PAbAC: A Privacy Preserving Attribute based Framework for Fine Grained Access Control in Clouds

Sana Belguith[1,2], Nesrine Kaaniche[3], Abderrazak Jemai[4], Maryline Laurent[3] and Rabah Attia[1,2]

[1]*Tunisia Polytechnic School, Laboratory of Electronic Systems and Communication Network, Tunis, Tunisia*

[2]*Telnet Holding, Telnet Innovation Labs, Tunis, Tunisia*

[3]*SAMOVAR, CNRS, Telecom SudParis, University Paris-Saclay, Paris, France*

[4]*Laboratory LIP2, University of Sciences of Tunis, Tunis, Tunisia*

Keywords: Cloud Storage Systems, Attribute-based Encryption, Attribute-based Signature, Data Confidentiality, Privacy.

Abstract: Several existing access control solutions mainly focus on preserving confidentiality of stored data from unauthorized access and the storage provider. Moreover, to keep sensitive user data confidential against untrusted servers, existing solutions usually apply cryptographic methods by disclosing data decryption keys only to authorized users. However, these solutions inevitably introduce a heavy computation overhead on the data owner for key distribution and data management when fine-grained data access control is desired. In addition, access control policies as well as users' access patterns are also considered as sensitive information that should be protected from the cloud. In this paper, we propose PAbAC, a novel privacy preserving Attribute-based framework, that combines attribute-based encryption and attribute-based signature mechanisms for securely sharing outsourced data via the public cloud. Our proposal is multifold. First, it ensures fine-grained cryptographic access control enforced at the data owner's side, while providing the desired expressiveness of the access control policies. Second, PAbAC preserves users' privacy, while hiding any identifying information used to satisfy the access control. Third, PAbAC is proven to be highly scalable and efficient for sharing outsourced data in remote servers, at both the client and the cloud provider side.

1 INTRODUCTION

Data security and privacy are major challenges in the adoption of cloud storage applications, mainly due to the loss of data control. It is commonly agreed that data encryption at the client side is a good alternative to mitigate data secrecy concerns. As such, the client preserves the decrypting keys out of reach of the cloud. Although encryption assures the confidentiality against curious cloud service providers, the use of conventional encryption approaches is not sufficient to support the enforcement of fine-grained access control policies. That is, data confidentiality preservation becomes more complicated, considering flexible data sharing among dynamic groups of users. First, access control policies should be flexible and distinguishable among users with different privileges. Second, dynamic groups require efficient sharing of deciphering keys between different authorized users. In fact, the subscription of a new group member should not require updating the secret keys of the remaining users. So that, the complexity of key management is minimized. Thus, the challenge is to define a comprehensive access control mechanism for outsourced data while both ensuring data confidentiality and protecting users' privacy.

For instance, with the involvement of a third-party cloud provider, a crucial issue is that access patterns may reveal privacy-sensitive information about users and potentially leak confidential information about the content. The confidentiality of outsourced data and the privacy of users are thus not assured if these sensitive data are not protected.

In this paper, we propose PAbAC, a novel privacy preserving Attribute-based framework, that combines Attribute Based Encryption (ABE) and Attribute Based Signature (ABS) mechanisms for securely sharing outsourced data via public clouds. PAbAC introduces a two-level access control model that combines *fine-grained access control* which ensures a comprehensive granularity for access rules, and *anonymous data access*, which allows the storage server to manage access requests with no need to learn the user identity nor his attributes. The original-

ity of PAbAC is multifold. First, PAbAC introduces a privacy preserving authentication scheme, based on a novel use of attribute-based signatures.The identity of the requesting entity client remains protected against the certifying authorities as well as the cloud service provider. Moreover, the combination between attribute based encryption mechanisms and attribute based signature scheme allows the cloud provider to control the bandwidth consumption, and then, the system's availability. In fact, the authentication of requesting users permits to mitigate *Flooding attacks* which exploit the bandwidth under provisioning vulnerability (Zunnurhain, 2012). Thus, only authorized users can download encrypted data.
Second, as a designed decentralised multi-authority attribute based framework, PAbAC reduces the bottleneck of maintaining a central authority for managing secret parameters. Additionally, it supports issuing a set of attributes from each attribute authority unlike other access control schemes which rely on issuing only one attribute per authority.
Third, the original use of attribute based encryption and the related attribute based signature guarantees fine grained access control to outsourced data and provides an effective key management in sharing scenarios. For instance, the PAbAC framework is highly scalable and offers interesting performances such as low computation and communication cost, at both the client and the cloud provider side.
Paper Organisation – The remainder of this work is as follows: Section 2 presents security considerations and design goals. Then, Section 3 reviews related work and introduces attribute based mechanisms. In Section 4, we describe the system model and review some preliminaries and cryptographic primitives. Afterwards, we detail the framework design and describe the prototype and its different procedures in Section 5. In Section 6, rigorous security discussions are given. Finally, theoretical performances analysis is provided in Section 7, before concluding in Section 8.

2 PROBLEM STATEMENT

Let us consider the following example, where a hospital supports fine-grained access control on Electronic Health Records (EHRs) and makes these records available to hospital employees through a public cloud. In accordance with regulations such as the Health Insurance Portability and Accountability Act (HIPAA) (HIP,), the hospital policies must specify which users can access which data item(s). In fact, a health-care information system based on cloud services is required to restrict access of protected medical records to eligible doctors while a client relation management system running on a cloud may allow access of patients' information to high-level executives of the hospital only. In many cases, hospital employees, mainly doctors, have to share patients' health information, in order to properly prescript treatments. Thus, they usually form dynamic sharing groups with different granted privileges. Therefore, it is noteworthy that data confidentiality preservation is not the only security concern. It is crucial to support flexible sharing of encrypted outsourced data among dynamic group of users, while protecting users' privacy. In a real e-health scenario, different medical organisations can be involved such as hospitals, research laboratories, pharmacies, health ministry as well as doctors and patients. Let us consider that a doctor shares his patients' EHR in a public cloud. On one hand, the shared data have to be protected from unauthorized access while ensuring fine grained access control for different authorized actors. Moreover, the data confidentiality must be preserved against a malicious cloud service provider. Thus, encryption on the client side should be applied while supporting flexible sharing of outsourced data among dynamic group of users.
On the other hand, the private identifying information of the involved cloud users, such as doctors and patients, must not be revealed to the remote server. For instance, the system should not reveal any private information related to a doctor, such as his professional card or its speciality, as well as his patients' data. That is, the disclosure of such information may be used to produce targeted advertisement related to the health condition of the patients, or to run statistical surveys. Thus, the design of PAbAC is motivated by providing the support of both robustness and efficiency while fulfilling the following properties:

- ***Data Confidentiality –*** PAbAC has to protect the secrecy of outsourced and encrypted data contents against both curious cloud service providers and malicious users.
- ***Flexible Access Control –*** our proposal should ensure flexible security policies among dynamic groups of users with different granted privileges, belonging to different groups.
- ***Privacy –*** PAbAC must protect group members' access patterns privacy, while requesting access to outsourced data. That is, the cloud server must be able to grant access with no need to additional identifying information of the requesting users.
- ***Low Computation Overhead –*** on one hand, for scalability reasons, the amount of computation at the cloud storage server should be minimized, as

the server may be involved in concurrent interactions. On the other hand, the proposed algorithms should also have low processing complexity, at the client side.

- ***Low Storage Cost –*** PAbAC should provide acceptable storage cost at the client side.

3 RELATED WORK

Several research works have been proposed in the literature in order to securely share data among groups of users while protecting their privacy (Horváth, 2015),(Kaaniche et al., 2014),(Wan et al., 2012),(Yu et al., 2010a), (Raykova et al., 2012), (Di Vimercati et al., 2010b). In order to prevent untrusted servers from accessing outsourced data, several solutions apply encryption mechanisms at the client side while disclosing the decryption keys to authorized users only (Benaloh et al., 2009),(Di Vimercati et al., 2010a), (Kaaniche et al., 2013), (Benaloh et al., 2009), (Di Vimercati et al., 2015). Although these methods ensure secure data access control, the key distribution remains a bottleneck. For instance, it becomes more complicated with the increase of the number of users. To deal with this concern, Wang et al. (Wang et al., 2009) propose to deliver the key management to the remote server while assuming that this latter is trusted. Moreover, in order to enforce authorization policies, De Vimercati et al. (Di Vimercati et al., 2007) proposed a novel solution aiming to enforce the access control to the outsourcing systems. De Vimercati et al. proposal is based on the application of selective encryption as a means to enforce authorizations while applying hierarchical key assignment schemes.
Recently, Attribute-based Cryptography appears as a promising technique, designed for ensuring fine grained access control for outsourced data. This cryptographic mechanism was introduced by Sahai and Waters in 2005 (Sahai and Waters, 2005).
In the following, we present Attribute based Encryption mechanisms (ABE) and their application in cloud environments in Section 3.1. Then, we introduce Attribute based Signature schemes (ABS) and review related work applying ABS for protecting access to outsourced data in cloud servers.

3.1 Attribute based Encryption (ABE)

In 2005, Sahai and Waters introduced the concept of Attribute Based Encryption (ABE), as a new mean for encrypted access control (Sahai and Waters, 2005). In ABE, ciphertexts are not necessarily encrypted to one particular user as in traditional public key cryptography. Instead both users' private keys and ciphertexts are associated with a set of attributes or a structure over attributes (Bethencourt et al., 2007). The user is able to decrypt a ciphertext if there is a match between his private key and the ciphertext. For instance, Goyal et al. distinguishes two ABE categories, namely: Key-Policy ABE (KP-ABE) (Goyal et al., 2006) and Ciphertext-Policy ABE (CP-ABE) (Bethencourt et al., 2007). Several works rely on ABE to provide fine grained access control for outsourced data (Hur and Noh, 2011),(Yu et al., 2010b),(Jahid et al., 2011). Although these schemes proposed efficient solutions to protect outsourced data, they require the use of a central trusted authority to manage all the attributes and issue the related secret keys to users in the system. Thus, this central authority is able to achieve a key escrow attack, due to its knowledge of the users' private keys.
Wang et al. (Wang et al., 2010) propose a hierarchical access control mechanism for cloud storage. Their scheme is based on the Bethencourt et al. CP-ABE scheme (Bethencourt et al., 2007) and hierarchical Identity based Encryption (IBE) (Horwitz and Lynn, 2002). This scheme relies on the use of several authorities arranged in a hierarchical way. However, it still relies on the trusted authority and fails, if the latter is compromised.
Recently, Lewko and Waters (Lewko and Waters, 2011) proposed a decentralized ABE scheme, where users could obtain their private keys from different attribute authorities. Each attribute authority is in charge for deriving a private key associated to a one attribute. The proposal (Lewko and Waters, 2011) did not require a central trusted server, which must remain active and uncorrupted throughout the lifetime of the system. Based on this decentralized architecture, there is no need for absolute trust in a single designated entity. However, Lewko and Waters (Lewko and Waters, 2011) assume that each attribute authority is responsible for issuing only one attribute. Moreover, in order to prevent collusion in such a setting, this scheme requires that each user has a unique global identifier (*GID*), which they must present to each authority. Unfortunately, due to the use of *GID*, the users cannot preserve their privacy against attribute authorities. In fact, while a user must present the same *GID* to each authority, colluding authorities can pool their data and build a *complete profile* of all of the attributes corresponding to each *GID*. However, this might be undesirable, particularly if the user uses the ABE system in many different settings and wants to keep information about some of those settings private. Based on the Lewko and Waters pro-

posal (Lewko and Waters, 2011), Ruj et al. (Ruj et al., 2011) proposed a distributed sharing scheme in cloud environments scheme with an attribute revocation extension (DACC). The DACC solution consists of using one or several key distribution centers responsible for issuing keys to data owners and users related to their attributes. The data owner encrypts data under an access structure and stores them in the cloud. The users, while matching their set of attributes to the access structure, can retrieve the data from the cloud. We must note that the DACC proposal also supports revocation of users, without redistributing keys to all the users of cloud services. In 2015, Horvá (Horváth, 2015) proposed a decentralized ABE scheme for securely sharing data in cloud computing systems based on the use of the decentralized Lewko and Waters scheme (Lewko and Waters, 2011). In order to achieve an efficient revocation scheme, this proposal relies on the use of an identity based user revocation mechanism to manage access rights for outsourced data. This proposed extension supports multiple independent attribute authorities in which revocation of specific users (e.g. based on users' identities) from the system is possible without updates of attribute public and secret keys.

Most of the mentioned approaches do not propose a mechanism to authenticate requesting users. Moreover, these schemes are based on the use of Lewko et al. decentralised ABE scheme (Lewko and Waters, 2011) which requires the use of the users' global identifiers *GIDs*. Thus, the privacy of the user is not protected against the attribute authorities.

3.2 Attribute based Signature (ABS)

Attribute-Based Signatures (ABS) (Maji et al., 2011) is a flexible primitive that enables a user to sign a message with fine grained access control over identifying information. In ABS, the user possesses a set of attributes, obtained from a trusted authority. This latter can sign a message with respect to a predicate satisfied by his attributes. The signature reveals no more that the fact that a single user with some set of attributes satisfying the predicate has attested to the message. Maji et al. presented a comparison of ABS with other signature-related-notions (Maji et al., 2011), such that ring signatures (Rivest et al., 2001) and group signatures (Chaum and Van Heyst, 1991) that can be considered as particular categories of attribute based signatures (El Kaafarani et al., 2014a). In (Maji et al., 2011), Maji et al. also introduced different applications of ABS including attribute-based messaging (Bobba et al., 2006), trust negotiation (Frikken et al., 2006) and leaking secrets. Some constructions of ABS consider multiple authorities while others only support a single attribute authority. Okamoto et al. (Okamoto and Takashima, 2013) and El Kaafarani et al. (El Kaafarani et al., 2014b) have proposed the first fully decentralized attribute based signatures schemes. These schemes consist of involving multiple attribute authorities in the system, with no reliance on a central authority. The security of attribute based signatures requires users' privacy and unforgeability. On one hand, users' anonymity requires that signatures reveal neither users' identities nor the attributes used in the signing algorithm. On the other hand, unforgeability requires that a user cannot forge a signature with respect to a signing predicate that the user attributes do not satisfy, even if this user colludes with other users (Ghadafi, 2015).

Several works rely on the attribute based signature to ensure the authentication of data owners and fine grained access control to outsourced data in the cloud. Indeed, Ruj et al. (Ruj et al., 2012) presented a privacy preserving authenticated access control scheme for securing data in clouds based on an attribute based signature scheme. In the proposed scheme, the cloud provider verifies the authenticity of the data owner without knowing the user's identity before storing information. To do so, this scheme uses a combination of the decentralized attribute based encryption (Lewko and Waters, 2011) and the multi-authority attribute based signature proposed by Maji et al. (Maji et al., 2011). This proposal relies on the use of a global identifier *GID* in order to issue private keys from attribute authorities to users, thus the attribute authorities can reveal the user's identity. Finally, the proposed authentication scheme is used to authenticate the data owner and there is no way to authenticate the requesting users. In (Zhao et al., 2011), Zhao et al. applied the ciphertext-policy attribute based encryption (CP-ABE) proposed by Bethencourt et al. (Bethencourt et al., 2007) combined with the Maji et al. (Maji et al., 2011) attribute based signature to ensure fine grained access control to outsourced data in the cloud. This proposal does not take interest in authenticating the requesting users. Moreover, it 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.

4 PAbAC SYSTEM

In this section, we present our system model in section 4.1 and the mathematical background is presented in section 4.2.

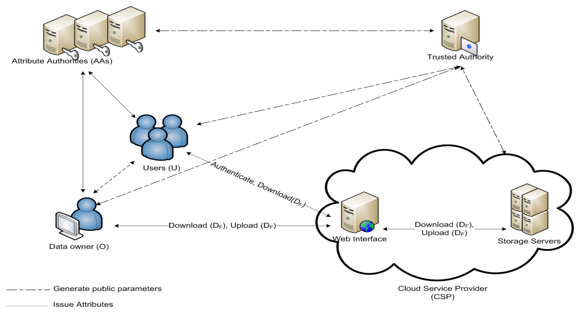
Attribute Authorities (AAs)
Trusted Authority
Users (U)
Authenticate, Download(D_F)
Data owner (O)
Download (D_F), Upload (D_F)
Web Interface
Download (D_F), Upload (D_F)
Storage Servers
Generate public parameters
Issue Attributes
Cloud Service Provider (CSP)

Figure 1: The main architecture entities and their interaction.

4.1 System Model

Figure 1 presents the different entities involved in PAbAC and defined as follows:
Cloud Service Provider (CSP): the CSP is assumed to have abundant storage capacity and computation resources in order to manage data storage service. The cloud provider consists of data servers and a data service manager. Data servers are responsible for storing data outsourced by the data owner while the data manager is in charge of controlling the accesses from requesting users to outsourced data and providing corresponding contents services.
Data Owner (O): the data owner is responsible for outsourcing data into the remote cloud servers and granting access privileges to other cloud users, by specifying an attribute-based access policy for each data file.
Users (U): a user is a cloud client that may access to outsourced content by data owners. In a nutshell, if a user has the set of attributes satisfying the access structure of the encrypted data file, he then may have access to data.
Trusted Authority (TA): TA is a trusted third party in our system which is responsible for generating and managing public parameters for both used mechanisms: attribute based encryption and attribute based signature.
Attribute Authority (AA): AA is a party responsible for deriving a public keys and issuing private keys to different users that are assigned to their attributes. The Attribute Authority can be considered as an Identity Provider. In PAbAC, any trusted party can become an attribute authority and there is no requirement for any global coordination other than the creation of an initial set of common public parameters.

4.2 Mathematical Background

In this section, we first introduce the access structure in section 4.2.1. Then, in section 4.2.2, we present the bilinear maps. Finally, we introduce our security assumptions.

4.2.1 Access Policies

Access policies can be represented by one of the following formats: i) Boolean functions of attributes, ii) Linear Secret Sharing Scheme (LSSS) matrix, or iii) Monotone span programs (Lewko and Waters, 2011).

Definition 1. *Access Structure*

Let $\{P_1, \cdots, P_n\}$ be a set of parties. A collection $\mathtt{A} \subseteq 2^{\{P_1,\cdots,P_n\}}$ is monotone if $\forall B, C$ if $B \in \mathtt{A}$ and $B \subseteq C$ then $C \in \mathtt{A}$ (Lewko and Waters, 2011).
An access structure is a collection $\mathtt{A}$ of non-empty subsets of $\{P_1, \cdots, P_n\}$, such as $\mathtt{A} \subseteq 2^{\{P_1,\cdots,P_n\}} \backslash \emptyset$.
We note that any access structure can be converted into a Boolean function. Boolean functions can be represented by an access tree, where the leaves present the attributes while the intermediate and the root nodes are the logical operators AND ($\wedge$) and OR ($\vee$).

Definition 2. *Linear Secret Sharing Schemes (*LSSS*)*

A Linear Secret Sharing Scheme LSSS over a set of parties P (Lewko and Waters, 2011) is defined as follows:

1. the shares of each party form a vector over $\mathtt{Z}_p$.
2. there exists a matrix A with n rows and l columns called the share-generating matrix for LSSS. $\forall i \in [1, \cdots, n]$, the i^{th} row of A is labeled by a party $\rho(i)$ (where ρ is function from $\{1, \cdots, n\}$ to P). When we consider the column vector $\vec{v} = [s, r_2, \cdots, r_n]$, where $s \in \mathtt{Z}_p$ is the secret to be shared, and $r_2, \cdots, r_n \in \mathtt{Z}_p$ are randomly chosen, then $A.\vec{v} = \vec{\lambda}$ is the vector n shares of the secret s according to LSSS.

In (Beimel, 1996), Beimel presents the algorithm that converts a boolean function (in the form of access tree) as a LSSS matrix.

Definition 3. *Monotone Span Programs*

For a field F and a variable set $S = \{a_1, ..., a_n\}$, a Monotone Span Program (Karchmer and Wigderson, 1993) is defined by a $\alpha \times \beta$ matrix A along with a labeling map ρ which associates each row in A with an element $a_i \in S$. The span program accepts a set γ if $1 \in Span(A_\gamma)$, where (A_γ) is the sub-matrix of A containing only rows with labels $a_1 \in \gamma$. In other words, the span program only accepts the set γ if there exists a vector s such that $s \cdot A_\gamma = [1, 0, ..., 0]$.

4.2.2 Bilinear Maps

An admissible symmetric pairing function $\hat{e}$ from $\mathbb{G}_1 \times \mathbb{G}_1$ in $\mathbb{G}_T$ has to be bilinear, non degenerate and efficiently computable. $\mathbb{G}_1$ and $\mathbb{G}_T$ are two multiplicative subgroups of a finite field. $\mathbb{G}_1$ and $\mathbb{G}_T$ have the same order N.

4.2.3 Complexity Assumptions

Our proposal is based on three cryptographic assumptions detailed as follows:

Definition 1. ***Discrete Logarithm Problem (DLP)***

Given a generator g of a multiplicative cyclic group $\mathbb{G}$ of order N, and given the public element $y = g^x \in \mathbb{G}$, the problem of finding x is called the Discrete Logarithm Problem.

Definition 2. ***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 3. ***Decisional Diffie Hellman problem (DDH)***

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 distinguishing between tuples of the form (g^a, g^b, g^{ab}) and (g^a, g^b, g^c) for some random integer c, is called the Decisional Diffie Hellman problem.

5 PAbAC: PROPOSED SOLUTION

5.1 Motivation

In order to achieve fine grained and privacy preserving access control to outsourced data in cloud storage, we combine two latest cryptographic techniques, CP-ABE and ABS. The choice of attribute based cryptography (ABC) is motivated by several reasons. First, we benefit from an easier key management system, thanks to the certificate-free feature of ABC. Second, ABC permits deriving public keys with no need for previous computation of corresponding private keys. That is, contrary to traditional public key derivation schemes, ABC does not require to generate the private key before the public key. Indeed, users have only to generate access structure and the related enciphering key to encrypt data before storage.
CP-ABE is much more appropriate to data outsourcing, since it enables the data owner to generate an access tree over selected attributes. Thanks to its flexibility in specifying different access rights for each individual user, ABE is considered as one of the most public key primitive which is appropriate for one-to-many communications. That is, data are encrypted under a set of attributes so that multiple users who possess proper keys can decrypt. This potentially makes encryption and key management more efficient. Moreover, the enciphering entity is not required to know the access control list.
In order to protect the requesting entity's privacy, PAbAC relies on using Attribute based Signature (ABS). In ABS, messages are signed with respect to an access structure. Thus, the CSP verifies that the requesting user having a set of attributes satisfying the access tree has indeed authenticated the message without revealing his identity or the set of attributes used in the signing procedure.
Our PAbAC framework is based on an original use of the identity based signature scheme presented by Waters (Waters, 2005) combined with the decentralized Attribute Based Encryption introduced by Lewko and Waters (Lewko and Waters, 2011) to achieve an extension of waters' scheme to a multi-authority attribute based encryption. This novel encryption scheme presented by PAbAC supports the issuance of a set of attribute obtained from the same authority. Moreover, PAbAC introduces an original multi-authority attribute based signature scheme based on an extension of Waters' identity based signature adapted to the multi-authority ABE encryption. Thus, PAbAC presents lower computation costs especially at the client side compared with the other access control schemes. The different notations used in this paper are listed in Table 1.

Table 1: The different notations used in this paper.

Notation	Description
S_U	Set of users' attributes
S_j	Set of attributes certified by the attribute authority AA_j
Sk_{S_j}	Secret keys related 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
ψ	Access policy

5.2 Overview

In PAbAC, there are two main actors: a data owner (O) and data users (U). The data owner 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. Then, the data owner stores the encrypted data in the cloud. When a user 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.
We suppose that each cloud user has already obtained the private keys related to his attributes from the corresponding attribute authorities. For an e-health use case, an attribute authority may be the hospital administration issuing the affiliation card of each doctor (i.e the professional card contains a set of attributes such as the name of the doctor, his affiliation, his serial number, $\cdots$).
Based on the required attributes, specified in the access structure ψ, the requesting user selects related private keys in order to decrypt the encrypted data file.
Our PAbAC proposal is defined upon the following seven algorithms. It involves three procedures on the basis of two 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.
The SYS_INIT procedure consists of three randomized algorithms for the generation of public parameters related to the involved attribute authorities referred to as `setup` and `setup`$_{\texttt{auth}}$, and the generation of users' private keys denoted by `keygen`.
The STORE procedure presents the data storage scenario. It consists of the `encdata` algorithm for the encryption of data files.
For the data retrieval scenario, the BACKUP procedure deals with the user' authentication, namely `sign` and `verif` and the data decryption algorithms referred to as `decdata`.

5.3 System Initialisation Procedure

The SYS_INIT procedure consists of three randomized algorithms, defined as follows:

- `setup` – this randomized algorithm takes as input the security parameter λ. It outputs the global public parameters PP defined as follows:

$$\mathtt{PP} = \{\mathbb{G}_1, \mathbb{G}_T, N, h, \hat{e}, u_0, \cdots, u_n\}$$

  where $\mathbb{G}$ and $\mathbb{G}_1$ are two multiplicative groups of order N, $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_T$ is a bilinear map, g, h are generators of $\mathbb{G}_1$ and $\{u_0 = g, \cdots, u_n\}$ are generators of $\mathbb{G}_1$ randomly chosen such as $u_i = g^{r^i}$.

- `setup`$_{\texttt{auth}}$ – the `setup`$_{\texttt{auth}}$ algorithm is executed by an attribute authority AA. The `setup`$_{\texttt{auth}}$ algorithm takes as inputs the public parameters PP and outputs the pair of private and public keys (sk_A, pk_A), where sk_A correspond to a random values α, and the related public key pk_A is defined as follows:

$$pk_A = \{\hat{e}(g_1, g_1)^{\alpha}\}$$

- `keygen` – this algorithm is performed by an attribute authority AA_j. It takes as input the global parameters PP, the attribute authority's secret key $\{sk_A\}$, a random value t where $t \in \mathbb{Z}_p$ and a set of attributes $S_j = \{a_1, \cdots, a_{n_j}\}$, where n_j is the number of attributes of S_j. It outputs the secret key sk_{S_j} related to the set of attributes S_j, as depicted by Algorithm 1.

Algorithm 1: `Keygen` procedure.

```
1: **Input:** the global parameters PP, the attribute authority's secret key $\{sk_{A_j}\}$ and a set of attributes $S_j$
2: **Output:** the secret key $sk_{S_j}$ related to the set of attributes $S_j$
3: $K \leftarrow {g_1}^{\alpha} \cdot h^t$;
4: $L \leftarrow {g_1}^t$;
5: $sk_{S_j} \leftarrow \{K, L\}$;
6: for all $i \in [1 \dots n_j]$ do
7:     $K_i \leftarrow {u_i}^t$;
8:     $sk_{S_j} \leftarrow sk_{S_j} \cup K_i$;
9: end for
10: return $sk_{S_j}$
```

5.4 Data Storage Procedure

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 as detailed in Section 4.2.
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 certified Attribute authority (AA) that is responsible

of issuing the required attributes.
After defining the access structure (A,ρ), the data owner encrypts the data file D_F, based on the `encdata` algorithm. We note that our encryption algorithm relies on the decentralized Lewko and Waters ABE scheme (Lewko and Waters, 2011). That is, we extend the (Lewko and Waters, 2011) proposal to support deriving a set of private keys related 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` algorithms, defined as follows:

- `encdata` – the encryption algorithm `encdata` is executed by the data owner. It takes as input the attribute authorities' public keys $\{pk_A\}$, the data file D_F, the public parameters PP and the access policy (A,ρ). 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 attribute i) defined as follows:
$$C_0 = D_F \hat{e}(g_1,g_1)^s \quad (1)$$
$$C_{1,i} = \hat{e}(g_1,g_1)^{\lambda_i} \quad (2)$$
$$C_{2,i} = {g_1}^{p_i} \quad (3)$$
$$C_{3,i} = {g_1}^{p_i}{g_1}^{w_i}{u_i}^{p_i} \quad (4)$$
Where $p_i, s \in \mathbb{Z}_N$ are random values selected by the data owner, $\lambda_i = \vec{A_i}\cdot\vec{v}$ where $\vec{v} \in {\mathbb{Z}_N}^l$ is a random vector with s as its first entry and $w_i = \vec{A_i}\cdot\vec{\tau}$ such as $\vec{\tau} \in {\mathbb{Z}_N}^l$ is a random vector with 0 as its first entry.

Figure 2 depicts the storage procedure STORE of the PAbAC framework.

5.5 Data Backup Procedure

For the data retrieval scenario, the BACKUP procedure starts with the user' authentication, with respect to the `sign` and `verif` algorithms and is achieved by the data decryption algorithm referred to as `decdata`. The figure 3 presents different interactions between the cloud provider and the requesting user for the data access procedure.

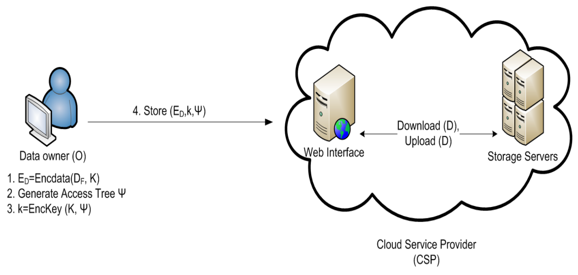

Figure 2: PAbAC data storage procedure STORE.

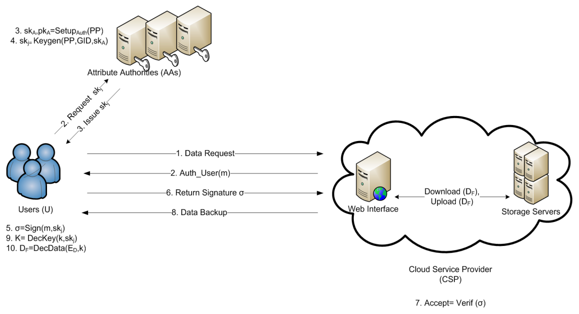

Figure 3: PAbAC data backup procedure BACKUP.

We detail, in Section 5.5.1, the different algorithms for user authentication with the cloud provider upon requesting access to an outsourced data file E_D. Then, we present, in Section 5.5.2, the algorithm needed for decrypting the outsourced data file.

5.5.1 Anonymous User Authentication

When a user (U) wants to access to the encrypted data file (E_D) outsourced by the data owner, the CSP has to first authenticate the user, with respect to the access tree ψ 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 signing predicate ψ, 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` – this algorithm takes as input the global public parameters PP, a random token m, a signing policy ψ and the set of attributes' secret keys $\{sk_{S_j}\}$ that satisfies the signing predicate. It outputs a signature σ.
In fact, the user first selects the sub-set of his attributes S_U that satisfies the signing predicate ψ, 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 $\rho : [n] \rightarrow S_U$. 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 randomizes his secret key sk_{S_j} as follows:

$$K' = Kh^{t'} = g_1{}^{\alpha} h^{t+t'} \quad (5)$$

$$L' = Lg_1{}^{t'} = g_1{}^{t+t'} \quad (6)$$

$$K'_i = K_i u_i{}^{t'} = u_i{}^{t+t'}, \forall i \in S_U \quad (7)$$

In the sequel, the requesting user's new private key is defined by $\{sk'_{S_j}\}$ such as $\{sk'_{S_j}\} = \{(K', L', K'_i)\}$.
Then, for each $i \in [1,n]$, the signer computes $x_i = (L')^{y_i}$ and sets $z = \prod_{i=1}^{n}((K'_{\rho(i)})^{y_i}$. Afterwards, the signer generates a random $r \in \mathbb{Z}_N$ and computes:

$$\sigma_1 = zK' g_1{}^{mr} \quad (8)$$

$$\sigma_2 = g_1{}^{r} \quad (9)$$

Finally, the signature for the message m generated by the user with respect to the signing policy (A, ρ) is set as follows:

$$\sigma = (x_1, \cdots, x_n, \sigma_1, \sigma_2) \quad (10)$$

- `verif` – this algorithm is a deterministic algorithm which takes as input an ordered list of attribute authorities' public keys $\{pk_A\}$, a random token m, a signature σ and a signing predicate ψ corresponding to (A, ρ) and outputs `accept` if σ is valid on m using the access policy (A, ρ) or `reject` otherwise. Afterwards, the CSP computes the vector $\vec{\beta} = [\beta_1 = 1, \beta_2, \cdots, \beta_n]$, 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 the following equation holds.

$$\hat{e}(g_1, \sigma_1) \overset{?}{=} pk_A \hat{e}(g_1{}^m, \sigma_2) \prod_{i=1}^{n} \hat{e}(h^{\mu_i} u_{\rho(i)}, x_i) \quad (11)$$

The correctness of the signature algorithm is detailed in the Section 6.4.

5.5.2 Data Retrieval

The data retrieval procedure consists of `decdata` algorithm, defined as follows:

- `decdata` – this algorithm takes as input the user secret decryption key $\{sk_{S_j}\}$, the public parameters `PP` and the ciphertext E_D and outputs the original data file D_F. If the requesting user has the required private keys $\{sk_{S_j}\}$ for a subset of rows A_i of A such that $[1, 0, \cdots, 0]$ is in the span of these rows, then the user proceeds as follows.
For each matrix row i, the user computes:

$$\frac{C_{1,i}.\hat{e}(L, C_{3,i})}{\hat{e}(LK_i, C_{2,i})} = \hat{e}(g_1, g_1)^{\lambda_i} \hat{e}(g_1{}^t, g_1)^{w_i} \quad (12)$$

$$\prod_i (\hat{e}(g_1, g_1)^{\lambda_i} \hat{e}(g_1{}^t, g_1)^{w_i})^{c_i} = \hat{e}(g_1, g_1)^s \quad (13)$$

Where $c_i \in \mathbb{Z}_N$ are constants such that $\sum_i c_i \vec{A}_i = [1, 0, \cdots, 0]$.
Then, the data file D_F can then be obtained as follows:

$$D_F = C_0 / \hat{e}(g_1, g_1)^s \quad (14)$$

The proof of correctness of the decryption algorithm is detailed in the Section 6.4.

6 SECURITY DISCUSSION

In this section, we discuss the resistance of PAbAC against two adversaries, based on two realistic threat models, defined hereafter in Section 6.1. We prove the security of our proposed scheme with respect to the security requirements introduced in Section 2.

6.1 Threat Model

For designing the most suitable security solutions for cloud sharing scenarios, we consider two adversaries: malicious cloud user and *honest but curious* cloud server.
Honest but Curious Cloud Server Adversary – this storage server honestly performs the operations defined by our proposed scheme, but it may actively attempt to gain knowledge of the outsourced sensitive data, such as access patterns.
Malicious User Adversary – this attacker can be an unauthorized user. As such, he targets to get access to the outsourced shared data. The objective of a malicious user is to convince the cloud server that he is a legitimate cloud user.

6.2 Confidentiality

In our proposed PAbAC scheme, data files are stored on an encrypted form in cloud servers relying on an attribute based encryption scheme, in order to ensure efficient access control. As such, the data confidentiality preservation is tightly related to security of the used attribute based encryption scheme.

Theorem 1. PAbAC guarantees data confidentiality of the outsourced data.

Proof. The proposed PAbAC framework is designed to ensure data confidentiality against both malicious users and curious cloud provider.
In PAbAC, the data owner is in charge of encrypting his data before outsourcing them to the cloud storage server. He is also responsible for defining an

access structure that points out who can access the outsourced data with respect to a set of attributes. Then, the cloud provider is responsible for sending data to requesting users after authenticating them, relying on the access policy defined by the data owner. As such, only the authorized users having the access structure's satisfying attributes can generate the deciphering keys.
In addition, while considering a curious cloud service provider who tries to gain knowledge about outsourced data file, this latter cannot access the outsourced data. As detailed in Section 5, our encryption algorithm relies on the Lewko and Waters proposal. That is, PAbAC inherits the security properties from (Lewko and Waters, 2011). In addition, data confidentiality preservation against malicious users and a curious cloud provider is ensured based on the security of our proposed access control scheme detailed in the Section 6.4 and the security level of the applied encryption scheme (c.f. Lemma 1). □

Lemma 1. Unauthorized users cannot decrypt the encrypted data.

Proof. The proof of this lemma is equivalent to the security of the data decryption algorithm. The correctness of our decryption algorithm is as follows:
A user can decrypt data if and only if it has a matching set of attributes. In fact, access structure ψ (and hence matrix A) is constructed if and only if there exists a set of rows A_i in A, and linear constants $c_i \in \mathbb{Z}_N$ such that $\sum_i c_i A_i = [1,0,\cdots,0]$.
We note that

$$\frac{C_{1,i}\hat{e}(L, C_{3,i})}{\hat{e}(LK_i, C_{2,i})} \tag{15}$$

$$= \frac{\hat{e}(g_1,g_1)^{\lambda_i}\hat{e}(g_1{}^t, g_1{}^{p_i})\hat{e}(g_1{}^t, g_1{}^{w_i})\hat{e}(g_1{}^t, u_i{}^{p_i})}{\hat{e}(u_i{}^t, g_1{}^{p_i})\hat{e}(g_1{}^t, g_1{}^{p_i})} \tag{16}$$

Thus

$$\prod_i (\hat{e}(g_1,g_1)^{\lambda_i}\hat{e}(g_1,g_1)^{tw_i})^{c_i} = \hat{e}(g_1,g_1)^s \tag{17}$$

We note that Equation (17) holds because $\lambda_i = \vec{A_i}\cdot\vec{v}$, $w_i = \vec{A_i}\cdot\vec{\tau}$, where $\vec{v}\cdot[1,0,\cdots,0] = s$ and $\vec{\tau}\cdot[1,0,\cdots,0] = 0$. In the sequel, an authorized user can obtain the data D_F as follows:

$$D_F = C_0/\hat{e}(g_1,g_1)^s \tag{18}$$

For an unauthorized user who does not possess the secret keys related to the set of attributes required for satisfying the access policy, it is impossible to compute $\sum_i c_i\vec{A_i} = [1,0,\cdots,0]$ (Lewko and Waters, 2011). Thus, $\hat{e}(g_1,g_1)^s$ cannot be calculated and the adversary cannot recover the data file D_F. □

6.3 Privacy

Based on an attribute based signature scheme, PAbAC ensures users' privacy against curious cloud provider. 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. But, beyond the ABS properties, our PAbAC scheme ensures the protection of the users identities' (non traceability property). In fact, the ABE scheme used does not reveal the encryptor identity neither the users' attributes used in the backup phase. The PAbAC inherits the non traceability property from the Lewko encryption scheme (Lewko and Waters, 2011).

Theorem 2. PAbAC signature scheme is a privacy preserving signature.

Proof. The PAbAC signature scheme does not reveal neither the identity of the signer nor the set of attributes used in the signing. Our signature scheme requires that the identity of the signer remains anonymous. Thus, PAbAC ensures that a signature does not reveal more information other than what can be already inferred from the signing predicate itself. The demonstration of this state is derived from the following lemmas. □

Lemma 2. PAbAC protects user's anonymity

Proof. In the authentication procedure, the user has to sign a random message received from the CSP. Based on the signature scheme introduced in PAbAC, the user signs the message using his private keys which have already been randomized. Thus, the generated signature does not reveal the attributes used neither the user's private keys. Based on the hardness of the Computational Diffie Hellman problem (CDH), the CSP can not deduce the user's private keys related to the used attributes from the signature $\sigma = (x_1,\cdots,x_n,\sigma_1,\sigma_2)$ received.
In addition, let us consider a curious cloud provider adversary that chooses a message m, a signing policy and two requesting users with two, possibly different, sets of attributes with the condition that both sets have to satisfy the signing policy. The adversary gets a signature by either signer and wins if it correctly guesses the signer. The curious provider has a negligible advantage to win the previous game. That is, the PAbAC signature scheme is based on the randomization of the signer secret keys. □

Lemma 3. PAbAC's signature scheme is unlinkable

Proof. Let us consider that a curious cloud provider aims to deduce identifying information about a requesting user by running different authentication sessions. In the PAbAC `sign` algorithm, the user randomises the attributes' secret keys received from the attributes authorities as follows:

$$K' = Kh^{t'} \tag{19}$$

$$L' = L{g_1}^{t'} \tag{20}$$

$$K'_i = K_i {u_i}^{t'} \forall i \in S_U \tag{21}$$

Then, in every authentication session, the user generates a new signature $\sigma = (x_1, \cdots, x_n, \sigma_1, \sigma_2)$ thanks to the selected random value t'. Moreover, the CSP 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). As such, while authenticating the same user based on different authentication sessions, a curious cloud service provider cannot identify the requesting user. □

6.4 Access Control to Data

PAbAC introduces two-level access control model that combines the authentication of the requesting users and the attribute based decryption algorithm. In the following, we demonstrate that our PAbAC access control enforcement is resistant against both malicious data users and a curious cloud service provider.

Theorem 3. 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 PAbAC, only users, having valid private keys related to their attributes, are able to access data stored in the cloud while successfully authentication with the cloud server. This is due to the correctness of our encryption and signature algorithms and the compliance of the unforgeability property of the PAbAC signature scheme inherited from (Waters, 2005).

Lemma 4. Data Decryption Correctness.

Proof. After receiving his attributes' secret keys $\{sk_{S_j}\}$, the authorized user first computes:

$$\frac{C_{1,i}\hat{e}(L, C_{3,i})}{\hat{e}(LK_i, C_{2,i})} = (\hat{e}(g_1,g_1)^{\lambda_i}\hat{e}(g_1,g_1)^{tw_i})^{c_i} \tag{22}$$

Then, he computes the constants $c_i \in \mathbb{Z}_N$ such that $\sum_i c_i \cdot \vec{A}_i = [1, 0, \cdots, 0]$. Then, $\hat{e}(g_1,g_1)^s$ could be obtained as follows:

$$\prod_i (\hat{e}(g_1,g_1)^{\lambda_i}\hat{e}(g_1,g_1)^{tw_i})^{c_i} \tag{23}$$

$$= \hat{e}(g_1,g_1)^{\sum_i \lambda_i c_i}\hat{e}(g_1,g_1)^{\sum_i tw_i c_i} = \hat{e}(g_1,g_1)^s \tag{24}$$

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 $\vec{\tau} = [0, \tau_2, \cdots, \tau_n]$. Consequently, we note that $\sum_i \lambda_i c_i = s$ and $\sum t w_i c_i = 0$.
Afterwards, the user can recover the data file D_F as follows:

$$D_F = C_0 \; / \hat{e}(g_1,g_1)^s$$

□

Lemma 5. Data signature correctness

Proof. When an authorized user wants to access outsourced data, he has to provide a correct signature, with respect to the access policy defined by the data owner, that can be verified by the CSP in an anonymous way. If $\sigma = (x_1, \cdots, x_l, \sigma_1, \sigma_2)$ is a valid signature of the message m for the predicate ψ, then

$$\sigma_1 = zK'g^m r \tag{25}$$

$$= {g_1}^{mr} h^{t+t'} {g_1}^{\alpha} \prod_{i=1}^{n} ({u_i}^{(t+t')})^{y_i} \tag{26}$$

Thus

$$\hat{e}(g_1, \sigma_1) = \hat{e}(g_1, {g_1}^{mr} h^{t+t'} {g_1}^{\alpha} \prod_{i=1}^{n} ({u_i}^{(t+t')})^{y_i}) \tag{27}$$

$$= pk_A \hat{e}(g_1, {g_1}^{r})^m \hat{e}(g_1, h^{t+t'}) \prod_{i=1}^{n} \hat{e}(g_1, {u_i}^{(t+t')})^{y_i}) \tag{28}$$

$$= pk_A \hat{e}(g_1, \sigma_2)^m \prod_{i=1}^{n} \hat{e}(x_i, h^{\mu_i} u_i) \tag{29}$$

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

$$\sum_{i=1}^{n} \mu_i y_i t = t \sum_{i=1}^{n} \mu_i y_i = t.1 = t \tag{30}$$

□

Theorem 4. Unauthorized entities are unable to access the outsourced data files

Lemma 6. PAbAC is secure against the collusion attack.

Proof. We recall that the unforgeability property ensures that even if requesting users collude and combine their attributes together, they cannot forge a signature that opens to a signer whose attributes do not satisfy the access policy. It also covers non-frameability and ensures that even if requesting users collude, they cannot frame a user who did not produce the signature. Similarly, malicious users cannot collude to decipher an encrypted data file if the attributes of each individual user do not satisfy the access policy, defined by the data owner.

Table 2: Performances Comparison for Different Access Control Mechanisms in Cloud Data Storage Environments.

Scheme	*Data Owner Comp.*	*CSP Comp.*	*User comp.*
(Zhao et al., 2011)	$E+(2n+1)E_1$	$\tau_P(3+2n)+(2nl+1)E_1$	$(2+3n+2nl)E_1+(2n+1)\tau_P+(n+1)E$
(Ruj et al., 2012)	$3nE_1+(2n+1)E+\tau_P$	$\tau_P(3+2n)+(2nl+1)E_1$	$(2+3n+2nl)E_1+2n\tau_P+nE$
(Ruj et al., 2011)	$3nE_1+(2n+1)E+\tau_P$	$--$	$2n\tau_P+nE$
(Ruj et al., 2014)	$3nE_1+(2n+1)E+\tau_P$	$\tau_P(3+2n)+(2nl+1)E_1$	$(2+3n+2nl)E_1+2n\tau_P+nE$
PAbAC	$(n+1)E+4nE_1+\tau_P$	$(2+n)\tau_P+(n+1)E_1$	$2(n+1)E_1+2n\tau_P+nE$

Let us consider two malicious users U_A and U_B having each a set of attributes X_A and X_B such as $X_A \cup X_B$ satisfy the access structure. Suppose that U_A gets $sk_{S_1} = (K_1, L_1, K_{i_1}) = ({g_1}^{\alpha} \cdot h^{t_1}, {g_1}^{t_1}, {u_{i_1}}^{t_1})$ and U_B gets $sk_{S_2} = (K_2, L_2, K_{i_2}) = ({g_1}^{\alpha} \cdot h^{t_2}, {g_1}^{t_2}, {u_{i_2}}^{t_2})$. The users collude to create a valid set of attributes and derive a secret key $sk_{S_{1\cup 2}} = sk_{S_1} \cup sk_{S_2}$ from the combination of the two user's keys. Then, the colluded malicious users try to decrypt the data as follows:

$$\frac{C_{1,i}\hat{e}(L, C_{3,i})}{\hat{e}(LK_i, C_{2,i})} = \tag{31}$$

$$\frac{\hat{e}(g_1, g_1)^{\lambda_i}\hat{e}({g_1}^{t_1+t_2}, {g_1}^{p_i})\hat{e}({g_1}^{t_1+t_2}, {g_1}^{w_i})\hat{e}({g_1}^{t_1+t_2}, {u_1}^{p_i})}{\hat{e}({u_{i_1}}^{t_1}{u_{i_2}}^{t_2}, {g_1}^{p_i})\hat{e}({g_1}^{t_1+t_2}, {g_1}^{p_i})} \tag{32}$$

Afterwards, the equation (12) cannot be resolved as detailed in (32). Thus, the malicious users can not recover the original data. □

Lemma 7. The CSP is unable to access the encrypted data files

Proof. The CSP cannot decipher encrypted data because it does not possess the secret keys $\{sk_{S_j}\}$, required for satisfying the access policy defined by the data owner. Even if the cloud provider colludes with other unauthorized users, it cannot decrypt data, since the PAbAC scheme is collusion resistant as detailed in Section 6.4. Moreover, we suppose that the attribute authorities AAs are not hosted by the CSP. Thus, even if some attribute authorities are compromised, the CSP cannot decipher data. □

Lemma 8. PAbAC is resistant to replay attacks.

Proof. In our proposed PAbAC signature scheme, the message m, sent by the CSP to the requesting user, is assumed to be different in each authentication session (i.e; m presents the cloud provider's identity concatenated with the current time). In fact, for two different authentication sessions α and β, the CSP produces two different messages m_α and m_β respectively. Consequently, a malicious user cannot generate a valid signature if he attempts a replay attack based on collected data from two different authentication sessions. □

7 PERFORMANCE ANALYSIS

In this section, we present the computation and storage complexities of the PAbAC protocol at both the client and cloud provider sides. For this purpose, we are interested by the computations performed at the data owner side in order to execute the STORE procedure. Moreover, 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 2 details the performance comparison with most closely related data sharing schemes in cloud environments.

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 $\hat{e}(g_1, g_1)$ 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 $4n$ exponentiations in $\mathbb{G}_1$ to calculate $C_{2,i}$ and $C_{3,i}$.
The BACKUP procedure is made up three algorithms `verif` executed by the CSP and `sign` and `decdata` runned by the data user (U). The user first signs a random message in order to authenticate with the cloud. To sign the message, the user performs $2(n+1)$ exponentiations in $\mathbb{G}_1$. Then, this latter executes $2n$ pairing to calculate $\hat{e}(L), C_{3,i})$ and $\hat{e}(K_i \cdot L, C_{2,i})$ 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 existent access control schemes (Ruj et al., 2012), (Ruj et al., 2011), (Ruj et al., 2014) are based on the Lewko's decentralized attribute based encryption scheme (Lewko and Waters, 2011). During the encryption phase, the data owner has to perform one pairing function $\hat{e}(g_1, g_1)$ 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 $3n$ in $\mathbb{G}_1$. In the data decryption phase, the data user performs n exponentiations in $\mathbb{G}_T$ and $2n$ pairing functions.
The zhao et al.'s proposal (Zhao et al., 2011) is based on the use of the CP-ABE scheme proposed by Bethencourt et al. (Bethencourt et al., 2007). To encrypt the data file, the data owner performs $(2n+1)$ exponentiations in $\mathbb{G}_1$ and one exponentiation in $\mathbb{G}_T$. The user while decrypting data performs $n+1$ exponentiations in $\mathbb{G}_T$ and $2n+1$ pairing functions. The proposals (Ruj et al., 2012),(Zhao et al., 2011), (Ruj et al., 2014) are based on the use of the attribute based signature scheme proposed by (Maji et al., 2011). 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 $3+2n$ pairing function and $2nl+1$ exponentiations in $\mathbb{G}_1$.

8 CONCLUSIONS

The growing need for secure cloud sharing services and the attractive properties of the Attribute based Cryptography lead us to combine them, thus, defining an innovative solution to the data outsourcing security and efficiency issues.
In this paper, we design a privacy preserving attribute based framework for fine grained access control, for dynamic groups in untrusted cloud storage environments. Our approach 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. Our theoretical performances analysis shows the efficiency of PAbAC in scalable data sharing, while considering the impact of the cryptographic operations at both the client and the cloud provider side.

ACKNOWLEDGEMENTS

This work is a part of the MOBIDOC project achieved under the PASRI program, funded by the European Union and administered by the ANPR.

REFERENCES

Health Insurance Portability and Accountability Act (HIPAA). https://www.hipaa.com/about/.

Beimel, A. (1996). *Secure schemes for secret sharing and key distribution*. PhD thesis, Technion-Israel Institute of technology, Faculty of computer science.

Benaloh, J., Chase, M., Horvitz, E., and Lauter, K. (2009). Patient controlled encryption: ensuring privacy of electronic medical records. In *The 2009 ACM workshop on Cloud computing security*, pages 103–114. ACM.

Bethencourt, J., Sahai, A., and Waters, B. (2007). Ciphertext-policy attribute-based encryption. In *IEEE Symposium on Security and Privacy, 2007.*, pages 321–334.

Bobba, R., Fatemieh, O., Khan, F., Gunter, C., Khurana, H., et al. (2006). Using attribute-based access control to enable attribute-based messaging. In *The 22nd Annual Computer Security Applications Conference*, pages 403–413. IEEE.

Chaum, D. and Van Heyst, E. (1991). Group signatures. In *Advances in CryptologyEUROCRYPT91*, pages 257–265. Springer.

Di Vimercati, S. D. C., Foresti, S., Jajodia, S., Paraboschi, S., Pelosi, G., and Samarati, P. (2010a). Encryption-based policy enforcement for cloud storage. In *Distributed Computing Systems Workshops (ICDCSW), 2010 IEEE 30th International Conference on*, pages 42–51. IEEE.

Di Vimercati, S. D. C., Foresti, S., Jajodia, S., Paraboschi, S., and Samarati, P. (2007). Over-encryption: management of access control evolution on outsourced data. In *Proceedings of the 33rd international conference on Very large data bases*, pages 123–134. VLDB endowment.

Di Vimercati, S. D. C., Foresti, S., Livraga, G., and Samarati, P. (2015). Selective and private access to outsourced data centers. In *Handbook on Data Centers*, pages 997–1027. Springer.

Di Vimercati, S. D. C. D., Foresti, S., Jajodia, S., Paraboschi, S., and Samarati, P. (2010b). Encryption policies for regulating access to outsourced data. *ACM Transactions on Database Systems (TODS)*, 35(2):12.

El Kaafarani, A., Chen, L., Ghadafi, E., and Davenport, J. (2014a). Attribute-based signatures with user-controlled linkability. In *Cryptology and Network Security*, pages 256–269. Springer.

El Kaafarani, A., Ghadafi, E., and Khader, D. (2014b). Decentralized traceable attribute-based signatures. In *Topics in Cryptology–CT-RSA 2014*, pages 327–348. Springer.

Frikken, K. B., Li, J., and Atallah, M. J. (2006). Trust negotiation with hidden credentials, hidden policies, and policy cycles. In *NDSS*. Citeseer.

Ghadafi, E. (2015). Stronger security notions for decentralized traceable attribute-based signatures and more efficient constructions. In *Topics in Cryptology—CT-RSA 2015*, pages 391–409. Springer.

Goyal, V., Pandey, O., Sahai, A., and Waters, B. (2006). Attribute-based encryption for fine-grained access control of encrypted data. In *The 13th ACM conference on Computer and communications security*, pages 89–98.

Horváth, M. (2015). Attribute-based encryption optimized for cloud computing. In *SOFSEM 2015: Theory and Practice of Computer Science*, pages 566–577. Springer.

Horwitz, J. and Lynn, B. (2002). Toward hierarchical identity-based encryption. In *Advances in CryptologyEUROCRYPT 2002*, pages 466–481. Springer.

Hur, J. and Noh, D. K. (2011). Attribute-based access control with efficient revocation in data outsourcing systems. *IEEE Transactions on Parallel and Distributed Systems*, 22(7):1214–1221.

Jahid, S., Mittal, P., and Borisov, N. (2011). Easier: Encryption-based access control in social networks with efficient revocation. In *The 6th ACM Symposium on Information, Computer and Communications Security*, pages 411–415. ACM.

Kaaniche, N., Boudguiga, A., and Laurent, M. (2013). Id based cryptography for cloud data storage. In *2013 IEEE Sixth International Conference on Cloud Computing*, pages 375–382. IEEE.

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

Karchmer, M. and Wigderson, A. (1993). On span programs. In *Structure in Complexity Theory Conference*, pages 102–111.

Lewko, A. and Waters, B. (2011). Decentralizing attribute-based encryption. In *Advances in Cryptology–EUROCRYPT 2011*, pages 568–588. Springer.

Maji, H. K., Prabhakaran, M., and Rosulek, M. (2011). Attribute-based signatures. In *Topics in Cryptology–CT-RSA 2011*, pages 376–392. Springer.

Okamoto, T. and Takashima, K. (2013). Decentralized attribute-based signatures. In *Public-Key Cryptography–PKC 2013*, pages 125–142. Springer.

Raykova, M., Zhao, H., and Bellovin, S. (2012). Privacy enhanced access control for outsourced data sharing. In *Financial Cryptography and Data Security*, volume 7397, pages 223–238.

Rivest, R. L., Shamir, A., and Tauman, Y. (2001). How to leak a secret. In *Advances in CryptologyASIACRYPT 2001*, pages 552–565. Springer.

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

Ruj, S., Stojmenovic, M., and Nayak, A. (2012). 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*, pages 556–563. IEEE.

Ruj, S., Stojmenovic, M., and Nayak, A. (2014). Decentralized access control with anonymous authentication of data stored in clouds. *IEEE Transactions on Parallel and Distributed Systems*, 25(2):384–394.

Sahai, A. and Waters, B. (2005). Fuzzy identity-based encryption. In *EUROCRYPT 2005*, pages 457–473. Springer.

Wan, Z., Liu, J. E., and Deng, R. H. (2012). Hasbe: a hierarchical attribute-based solution for flexible and scalable access control in cloud computing. *IEEE Transactions on Information Forensics and Security*, 7(2):743–754.

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

Wang, W., Li, Z., Owens, R., and Bhargava, B. (2009). Secure and efficient access to outsourced data. In *The 2009 ACM workshop on Cloud computing security*, pages 55–66. ACM.

Waters, B. (2005). Efficient identity-based encryption without random oracles. In *Advances in Cryptology–EUROCRYPT 2005*, pages 114–127. Springer.

Yu, S., Wang, C., Ren, K., and Lou, W. (2010a). Achieving secure, scalable, and fine-grained data access control in cloud computing. In *INFOCOM IEEE Proceedings 2010*, pages 1–9.

Yu, S., Wang, C., Ren, K., and Lou, W. (2010b). Attribute based data sharing with attribute revocation. In *The 5th ACM Symposium on Information, Computer and Communications Security*, pages 261–270.

Zhao, F., Nishide, T., and Sakurai, K. (2011). Realizing fine-grained and flexible access control to outsourced data with attribute-based cryptosystems. In *Information Security Practice and Experience*, pages 83–97. Springer.

Zunnurhain, K. (2012). Fapa: a model to prevent flooding attacks in clouds. In *The 50th Annual Southeast Regional Conference*, pages 395–396. ACM.