*

Proof. We define a PPT algorithm adversary $\mathcal{A}$ running the Exp^{ind} security game defined in Section IV-C1 with an entity $\mathcal{B}$. This entity $\mathcal{B}$ is also running the Lewko et al's CPA-security game (Lewko-Game) with a challenger $\mathcal{C}$. The objective of the proof is to show that the advantage of the adversary $\mathcal{A}$ to win the Exp^{ind} game is smaller than the advantage of the entity $\mathcal{B}$ to win Lewko-Game. Hereafter $\mathcal{A}$, $\mathcal{B}$ and $\mathcal{C}$ interactions are described, with $\mathcal{A}$ running the following steps and algorithms, as specified in the Exp^{ind} game:

Initialisation – in this phase, the adversary $\mathcal{A}$ gives the algorithm $\mathcal{B}$ a challenge access structure $\Psi^* = (A^*, \rho^*)$.

Setup – $\mathcal{B}$ runs the `setup` to generate the public parameters. For instance, it sets two multiplicative groups $\mathbb{G}_1$ and $\mathbb{G}_T$ of order N, a bilinear map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_T$, g a generator of $\mathbb{G}_1$ and a collision resistant hash function $\mathcal{H} : \{0,1\}^* \to \mathbb{Z}_N$. Finally, it outputs the public parameters PP defined as $\mathtt{PP} = \{\mathbb{G}_1, \mathbb{G}_T, N, \mathcal{H}, h, \hat{e}, u_0, \cdots, u_q\}$.

In addition, $\mathcal{B}$ calls the challenger $\mathcal{C}$ to execute the $\texttt{setup}_{\texttt{auth}}$ algorithm to generate the attribute authorities public keys. Then, $\mathcal{C}$ chooses two random numbers $\alpha_j, t_j \in \mathbb{Z}_N$. Then, it generates the attribute authorities public keys $\{pk_{AA_j}\}$ defined as $pk_{AA_j} = (\hat{e}(g,g)^{\alpha_i}, g^{t_i})$. Finally, $\mathcal{C}$ sends to $\mathcal{B}$ the attribute authorities public keys $\{pk_{AA_j}\}$ which are sent next by $\mathcal{B}$ to $\mathcal{A}$.

Queries phase 1 – for each session k, the adversary issues a key query by submitting a set of attributes S_j and his identity Id. Then, the algorithm $\mathcal{B}$ uses the challenger $\mathcal{C}$ to generate and return the corresponding secret keys to the adversary $\{sk_{S_j}\}_k = \{(M_j, L_j, K_{i,j})\}_k$. Afterwards, the challenger $\mathcal{C}$ chooses $f_{Uj} \in \mathbb{Z}_N$ and sets $\{sk_{S_j}\}_k = \{(M_j, L_j, K_{i,j})\}_k = \{(g^{\alpha_j} h^{t_j} h^{f_{U_j}} h^{t_j \mathcal{H}(Id)}, g^{t_j} g^{f_{U_j}}, {u_i}^{t_j} {u_i}^{f_{U_j}})\}_k$. The secret keys sk_{S_j} are returned to $\mathcal{B}$ who forwards them to the adversary $\mathcal{A}$.

Challenge – the adversary $\mathcal{A}$ sends two different equal length messages $\{M_0, M_1\} \in \mathbb{G}_T$ to the algorithm $\mathcal{B}$. This latter chooses a bit $b \in \{0,1\}$ and sends M_b to the challenger $\mathcal{C}$. Afterwards, $\mathcal{C}$ chooses a bit $b \in \{0,1\}$ and encrypts M_b under the challenge access structure $\Psi^* = (A^*, \rho^*)$. The challenger, in his turn, selects a random bit $b \in [0,1]$ and encrypts a message $M_b \in \{M_0, M_1\}$ as the challenge ciphertext E_{Db}. To do so, it selects the random values $p, s, v_{i \in [1,l]}, w_{i \in [1,l]} \in Z_N$. Then, it composes the vectors $\vec{b} = [s, v_2, \cdots, v_l]$ and $\vec{\tau} = [0, w_2, \cdots, w_l]$ and computes the shares $\lambda_i = \vec{A_i} \cdot \vec{v}$ and $w_i = \vec{A_i} \cdot \vec{w}$. The challenger computes the challenge ciphertext as follows:

$$\begin{cases} C_{0b} = M_b \hat{e}(g,g)^s \\ C_{1,i_b} = \hat{e}(g,g)^{\lambda_i} \\ C_{2,i_b} = g^p {u_i}^p \\ C_{3,i_b} = g^p g^{w_i} {u_i}^p \end{cases}$$

Then, the challenger forwards the generated ciphertext E_{Db} to the adversary $\mathcal{A}$.

Queries phase 2 – $\mathcal{A}$ continues to query a polynomially bounded number of queries and $\mathcal{B}$ answers as in **Queries Phase 1**, except that the adversary $\mathcal{A}$ can not query secret keys related to a set of attributes which satisfy the access policy $\Psi^* = (A^*, \rho^*)$.

Guess – the adversary $\mathcal{A}$ outputs a bit b'. Then, $\mathcal{B}$ sends b' to $\mathcal{C}$ as its guess about b. If $b' = b$, $\mathcal{C}$ answers 1 as the solution to the given instance of the DBDH problem with respect to Definition 6 as introduced in Lewko et al.'s security analysis [11].

The adversary tries to guess the challenge ciphertext E_{Db} was generated using which message between the two chosen plaintext messages M_0 and M_1. Hence, the adversary tries to distinguish between $C_{01} = M_1 \hat{e}(g,g)^s$ and $C_{02} = M_2 \hat{e}(g,g)^s$. Concretely, considering the modified game detailed above, the adversary must distinguish between $C_0 = \hat{e}(g,g)^s$ and $C_0 = \hat{e}(g,g)^T$ for T uniformly chosen random from $\mathbb{Z}_N$.

Likely to Lewko et al. scheme [11], we assume that the adversary's $\mathcal{A}$ view in the simulation (i.e. considering that $C_0 = \hat{e}(g,g)^s$) is identically distributed to his view if $C_0 = \hat{e}(g,g)^T$.

The adversary $\mathcal{A}$ outputs a bit b'. The probability to break the instance of Exp^{ind} game is smaller than the Lewko-Game, as it is necessary for $\mathcal{B}$ to win the game for $\mathcal{A}$ to be able to get the right E_{Db} values, and try to guess the value of b. As such $Pr[Exp_{\mathcal{A}}{}^{Lewko}(1^{\xi})] \geq Pr[Exp_{\mathcal{A}}{}^{conf-real}(1^{\xi})]$, and the advantage of adversary A is negligible. Thus, our Ins-PAbAC scheme satisfies the confidentiality property against the static corruption of the attribute authorities. □

C. Unforgeability

Our accountable Ins-PAbAC framework is considered to be unforgeable if the signature scheme is unforgeable. For instance, the user U is led to authenticate with the cloud in order to get access to the encrypted files. As a consequence, the unforgeability property of the Ins-PAbAC scheme is tightly related to the unforgeability property of the used attribute based signature.

In the following proof, we prove that our Ins-PAbAC scheme is secure against chosen-message attack (UF-CMA) with respect to Theorem 2.

Theorem 2. For any adversary $\mathcal{A}$, against chosen-message attack (UF-CMA), our Ins-PAbAC scheme is unforgeable according to Definition 2 with respect to the hardness of the CDH assumption (Definition 5).

Proof. We consider an algorithm $\mathcal{B}$ that uses the adversary $\mathcal{A}$ as a black-box and that solves the CDH problem. The adversary $\mathcal{A}$ proceeds as follows:

Initialisation – in this phase, the adversary $\mathcal{A}$ gives the algorithm $\mathcal{B}$ a challenge access structure $\Psi^* = (A^*, \rho^*)$.

Setup – $\mathcal{B}$ runs the $\texttt{setup}$ to generate the public parameters. For instance, it sets two multiplicative groups $\mathbb{G}_1$ and $\mathbb{G}_T$ of order N, a bilinear map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_T$, g a generator of $\mathbb{G}_1$ and a collision resistant hash function $\mathcal{H} : \{0,1\}^* \to \mathbb{Z}_N$. Finally, it outputs the public parameters $\texttt{PP}$ defined as $\texttt{PP} = \{\mathbb{G}_1, \mathbb{G}_T, N, \mathcal{H}, h, \hat{e}, u_0, \cdots, u_q\}$.

In addition, $\mathcal{B}$ calls the challenger $\mathcal{C}$ to execute the $\texttt{setup}_{\texttt{auth}}$ algorithm to generate the attribute authorities public keys. Then, $\mathcal{C}$ chooses two random numbers $\alpha_j, t_j \in \mathbb{Z}_N$. Then, it generates the attribute authorities public keys $\{pk_{AA_j}\}$ defined as $pk_{AA_j} = (\hat{e}(g,g)^{\alpha_i}, g^{t_i})$. Finally, $\mathcal{C}$ sends to $\mathcal{B}$ the attribute authorities public keys $\{pk_{AA_j}\}$ which are sent next by $\mathcal{B}$ to $\mathcal{A}$.

Queries phase – when the adversary requests $\mathcal{H}(Id)$ for an identity Id for the first time, the challenger chooses a random value $r_{Id} \in \mathbb{Z}_N$ and gives it to the adversary as $\mathcal{H}(Id)$. It stores the value $g^{r_{Id}}$ to the adversary. Thus, this latter stores $g^{r_{Id}}$ so that it can reply consistently to any subsequent requests for $\mathcal{H}(Id)$. For each session k, the adversary issues a key query by submitting a set of attributes S_j and his identity Id. Then, the algorithm $\mathcal{B}$ uses the challenger $\mathcal{C}$ to generate and return the corresponding secret keys to the adversary $\{sk_{S_j}\}_k = \{(M_j, L_j, K_{i,j})\}_k$. Afterwards, the challenger $\mathcal{C}$ chooses $f_{U_j} \in \mathbb{Z}_N$ and sets $\{sk_{S_j}\}_k = \{(M_j, L_j, K_{i,j})\}_k = \{(g^{\alpha_j} h^{t_j} h^{f_{U_j}} h^{t_j \mathcal{H}(Id)}, g^{t_j} g^{f_{U_j}}, {u_i}^{t_j} {u_i}^{f_{U_j}})\}_k$. The secret keys sk_{S_j} are returned to $\mathcal{B}$ who forwards them to the adversary $\mathcal{A}$.

In addition, the adversary is able to request the signature of any message m under any access structure (A,ρ). Hence, the challenger $\mathcal{C}$ computes the vector $\vec{y}$ such as $\psi(S_U)=1$ and $\vec{y}\cdot\vec{A}=[1,0,\cdots,0]$. Then, $\mathcal{C}$ chooses two random values $r,t\in\mathbb{Z}_N$ and executes the $\texttt{sign}(\texttt{PP},m,(A,\rho),\{sk_{S_j}\}_k)$ algorithm and computes the signature Σ as follows:

$$\begin{cases} \sigma_{1k}=\prod_{i=1}^{n}\prod_{j=1}^{\kappa}[(K_{\{\rho(i),j\}_k}u_i{}^t)]^{y_i}h^{mt}g^{mr}h^t \\ \sigma_{2k}=\prod_{j=1}^{\kappa}M_{j_k}h^t \\ \sigma_{3k}=\prod_{j=1}^{\kappa}L_{j_k}g^t \\ \sigma_{4k}=\hat{e}(g,g^rh^t) \\ x_{ik}=[\prod_{j=1}^{\kappa}L_{j_k}g^t]^{y_i} \\ z_{j_k}=\hat{e}(\check{pk}_{AA_j},h^{\mathcal{H}(id)})\hat{e}(g,h)^t \end{cases}$$

Forgery – the adversary $\mathcal{A}$ tries to forge a signature Σ^* of a message m^* while relying on several sessions. Obviously, $\mathcal{A}$ tries a forgery attack against the *CDH* assumption (Definition 5) considering that σ_1 is a product, over the set of user's attributes and involved attributes authorities, of a randomization of the user's secret keys. Knowing that this randomization is required for deriving the remaining signature elements, $\mathcal{A}$ must break the *CDH* assumption. Similarly, since σ_2 and σ_3 are a randomization of the user's secret keys, over the set of involved attributes authorities, $\mathcal{A}$ is led to break the *CDH* assumption. Moreover, $\mathcal{A}$ tries a forgery attack against the *CDH* assumption to produce x_i which is a randomized product of the randomized user's secret key, over the set of involved attributes authorities. We state that the complexity of the *CDH* assumption has been studied in [59] and it is proved to be hard to solve (i.e. a $(t,\varepsilon)CDH$ group is a group for which the $Exp(\mathcal{A},t)\le\varepsilon$ for every PPT adversary running in a time t).

In the sequel, with respect to the hardness of the *CDH* assumption, the adversary $\mathcal{A}$ cannot win the Exp^{unf} security game with non-negligible advantage.

Thus, our Ins-PAbAC construction is unforgeable against the chosen message attack. □

D. Privacy

Based on an attribute based signature scheme, Ins-PAbAC ensures users' privacy against curious cloud service providers. In our proposed scheme, the requesting data user has to authenticate with the cloud provider. As such, U has to sign a message received from the cloud service provider with respect to the access structure defined by the data owner. The CSP is responsible for verifying the user's access rights without knowing neither his identity nor the attributes used to sign the message.

Ins-PAbAC inherits the privacy preserving property from the Waters signature scheme [58].

Theorem 3. *For any adversary $\mathcal{A}$, against the privacy property, our Ins-PAbAC scheme is computationally private according to the Definition 3 with respect to the hardness of the CDH assumption (Definition 5).*

Proof. We consider an algorithm $\mathcal{B}$ that uses the adversary $\mathcal{A}$ which tries to distinguish between two honestly derived signatures related to two different sets of attributes.

The privacy game begins when the challenger $\mathcal{C}$ executes the $\texttt{Setup}$ algorithm to generate the public parameters.

Setup – $\mathcal{B}$ runs the $\texttt{setup}$ to generate the public parameters. For instance, it sets two multiplicative groups $\mathbb{G}_1$ and $\mathbb{G}_T$ of order N, a bilinear map $\hat{e}:\mathbb{G}_1\times\mathbb{G}_1\to\mathbb{G}_T$, g a generator of $\mathbb{G}_1$ and a collision resistant hash function $\mathcal{H}:\{0,1\}^*\to\mathbb{Z}_N$. Finally, it outputs the public parameters PP defined as $\texttt{PP}=\{\mathbb{G}_1,\mathbb{G}_T,N,\mathcal{H},h,\hat{e},u_0,\cdots,u_q\}$.

In addition, $\mathcal{B}$ calls the challenger $\mathcal{C}$ to execute the $\texttt{setup}_{\texttt{auth}}$ algorithm to generate the attribute authorities public keys. Then, $\mathcal{C}$ chooses two random numbers $\alpha_j,t_j\in\mathbb{Z}_N$. Then, it generates the attribute authorities public keys $\{pk_{AA_j}\}$ defined as $pk_{AA_j}=(\hat{e}(g,g)^{\alpha_i},g^{t_i})$. Finally, $\mathcal{C}$ sends to $\mathcal{B}$ the attribute authorities public keys $\{pk_{AA_j}\}$ which are next sent by $\mathcal{B}$ to $\mathcal{A}$.

Challenge phase – the adversary $\mathcal{A}$ chooses an access structure $\psi=(A,\rho)$, two sets of attributes S_{j_1} and S_{j_2} satisfying the monotone access structure $\psi=(A,\rho)$ such that $\psi(S_{j_1})=\psi(S_{j_2})=1$ and a message m and sends them to the challenger $\mathcal{C}$.

Afterwards, the challenger $\mathcal{C}$ picks a random bit $b\in\{1,2\}$ and executes the KeyGen algorithm such that $sk_{S_{j_b}}=\texttt{Keygen}(\texttt{PP},\{sk_{AA_j}\},Id_C,S_{j_b})=\{(M_j,L_j,K_{i,j})\}_b=(g^{\alpha_{j_b}}h^{t_{j_b}}h^{f_{U_{j_b}}}h^{t_{j_b}\mathcal{H}(Id)},g^{t_{j_b}}g^{f_{U_{j_b}}},u_i{}^{t_{j_b}}u_i{}^{f_{U_{j_b}}})$ and outputs the signature $\Sigma_b=\texttt{sign}(\texttt{PP},m,(A,\rho),sk_{S_{j_b}})$ as follows:

$$\begin{cases} \sigma_{1b}=\prod_{i=1}^{n}\prod_{j=1}^{\kappa}[(K_{\{\rho(i),j\}_k}u_i{}^t)]^{y_i}h^{m_bt}g^{m_br}h^t \\ \sigma_{2b}=\prod_{j=1}^{\kappa}M_{j_k}h^t \\ \sigma_{3b}=\prod_{j=1}^{\kappa}L_{j_k}g^t \\ \sigma_{4b}=\hat{e}(g,g^rh^t) \\ x_{ib}=[\prod_{j=1}^{\kappa}L_{j_k}g^t]^{y_i} \\ z_{j_b}=\hat{e}(\check{pk}_{AA_j},h^{\mathcal{H}(id)})\hat{e}(g,h)^t \end{cases}$$

Then, the signature Σ_b is sent to the adversary $\mathcal{A}$ as a challenge signature.

Guess – the adversary $\mathcal{A}$ tries to guess which set of attributes S_{j_1} or S_{j_2} was used to generate the signature Σ_b. $\mathcal{A}$ outputs a bit b' and wins the game if $b'=b$.

As the two signatures are derived by the same user having Id as identity for the same message m and with respect to the same access structure Ψ, they are identically distributed using the two sets of attributes.

For the attribute set S_{j_1}, the user's secret key is $sk_{S_{j_1}}=\{(M_j,L_j,K_{i,j})\}_1=(g^{\alpha_{j_1}}h^{t_{j_1}}h^{f_{U_{j_1}}}h^{t_{j_1}\mathcal{H}(Id)},g^{t_{j_1}}g^{f_{U_{j_1}}},$

${u_i}^{t_{j1}} {u_i}^{f_{U j_1}})$. And, the user's secret key generated based on the attribute set S_{j_1}, is $sk_{S_{j_2}} = \{(M_j, L_j, K_{i,j})\}_1 = (g^{\alpha_{j2}} h^{t_{j2}} h^{f_{U j_2}} h^{t_{j2}{}^{\mathcal{H}(Id)}}, g^{t_{j2}} g^{f_{U j_2}}, {u_i}^{t_{j2}} {u_i}^{f_{U j_2}})$.

As $\Psi(S_{j_1}) = \Psi(S_{j_2}) = 1$, obviously, we can prove that the signature generated using the set of attributes S_{j_1} (in other words, using the secret key $sk_{S_{j_1}}$) is similar to the signature generated using the set of attributes S_{j_2} (using the secret key $sk_{S_j 2}$).

The adversary cannot distinguish the output of oracles better than a flipping coin. As such, the probability of predicting b is equal to $\frac{1}{2}$. Hence, the adversary $\mathcal{A}$ cannot deduce the set of attributes used to generate the signature with respect to the hardness of the *CDH* problem (Definition 5).

Intuitively, we can prove that an ABS signature created using S_{j_1} can also be generated using S_{j_2}. Thus, our proposed Ins-PAbAC scheme is computationally private. □

E. Anonymity Removal

As detailed in Section IV-C4, the anonymity removal property aims at proving that there is no adversary that can (i) produce a signature while the inspection authority traces another signing user; and (ii) generate a non-traceable signature which has been successfully verified by the service provider. Recall that the second requirement covers the resistance against forgery attacks, discussed in subsection VI-C.

Theorem 4. *Our Ins-PAbAC scheme satisfies the inspection property under an anonymity removal attack with respect to the Definition 4.*

Proof. The proof of Theorem 4 consists on proving that an adversary cannot generate a signature, that cannot be traced, as detailed in the Exp^{AnR} game. Indeed, an adversary $\mathcal{A}$ has access to both secret keys and associated attributes of any user (i.e., corrupted tracing authority). However, unlike the unforgeability Exp^{Unf} game, we assume that all attributes' authorities are considered as honest.
Note that this assumption is required. In fact, if an attacker is able to hold the private key of any attribute authority, he would easily grant attributes/respective private key to dummy users that results in an untraceable signature.

For this proof, we consider an algorithm $\mathcal{B}$ that uses the adversary $\mathcal{A}$ as a black-box. The adversary $\mathcal{A}$ proceeds as follows:

Initialisation – in this phase, the adversary $\mathcal{A}$ gives the algorithm $\mathcal{B}$ a challenge access structure $\Psi^* = (A^*, \rho^*)$.

Setup – $\mathcal{B}$ runs the `setup` to generate the public parameters. For instance, it sets two multiplicative groups $\mathbb{G}_1$ and $\mathbb{G}_T$ of order N, a bilinear map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_T$, g a generator of $\mathbb{G}_1$ and a collision resistant hash function $\mathcal{H} : \{0,1\}^* \to \mathbb{Z}_N$. Finally, it outputs the public parameters PP defined as PP $= \{\mathbb{G}_1, \mathbb{G}_T, N, \mathcal{H}, h, \hat{e}, u_0, \cdots, u_q\}$.

In addition, $\mathcal{B}$ calls the challenger $\mathcal{C}$ to execute the $\texttt{setup}_{\texttt{auth}}$ algorithm to generate the attribute authorities public keys. Then, $\mathcal{C}$ chooses two random numbers $\alpha_j, t_j \in \mathbb{Z}_N$. Then, it generates the attribute authorities public keys $\{pk_{AA_j}\}$ defined as $pk_{AA_j} = (\hat{e}(g,g)^{\alpha_i}, g^{t_i})$. Finally, $\mathcal{C}$ sends to $\mathcal{B}$ the attribute authorities public keys $\{pk_{AA_j}\}$ which are next sent by $\mathcal{B}$ to $\mathcal{A}$.

Queries phase – when the adversary requests $\mathcal{H}(Id)$ for an identity Id for the first time, the challenger chooses a random value $r_{Id} \in \mathbb{Z}_N$ and gives it to the adversary as $\mathcal{H}(Id)$. It stores the value $g^{r_{Id}}$ to the adversary. Thus, this latter stores $g^{r_{Id}}$ so that it can reply consistently to any subsequent requests for $\mathcal{H}(Id)$. For each session k, the adversary issues a key query by submitting a set of attributes S_j and his identity Id. Then, the algorithm $\mathcal{B}$ uses the challenger $\mathcal{C}$ to generate and return the corresponding secret keys to the adversary $\{sk_{S_j}\}_k = \{(M_j, L_j, K_{i,j})\}_k$. Afterwards, the challenger $\mathcal{C}$ chooses $f_{U_j} \in \mathbb{Z}_N$ and sets $\{sk_{S_j}\}_k = \{(M_j, L_j, K_{i,j})\}_k = \{(g^{\alpha_j} h^{t_j} h^{f_{U_j}} h^{t_j \mathcal{H}(Id)}, g^{t_j} g^{f_{U_j}}, {u_i}^{t_j} {u_i}^{f_{U_j}})\}_k$. The secret keys sk_{S_j} are returned to $\mathcal{B}$ who forwards them to the adversary $\mathcal{A}$.

In addition, the adversary is able to request the signature of any message m under any access structure (A, ρ).

Output – the adversary generates a signature Σ^* on a message m^* selected by the challenger, with respect to the challenge access structure (A^*, ρ^*). The adversary wins the Exp^{AnR} game if Σ^* is untraceable by the challenger $\mathcal{C}$, relying on the `trace` algorithm.

By absurd-um, let us suppose that an adversary $\mathcal{A}$ can win the Exp^{AnR} anonymity removal game. Indeed, two main key-points are deduced. First, the inspection authority traces a signature, produced by the adversary $\mathcal{A}$ during the anonymity removal attack, on a new user U'. To do so, $\mathcal{A}$ is led to know the identity of the new user U', extract his private key and generate a valid signature Σ related to U'. In other words, the adversary must win the unforgeability game which contradicts the unforgeability property proved above in Section VI-C. Second, the adversary can not compute a valid signature Σ where the `trace` algorithm outputs an error message thanks to Equation 3 of the `verif` algorithm.

As a consequence, the probability of winning the Exp^{AnR} game is negligible. Hence, our Ins-PAbAC scheme achieves the inspection feature. □

VII. Theoretical Performance Analysis

In this section, we present the computation and storage complexities of our accountable Ins-PAbAC protocol. To this end, we are interested in the computations performed at the data owner side in order to execute the STORE procedure. In addition, we will consider the computation cost related to the execution of the BACKUP procedure by both the user (U) and the cloud service provider (CSP).
In the following, we denote by:

- E_1 : exponentiation in $\mathbb{G}_1$
- E : exponentiation in $\mathbb{G}_T$
- τ_P : computation of a pairing function $\hat{e}$

Table III details the performance comparison with most closely related data sharing schemes in cloud environments. In addition, Table IV details the communication costs of Ins-PAbAC.

TABLE III: Access Control Mechanisms in Cloud Data Storage Environments: Computation Cost Comparison

Scheme	*Data Owner Comp.*	*CSP Comp.*	*User comp.*	*IA comp.*	*Edge Server comp.*
[28]	$E+(2n+1)E_1+\tau_P$	$nE_1+2(n+1)\tau_P$	$2E_1(n+1)+nE+(3+n)\tau_P$	–	–
[60]	$E+(2n+1)E_1+\tau_P$	–	$(2+n)\tau_P+nE$	–	–
[48]	$(2n+1)E+(3n+3r)E_1+\tau_P$	–	$(n+r)E+2(n+r)\tau_P$	–	–
[47]	$(2n+1)E+(5n+2+2nl)E_1+\tau_P$	$\tau_P(3+n+l)+(2nl+1)E_1$	$2n\tau_P+nE$	–	–
[37]	$E+(5n+3+2nl)E_1+\tau_P$	$\tau_P(3+n+l)+(2nl+1)E_1$	$(2n+1)\tau_P+(n+1)E$	–	–
[43]	$(2n+1)E+(5n+2+2nl)E_1+\tau_P$	$\tau_P(3+n+l)+(2nl+1)E_1$	$2n\tau_P+nE$	–	–
[42]	$3nE_1+(2n+1)E+\tau_P$	$--$	$2n\tau_P+nE$	–	–
[61] (Delegated decryption)	$6E_1+2E_T+O(\mathcal{H})$	–	$4\tau_p$	–	$E_T+2O(\mathcal{H})$
[51] (Delegated decryption)	$5E_1+E_T+3O(\mathcal{H})$	–	$E_T+3O(\mathcal{H})$	–	$3n\tau_p+E_T$
[62] (Delegated decryption)	$(k-m+2+r)E_1+E+1$	$tE_1/(2t+2)E_1+\tau_p$	$2\tau_p+E_1+nE$	–	$E+2O(\mathcal{H})$
Ins-PAbAC	$(n+1)E+5nE_1+\tau_P$	$(2+n)\tau_P+(n+1)E_1$	$(4n+6)E_1+(2n+\kappa+2)\tau_P+(n+2)E$	$E_1+2E+\tau_p$	–

The STORE procedure consists of performing the encryption algorithm `encdata`. During this procedure, the data owner has to encrypt the data file. As such, he calculates one pairing function $\hat{e}(g,g)$ and nE exponentiations in $\mathbb{G}$ to compute each of $C_{1,i}$ where n is the number of attributes. In addition, the data owner executes $5n$ exponentiations in $\mathbb{G}_1$ to calculate $C_{2,i}$ and $C_{3,i}$. The BACKUP procedure is made up of three algorithms, such that `verif` is executed by the cloud server while `sign` and `decdata` are run by the data user (U). The user first signs a random message in order to authenticate with the cloud server. To sign the message, the user performs $(4n+6)$ exponentiations in $\mathbb{G}_1$, 2 exponentiations in $\mathbb{G}$ and $(\kappa+2)\tau_P$ pairing functions. Then, this latter executes $2n$ pairing to calculate $\hat{e}(K_i \cdot L, C_{3,i})$ and $\hat{e}(K_i \cdot L, C_{2,i})$ and n exponentiations in $\mathbb{G}$ to decrypt the data file. In the verification phase, the CSP executes the `verif` algorithm. As such, the cloud provider performs $(n+2)$ pairing functions' computations and $n+1$ exponentiations in $\mathbb{G}_1$.

The access control schemes [42], [43], [47] are based on the Lewko's decentralized attribute based encryption scheme [11]. During the encryption phase, the data owner has to perform one pairing function $\hat{e}(g,g)$ and $2n$ exponentiations in $\mathbb{G}_T$ to calculate each of $C_{1,i}$. In addition, to calculate $C_{2,i}$ and $C_{3,i}$, the data owner performs $3n$ exponentiations in $\mathbb{G}_1$. In the data decryption phase, the data user performs n exponentiations in $\mathbb{G}_T$ and $2n$ pairing functions. Huang et al. [28] and Li et al. [60] proposals consist of performing one exponentiation in $\mathbb{G}_T$, one pairing function and $2n+1$ exponentiations in $\mathbb{G}_1$ in order to generate the ciphertext. In the decryption process, $2+n$ pairing functions and n exponentiations in $\mathbb{G}_T$ are performed. The Zhao et al.'s proposal [37] is based on the use of the CP-ABE scheme proposed by Bethencourt et al. [8]. To encrypt the data file, the data owner performs $(2n+1)$ exponentiations in $\mathbb{G}_1$ and one exponentiation in $\mathbb{G}_T$. While decrypting data, the user performs $n+1$ exponentiations in $\mathbb{G}_T$ and $2n+1$ pairing functions.

The proposals [43],[37], and [47] are based on the use of the attribute based signature scheme proposed by [38]. In order to sign the message, the user performs $2+3n+2nl$ exponentiations in $\mathbb{G}_1$, where n is the number of rows of the access matrix A and l presents the number of columns of A. In the verification phase, the CSP has to perform $3+n+l$ pairing functions and $2nl+1$ exponentiations in $\mathbb{G}_1$. In Huang's et al. [28] proposal, the message's signature requires the computation of $2(n+1)$ pairing functions. Besides, the CSP has to compute $2(n+1)$ pairing functions and n exponentiations in $\mathbb{G}_1$ to perform the signature's verification.

Unlike most of the mentioned solutions above, our accountable Ins-PAbAC framework introduces the accountability feature. Hence, to reveal a user identity, the inspection authority IA performs one exponentiation in $\mathbb{G}_1$, one pairing function and 2 exponentiations in $\mathbb{G}_T$.

As depicted by Table IV, the communications overhead of Ins-PAbAC is acceptable in a cloud environment. Compared to other state of the art schemes, Ins-PAbAC involves the inspection procedure which requires forwarding the access policy and the signature to the inspection authority. Besides, during the user's authentication, the cloud server forwards a random message which may be a short-sized message to be signed and returned by the user. In addition, state of the art schemes use two different access policies for signing ($\Psi_{encrypt}$) and encrypting (Ψ_{sign}). Therefore, the communication overhead between users and attributes authorities grows because they receive two different sets of secret keys for signing and decrypting $\{sk_{S_j}\}_{sign}$ and $\{sk_{S_j}\}_{encrypt}$, respectively. However, Ins-PAbAC requires only one access policy for both procedures, therefore users receive only one set of secret keys to sign and decrypt data.

In Table III, we reviewed some ABE schemes [51], [61], [62] that support decryption delegation. In these schemes, the data user delegates the decryption algorithm execution to an edge server. This latter partially decrypts the ciphertext then returns the result to the user who only executes few mathematical operations to retrieve the plaintext. This technique can be applied to Ins-PAbAC as well in order to achieve less computation overheads on the data user side.

Above all, the computation overheads of the accountable Ins-PAbAC remain competitive while providing additional features such as the accountability feature.

VIII. IMPLEMENTATION RESULTS

In this section, we present the implementation results related to our accountable Ins-PAbAC scheme. We conduct a number of experiments to evaluate the overhead of the cryptographic algorithms of the Ins-PAbAC scheme at the client side as well as at the CSP side, namely: `setup`, `setup`$_{\texttt{auth}}$, `keygen`, `encdata`, `sign`, `verif` and `decdata`. Hence, we first present the implementation context. Then, we analyse the Ins-PAbAC performances.

TABLE IV: Access Control Mechanisms in Cloud Data Storage Environments: Communication Cost Comparison

Operation	Ins-PAbAC Scheme Communication Cost	State of the Art Schemes Communication Cost [37], [28], [43]	Ins-PAbAC's Conference Version [10]
System Initialisation	$(\|PP\| + \Omega \times (\|sk_{AA}\| + \|pk_{AA}\|)$ bits	$(\|PP\| + \Omega \times (\|sk_{AA}\| + \|pk_{AA}\|)$ bits	$(\|PP\| + \Omega \times (\|sk_{AA}\| + \|pk_{AA}\|)$ bits
User Key Issuing	$(\|\Omega\| \times \|n_j\| \times \|sk_{S_j}\|)$	$(\|\Omega\| \times \|n_j\| \times (\{\|sk_{S_j}\}_{sign}\| + \{\|sk_{S_j}\}_{encrypt}\|))$ bits	$(\|\Omega\| \times \|n_j\| \times \|sk_{S_j}\|)$
Data Storage	$(\|CT\| + \|\Psi\|)$ bits	$(\|CT\| + \|\Psi_{sign}\| + \|\Psi_{encrypt}\| + \|\Sigma\| + \|m\|)$ bits	$(\|CT\| + \|\Psi\|)$
User Authentication	$(\|\Sigma\| + \|m\| + \|\Psi\|)$ bits	–	$(\|\Sigma\| + \|m\| + \|\Psi\|)$ bits
System Inspection	$(\|\Sigma\| + \|\Psi\|)$ bits	–	–

TABLE V: Performances of the Used Machines

Machine	OS	Processor	Memory	CPU
ASUS	Ubuntu 12.04	Intel Core i7-5500U	8 GB	3 GHZ
DELL	Ubuntu 12.04	Intel Core i5-3337U	4 GB	1.8 GHZ

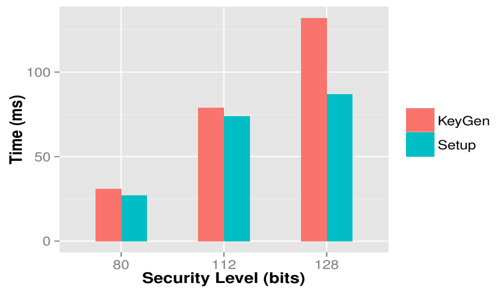

Fig. 2: System Initialisation Procedure Computation Cost Considering Different Security Levels

A. Context

The performances of our proposal are evaluated on a simulated Ins-PAbAC framework, based on the OpenStack Storage system (Swift) [12]. Our cryptographic algorithms are implemented using the following cryptographic libraries: Open-SSL [63], GMP [64] and Pairing Based Cryptography (PBC) [65]. The measurements were conducted at the user and cloud sides on two machines which properties are presented in Table V. For our tests, we used 1000 samples for getting the average durations.

B. Computation Cost Evaluation

In this section, we discuss the processing time of the different algorithms depending on various parameters. In order to evaluate the processing time of our proposed framework Ins-PAbAC, we have implemented its different procedures such as the system initialisation procedure, the data storage procedure and the data backup procedure.

As the system initialisation procedure consists of setup, $\mathtt{setup_{auth}}$ and keygen algorithms, we have evaluated their processing cost while considering different security levels (i.e. $80, 112, 128$). In cryptography, the brute-force attack consists in checking all possible keys until the correct one is found (i.e; with a key of length S bits, there are 2^S possible keys). S is defined as the security level in symmetric cryptography. In asymmetric cryptography, the security level of an algorithm is defined with respect to the hardness of solving a mathematical problem such as the Discrete Logarithm Problem (DLP). The time required to resolve the DLP problem is much less important than trying the 2^S keys by a brute-force attack. That is why, public key cryptosystems must

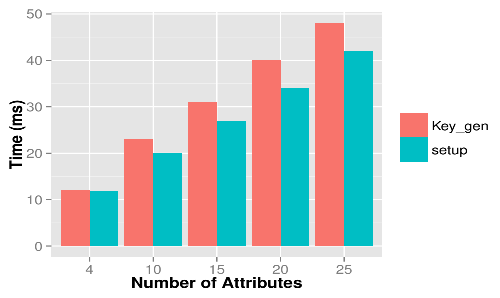

Fig. 3: System Initialisation Procedure Computation Cost Considering Different Numbers of Attributes

be longer than symmetric algorithm keys. For example, a 1024 RSA key-length bits provides a 80 key-length equivalent key of a symmetric algorithm. As shown in Figure 2, the computation cost of the algorithms increases while increasing the security level. Hence, under a security level equal to 128, the processing cost of the setup algorithm is equal to 87 ms. Moreover, the processing time of the combined $\mathtt{setup_{auth}}$ and keygen algorithms reaches 132 ms. Since the attribute based encryption and signature schemes depend on the use of a set of attributes, we focus on the time performance of the system initialisation procedure while considering different number of attributes. As depicted by Figure 3, the computation costs of the setup, the $\mathtt{setup_{auth}}$ and the keygen algorithms raise linearly with the number of attributes used. For instance, we use different numbers of attributes, 4, 10, 15, 20 and 25 to evaluate the different algorithms. The computation costs of the setup algorithm and the combined $\mathtt{setup_{auth}}$ and keygen algorithms begin by 11 ms and 12 ms, respectively, using 4 attributes, then they go up to 42 ms and 48 ms using 25 attributes. The data storage procedure is based on the execution of the data encryption algorithm encdata. In this evaluation, we studied the performance of the encdata algorithm while varying the number of attributes. Hence, Figure 4 shows that the encryption algorithm takes 20.87 ms using 4 attributes. The computation overhead reaches 68.45 ms using 14 attributes. Hence, the computation cost of the encdata algorithm increases with the augmentation of the number of attributes. As such, Ins-PAbAC presents an interesting processing cost for resource constrained devices while providing fine grained flexible access control.

The data backup procedure consists of the two different layers mainly the anonymous user authentication and the data retrieval procedures. We first perform the decryption algorithm decdata while changing the number of involved attributes. As depicted by Figure 4, the decdata algorithm requires less than 10 ms using 4 attributes. However, its computa-

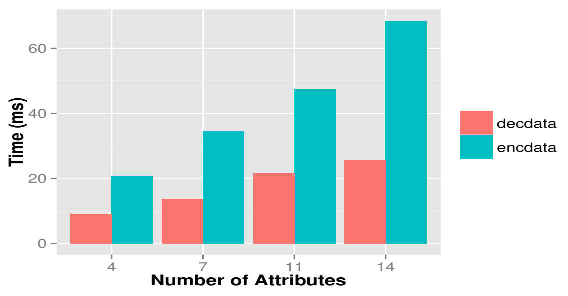

Fig. 4: Computation Overhead of the Encryption and Decryption Algorithms

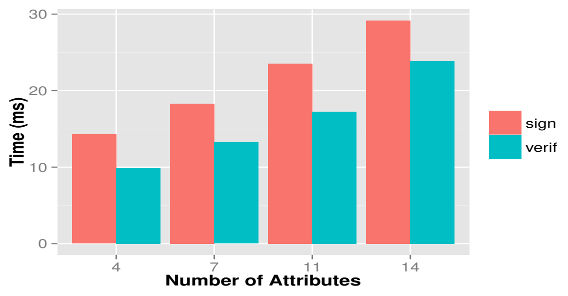

Fig. 6: Computation Overhead of the Signature and Verification Algorithms

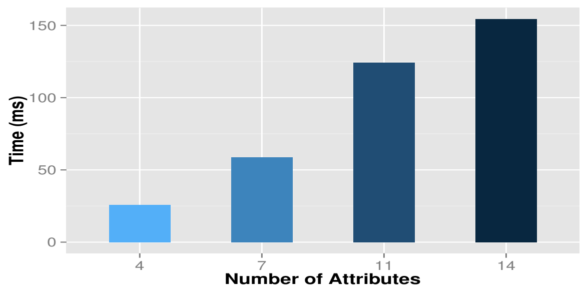

Fig. 5: Access Tree Computation Cost

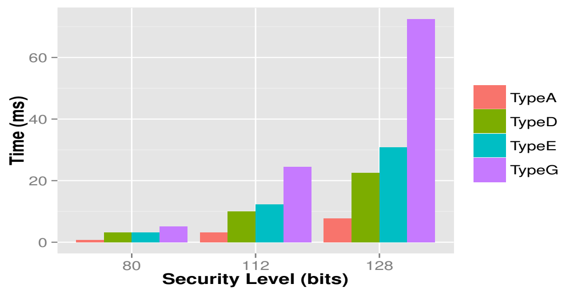

Fig. 7: Pairing Function Computational Cost

tion cost increases to reach 25.53 ms using 14 attributes. Hence, Ins-PAbAC introduces an attractive computation cost, especially for limited capacities devices. We can observe that the difference between the encryption time and the decryption time is considerably noticeable due to the fact that the encryption phase includes also the access structure generation phase. Hence, to investigate the impact of the access structure generation, we conduct experiments in order to evaluate its computation cost as illustrated in Figure 5. We mainly note that the average time needed to generate an access structure increases with the number of attributes. The access structure generation takes 154 ms while considering 14 attributes. We can conclude that the processing overhead is very important due to the heavy task of this algorithm which first takes as input the set of attributes then defines the relation between them.

In order to evaluate the performances of the anonymous user authentication phase which is the second layer of the data backup procedure, we examine the computation cost of the `sign` and the `verif` algorithms (cf. Figure 6) depending on the number of attributes. The `sign` algorithm execution takes less than 15 ms using 4 attributes and approximately 29 ms while increasing the number of attributes to 14.

Besides, the computation cost of the `verif` algorithm begins by 10 ms and raises to 23.88 ms while involving 14 attributes. We mainly notice that computation costs of the `sign` and the `verif` algorithms increases along with the number of attributes.

Above all, these computation overheads remain attractive while providing a user anonymous authentication, flexible access control and better security to outsourced data.

As our accountable Ins-PAbAC framework relies on the use of bilinear maps as well as mathematical operations in a multiplicative group, we investigate the impacts of these operations on the performances of our proposal while considering three security levels (cf. Figure 7, Figure 9 and Figure 8). On one hand, we choose to implement two symmetric pairing functions mainly type A and type E as well as two asymmetric pairing functions, type D and type G [65]. As shown in Figure 7, we notice that type A pairing function is slower than type D pairing function respectively type E and type G. In addition, the processing overheads of the different pairing functions increase along with the security level. For instance, type A pairing function requires 0.76 ms considering a security level equal to 80, however, type D pairing function takes 03.23 ms under the same security level. Besides, while considering a security level equal to 128, type A, type D and type G pairing functions' time durations are equal to 07.79ms, 22.6 ms and 72.46 ms, respectively. As such, the type of the pairing function should be taken into account, while implementing a cryptographic mechanism.

On the other hand, we evaluate the computation cost of multiplication and exponentiation operations as these elementary operations are an important criterion to evaluate the system performances.

Figure 9 and Figure 8 illustrate that the average time of both multiplication and exponentiation operations increases along with the security level. We must notice that these results have been obtained while choosing type A pairing function and fixing three security levels. First, the exponentiation requires 0.112 ms and 0.727 considering security levels equal to 80 and 128, respectively. Second, the multiplication overhead begins from 0.001 ms while fixing a security level equal to 80 and goes up to 0.003 ms considering a security level equal to 128.

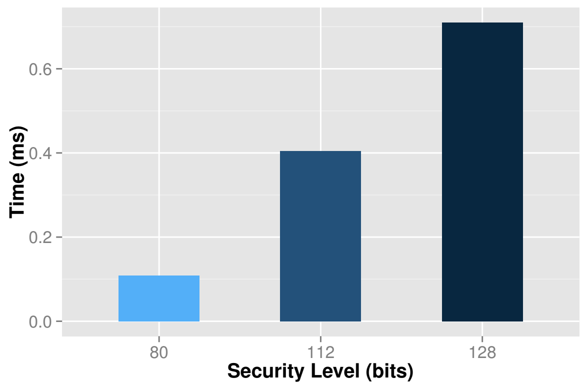

Fig. 8: Exponentiation Computation Costs

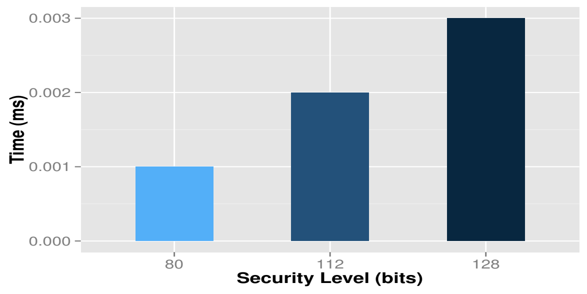

Fig. 9: Multiplication Computation Costs

IX. Conclusions

With the emergence of distributed applications and the enforcement of several laws and regulations aiming at protecting personal data, the main challenging concern is the design of privacy preserving, efficient and secure access control schemes to outsourced data contents on remote distributed servers.

This paper presented Ins-PAbAC, an accountable privacy preserving attribute-based framework that relies on attribute based cryptographic techniques for securely sharing outsourced data contents via public cloud servers. The proposed framework ensures the confidentiality of outsourced data in public untrusted cloud servers and defines efficient data sharing in dynamic groups. That is, flexible access control policies are enforced among users belonging to separate groups with different privileges. Moreover, Ins-PAbAC introduces the accountability feature allowing an inspection authority to reveal the user's identity if needed. In addition, based on formal system and security models, Ins-PAbAC is proven to ensure strong security properties in the random oracle model, namely confidentiality, unforgeability, privacy preservation and anonymity removal. Through experimental evaluation built upon OpenStack Swift testbed, Ins-PAbAC is proven efficient in scalable data sharing, while considering the impact of the cryptographic operations at the client side as well as at the CSP side.

As future work, we aim at exploring decryption outsourcing techniques to be applied on Ins-PAbAC in order to achieve a reasonable computation cost at the data user side.

References

[1] S. Xu, G. Yang, Y. Mu, and X. Liu, "Efficient attribute-based encryption with blackbox traceability," in *International Conference on Provable Security*. Springer, 2018, pp. 182–200.

[2] W. Kong, Y. Lei, and J. Ma, "Data security and privacy information challenges in cloud computing," *International Journal of Computational Science and Engineering*, vol. 16, no. 3, pp. 215–218, 2018.

[3] X. A. Wang, F. Xhafa, J. Ma, and Z. Zheng, "Controlled secure social cloud data sharing based on a novel identity based proxy re-encryption plus scheme," *Journal of Parallel and Distributed Computing*, vol. 130, pp. 153–165, 2019.

[4] N. Kaaniche and M. Laurent, "Data security and privacy preservation in cloud storage environments based on cryptographic mechanisms," *Computer Communications*, vol. 111, pp. 120–141, 2017.

[5] S. Cui, S. Belguith, P. De Alwis, M. R. Asghar, and G. Russello, "Malicious entities are in vain: Preserving privacy in publish and subscribe systems," in *2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*. IEEE, 2018, pp. 1624–1627.

[6] Y. Atwady and M. Hammoudeh, "A survey on authentication techniques for the internet of things," in *Proceedings of the International Conference on Future Networks and Distributed Systems*. ACM, 2017, p. 8.

[7] N. Kaaniche, M. Laurent, P.-O. Rocher, C. Kiennert, and J. Garcia-Alfaro, "Pcs, a privacy-preserving certification scheme," in *Data Privacy Management, Cryptocurrencies and Blockchain Technology*. Springer, 2017, pp. 239–256.

[8] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attribute-based encryption," in *IEEE Symposium on Security and Privacy, 2007.*, 2007, pp. 321–334.

[9] R. Amin, N. Kumar, G. Biswas, R. Iqbal, and V. Chang, "A light weight authentication protocol for iot-enabled devices in distributed cloud computing environment," *Future Generation Computer Systems*, vol. 78, pp. 1005–1019, 2018.

[10] S. Belguith, N. Kaaniche, A. Jemai, M. Laurent, and R. Attia, "Pabac: a privacy preserving attribute based framework for fine grained access control in clouds," in *13th IEEE International Conference on Security and Cryptography(Secrypt)*, 2016, pp. 133–146.

[11] A. Lewko and B. Waters, "Decentralizing attribute-based encryption," in *Advances in Cryptology–EUROCRYPT 2011*. Springer, 2011, pp. 568–588.

[12] " OpenStack Open Source Cloud Computing Software (SWIFT)," http://www.openstack.org/software/, 2016.

[13] X. A. Wang, J. Ma, F. Xhafa, M. Zhang, and X. Luo, "Cost-effective secure e-health cloud system using identity based cryptographic techniques," *Future Generation Computer Systems*, vol. 67, pp. 242–254, 2017.

[14] S. Archondakis, E. Vavoulidis, M. Nasioutziki, O. Oustampasidou, A. Daniilidis, A. Vatopoulou, A. Papanikolaou, and K. Dinas, "Mobile health applications and cloud computing in cytopathology: Benefits and potential," in *Mobile Health Applications for Quality Healthcare Delivery*. IGI Global, 2019, pp. 165–202.

[15] "Health Insurance Portability and Accountability Act (HIPAA)," https://www.hipaa.com/about/, 2015.

[16] Regulation(EU), "2016/679 of the european parliament and of the council of 27 april 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing directive 95/46/ec (general data protection regulation), ojeu l 119/1 of 4.05.2016." 2016.

[17] N. Kaaniche, M. Laurent, and M. El Barbori, "Cloudasec: A novel publickey based framework to handle data sharing security in clouds," in *11th IEEE International Conference on Security and Cryptography(Secrypt)*, 2014.

[18] S. Belguith, N. Kaaniche, M. Mohamed, and G. Russello, "Coop-daab: cooperative attribute based data aggregation for internet of things applications," in *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"*. Springer, 2018, pp. 498–515.

[19] S. Belguith, N. Kaaniche, and M. Hammoudeh, "Analysis of attribute-based cryptographic techniques and their application to protect cloud services," *Transactions on Emerging Telecommunications Technologies*, p. e3667.

[20] X. A. Wang, J. Weng, J. Ma, and X. Yang, "Cryptanalysis of a public authentication protocol for outsourced databases with multi-user modification," *Information Sciences*, vol. 488, pp. 13–18, 2019.

[21] N. Kaaniche and M. Laurent, "Privacy-preserving multi-user encrypted access control scheme for cloud-assisted iot applications," in *2018 IEEE 11th International Conference on Cloud Computing (CLOUD)*. IEEE, 2018, pp. 590–597.

[22] A. Sahai and B. Waters, Fuzzy identity-based encryption, "Fuzzy identity-based encryption," in *EUROCRYPT 2005*. Springer, 2005, pp. 457–473.

[23] V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-based encryption for fine-grained access control of encrypted data," in *The 13th ACM conference on Computer and communications security*, 2006, pp. 89–98.

[24] S. Belguith, N. Kaaniche, M. Mohamed, and G. Russello, "C-absc: Cooperative attribute based signcryption scheme for internet of things applications," in *2018 IEEE International Conference on Services Computing (SCC)*. IEEE, 2018, pp. 245–248.

[25] N. Kaaniche and M. Laurent, "Attribute based encryption for multi-level access control policies," in *SECRYPT 2017: 14th International Conference on Security and Cryptography*, vol. 6. Scitepress, 2017, pp. 67–78.

[26] S. Belguith, N. Kaaniche, and G. Russello, "Pu-abe: Lightweight attribute-based encryption supporting access policy update for cloud assisted iot," in *2018 IEEE 11th International Conference on Cloud Computing (CLOUD)*. IEEE, 2018, pp. 924–927.

[27] S. Belguith, N. Kaaniche, G. Russello *et al.*, "Lightweight attribute-based encryption supporting access policy update for cloud assisted iot," in *Proceedings of the 15th International Joint Conference on e-Business and Telecommunications-Volume 1: SECRYPT*. SciTePress, 2018, pp. 135–146.

[28] Q. Huang, Y. Yang, and M. Shen, "Secure and efficient data collaboration with hierarchical attribute-based encryption in cloud computing," *Future Generation Computer Systems*, vol. 72, pp. 239–249, 2017.

[29] Y. Zhao, P. Fan, H. Cai, Z. Qin, and H. Xiong, "Attribute-based encryption with non-monotonic access structures supporting fine-grained attribute revocation in m-healthcare." *IJ Network Security*, vol. 19, no. 6, pp. 1044–1052, 2017.

[30] J. Zhou, H. Duan, K. Liang, Q. Yan, F. Chen, F. R. Yu, J. Wu, and J. Chen, "Securing outsourced data in the multi-authority cloud with fine-grained access control and efficient attribute revocation," 2017.

[31] J. Wei, W. Liu, and X. Hu, "Secure and efficient attribute-based access control for multiauthority cloud storage," *IEEE Systems Journal*, vol. 12, no. 2, pp. 1731–1742, 2018.

[32] Y. Yang, X. Chen, H. Chen, and X. Du, "Improving privacy and security in decentralizing multi-authority attribute-based encryption in cloud computing," *IEEE Access*, vol. 6, pp. 18 009–18 021, 2018.

[33] V. K. A. Sandor, Y. Lin, X. Li, F. Lin, and S. Zhang, "Efficient decentralized multi-authority attribute based encryption for mobile cloud data storage," *Journal of Network and Computer Applications*, vol. 129, pp. 25–36, 2019.

[34] D. Khader, "Attribute based group signature with revocation." *IACR Cryptology ePrint Archive*, vol. 2007, p. 241, 2007.

[35] H. K. Maji, M. Prabhakaran, and M. Rosulek, "Attribute-based signatures: Achieving attribute-privacy and collusion-resistance." *IACR Cryptology ePrint Archive*, vol. 2008, p. 328, 2008.

[36] J. Li, M. H. Au, W. Susilo, D. Xie, and K. Ren, "Attribute-based signature and its applications," in *Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security*. ACM, 2010, pp. 60–69.

[37] F. Zhao, T. Nishide, and K. Sakurai, "Realizing fine-grained and flexible access control to outsourced data with attribute-based cryptosystems," in *Information Security Practice and Experience*. Springer, 2011, pp. 83–97.

[38] H. K. Maji, M. Prabhakaran, and M. Rosulek, "Attribute-based signatures," in *Topics in Cryptology–CT-RSA*. Springer, 2011, pp. 376–392.

[39] G. Wang, Q. Liu, and J. Wu, "Hierarchical attribute-based encryption for fine-grained access control in cloud storage services," in *The 17th ACM conference on Computer and communications security*. ACM, 2010, pp. 735–737.

[40] S. Yu, C. Wang, K. Ren, and W. Lou, "Achieving secure, scalable, and fine-grained data access control in cloud computing," in *INFOCOM IEEE Proceedings 2010*, 2010, pp. 1–9.

[41] B. Waters, "Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization," in *Public Key Cryptography–PKC 2011*. Springer, 2011, pp. 53–70.

[42] S. Ruj, A. Nayak, and I. Stojmenovic, "Dacc: Distributed access control in clouds," in *IEEE 10th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, 2011, pp. 91–98.

[43] S. Ruj, M. Stojmenovic, and A. Nayak, "Privacy preserving access control with authentication for securing data in clouds," in *The 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid), 2012*. IEEE, 2012, pp. 556–563.

[44] M. Li, S. Yu, Y. Zheng, K. Ren, and W. Lou, "Scalable and secure sharing of personal health records in cloud computing using attribute-based encryption," *IEEE Transactions on Parallel and Distributed Systems*, vol. 24, no. 1, pp. 131–143, 2013.

[45] K. Yang and X. Jia, "Expressive, efficient, and revocable data access control for multi-authority cloud storage," *IEEE transactions on parallel and distributed systems*, vol. 25, no. 7, pp. 1735–1744, 2014.

[46] A. El Kaafarani, E. Ghadafi, and D. Khader, "Decentralized traceable attribute-based signatures," in *Topics in Cryptology–CT-RSA 2014*. Springer, 2014, pp. 327–348.

[47] S. Ruj, M. Stojmenovic, and A. Nayak, "Decentralized access control with anonymous authentication of data stored in clouds," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 2, pp. 384–394, 2014.

[48] M. Horváth, "Attribute-based encryption optimized for cloud computing," in *International Conference on Current Trends in Theory and Practice of Informatics*. Springer, 2015, pp. 566–577.

[49] Z. Liu, H. Yan, and Z. Li, "Server-aided anonymous attribute-based authentication in cloud computing," *Future Generation Computer Systems*, vol. 52, pp. 61–66, 2015.

[50] H. Hong and Z. Sun, "An efficient and traceable kp-abs scheme with untrusted attribute authority in cloud computing," *Journal of Cloud Computing*, vol. 5, no. 1, p. 1, 2016.

[51] S. Belguith, N. Kaaniche, M. Laurent, A. Jemai, and R. Attia, "Phoabe: Securely outsourcing multi-authority attribute based encryption with policy hidden for cloud assisted iot," *Computer Networks*, vol. 133, pp. 141–156, 2018.

[52] M. H. Ibrahim, S. Kumari, A. K. Das, and V. Odelu, "Attribute-based authentication on the cloud for thin clients," *The Journal of Supercomputing*, pp. 1–33, 2018.

[53] H. Cui, R. H. Deng, and G. Wang, "An attribute-based framework for secure communications in vehicular ad hoc networks," *IEEE/ACM Transactions on Networking*, 2019.

[54] Y. S. Rao and R. Dutta, "Efficient attribute-based signature and signcryption realizing expressive access structures," *International Journal of Information Security*, vol. 15, no. 1, pp. 81–109, 2016.

[55] N. Kaaniche and M. Laurent, "Attribute-based signatures for supporting anonymous certification," in *European Symposium on Research in Computer Security*. Springer, 2016, pp. 279–300.

[56] M. Chase and S. S. Chow, "Improving privacy and security in multi-authority attribute-based encryption," in *Proceedings of the 16th ACM conference on Computer and communications security*. ACM, 2009, pp. 121–130.

[57] D. H. Phan and D. Pointcheval, "On the security notions for public-key encryption schemes," in *International Conference on Security in Communication Networks*. Springer, 2004, pp. 33–46.

[58] B. Waters, "Efficient identity-based encryption without random oracles," in *Advances in Cryptology–EUROCRYPT 2005*. Springer, 2005, pp. 114–127.

[59] D. Boneh, B. Lynn, and H. Shacham, "Short signatures from the weil pairing," in *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, 2001, pp. 514–532.

[60] L. Li, X. Chen, H. Jiang, Z. Li, and K.-C. Li, "P-cp-abe: Parallelizing ciphertext-policy attribute-based encryption for clouds," in *Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD), 2016 17th IEEE/ACIS International Conference on*. IEEE, 2016, pp. 575–580.

[61] J. Li, F. Sha, Y. Zhang, X. Huang, and J. Shen, "Verifiable outsourced decryption of attribute-based encryption with constant ciphertext length," *Security and Communication Networks*, vol. 2017, 2017.

[62] S. Belguith, N. Kaaniche, and G. Russello, "Cups: Secure opportunistic cloud of things framework based on attribute based encryption scheme supporting access policy update," *Security and Privacy*, 2019.

[63] "The openssl project," https://www.openssl.org/, 2016.

[64] "Gmp library," http://http://gmplib.org/, 2016.

[65] B. Lynn, "On the implementation of pairing-based cryptosystems," Ph.D. dissertation, Stanford University, 2007.