An Integrated Privacy Preserving Attribute Based Access Control Framework Supporting Secure Deduplication

Runhua Xu, James Joshi, *Senior Member, IEEE* and Prashant Krishnamurthy, *Member, IEEE*

Abstract—Recent advances in information technologies have facilitated applications to generate, collect or process large amounts of sensitive personal data. Emerging cloud storage services provide a better paradigm to support the needs of such applications. Such cloud based solutions introduce additional security and privacy challenges when dealing with outsourced data including that of supporting fine-grained access control over such data stored in the cloud. In this paper, we propose an *integrated, privacy-preserving user-centric* attribute based access control framework to ensure the security and privacy of users' data outsourced and stored by a cloud service provider (CSP). The core component of the proposed framework is a novel *privacy-preserving, revocable ciphertext policy attribute-based encryption* (PR-CP-ABE) scheme. To support advanced access control features like *write access* on encrypted data and *privacy-preserving access policy updates*, we propose extended Path-ORAM access protocol that can also prevent *privacy disclosure of access patterns*. We also propose an integrated *secure deduplication* approach to improve the storage efficiency of CSPs while protecting data privacy. Finally, we evaluate the proposed framework and compare it with other existing solutions with regards to the security and performance issues.

Index Terms—Cryptography-based Access Control; Data Security and Privacy; Attribute-based Encryption; Secure Deduplication

1 Introduction

Recent advances in information technologies have enabled applications to generate, collect, or process large amounts of privacy-sensitive data. Even though personalized applications (e.g., healthcare applications) have been proposed recently, their deployment and maintenance costs are significantly higher because of the increasingly challenging security, privacy, and management issues [2].

To cope with increased storage capacity requirements and complex data management issues, cloud based storage services have become a very promising alternative for individual users as well as organizations [3]. A cloud storage service helps to aggregate users' or organizations' distributed data from different applications [4]. They, however, introduce additional security and privacy challenges such as those related to data privacy, access control and secure storage. Although encrypting the privacy-sensitive data that is outsourced to the cloud storage can ensure data confidentiality, providing fine-grained access control on such data is still a significant challenge [5]. The mechanisms used for outsourcing data to the cloud storage may further introduce privacy issues [5], [6], [7]. For instance, an adversary may be able to analyze access patterns when a user accesses data stored in the cloud to infer privacy sensitive information about the user [6].

- *R. Xu, J. Joshi and P. Krishnamurthy are with School of Computing and Information, University of Pittsburgh, PA, USA, 15260. Emails: runhua.xu@pitt.edu, jjoshi@pitt.edu,prashant@sis.pitt.edu*
- *A preliminary version of this work appears in the proceedings of 2016 IEEE 9th International Conference on Cloud Computing [1]. This is the extended version.*

Several cryptography-based access control schemes have been proposed recently to tackle the challenges related to ensuring data confidentiality by using encryption and providing fine-grained access control over encrypted outsourced data. These solutions aim to ensure that users can access their encrypted outsourced data at various levels of granularity. Ciphertext Policy Attribute based Encryption (CP-ABE) [8] provides one promising approach for fine-grained access control on such data stored in the cloud storage [9]. However, several challenges need to be tackled before CP-ABE schemes can be used in these applications. For example, original CP-ABE schemes do not support features like *privilege revocation*, and *write access* on encrypted data and *policy updates*. In a CP-ABE scheme, an access policy that is attached to the ciphertext may include several pieces of privacy sensitive information such as *social security number*, *affiliation*, *age* and *zip code* that need to be protected against an unauthorized disclosure. Besides, it is crucial that fully forward and backward secrecy is assured [10].

For cloud based applications, one key assumption made by existing approaches is that cloud storage providers (CSPs) are *honest-but-curious*, which means the CSPs will try to gather users' information related to the outsourced data while providing the services honestly. Even though sensitive personal data can be protected by encryption, it is possible for an adversary to analyze access patterns and infer some sensitive information [6]. To address the issue of privacy disclosure through access patterns, various mechanisms such as Oblivious Random Access Memory (ORAM) have been proposed in the literature [7], [11], [12], [13], [14]. However, these approaches have not been integrated with cryptography-based access control approaches.

In addition, the increasing volume of encrypted data

is also a huge challenge for cloud service providers. In particular, CSPs need to minimize the cost of storing outsourced data while protecting it. To eliminate redundant data in the storage in order to reduce storage cost, CSPs typically employ some data deduplication techniques [15], [16]. As conventional encryption schemes prevent CSPs from identifying duplicates over encrypted data, there exists a tension between deduplication and confidentiality of data. Approaches such as Convergent Encryption (CE) [17], Message-locked Encryption (MLE) [18], [19], and Server-aided Encryption [20] provide solutions to support deduplication over encrypted data. However, to the best of our knowledge, existing secure deduplication solutions only utilize symmetric cryptography but not asymmetric cryptography systems; hence, they cannot be directly used in cryptography-based access control systems such as CP-ABE based system.

With the rapid adoption of cloud based data storage and services, it is increasingly becoming critical that appropriate security and privacy solutions are available for cloud computing environments. In particular, it is very critical to design an integrated mechanism that can address the cryptography-based access control and access pattern privacy requirements, and the secure deduplication issues together. Although there exist several approaches to address secure deduplication and access pattern related privacy issues, none has attempted to integrate them together, and with cryptography-based access control mechanisms. There still exist several challenges in integrating existing approaches that tackle the secure deduplication and access pattern privacy issues with a cryptography-based access control mechanism, as follows:

(**a**) Existing secure deduplication schemes only handle the encrypted data using symmetric encryption approaches such as MLE and server-aided encryption; hence, they cannot support the cryptography-based access control scenarios because of the lack of a mechanism to verify the *ownership* or *valid access authorization* over encrypted data generated by the access control approaches such as an ABE-based scheme, where the encrypted data can be updated by both the data owner and the authorized data users with a *write* permission. For instance, existing secure deduplication approaches cannot support the cryptography-based access control scenarios where an authorized user, who has the *write* access on the encrypted data but does not *own* it, wants to append content to it.

(**b**) Moreover, the issue of privacy disclosure through access patterns has not been considered in existing cryptography-based access control mechanisms, while the existing solutions (i.e., the ORAM-based approaches) tackling the access pattern leakage issue only focus on how to support a read/write access by continuously shuffling data location in the CSP's disk. Besides, there still exist some gaps as discussed later in the integration of *path-ORAM tree* used in the ORAM-based approach and *Merkle tree* used in existing secure deduplication mechanism.

Existing approaches, thus, provide partial and/or issue-specific solutions to aforementioned challenges, but do not address these in an integrated way. In this paper, we propose an integrated, privacy-preserving user (or organization) centric attribute based access control framework to support fine-grained access control that includes read/write access over encrypted data with revocation capability, privacy-preserving access policy updates, secure data deduplication, and the protection of the privacy of access patterns. Following are the key contributions of our proposed work:

(***i***). The core component of the proposed framework is a novel access control approach using privacy-preserving revocable ciphertext policy attribute-based encryption (PR-CP-ABE). It supports immediate revocation of attributes, and prevents leakage of privacy sensitive information that is possible through the access structure used. Further, *Linear Secret Sharing Scheme (LSSS)* matrix is used as the access structure. LSSS has been proven to be an expressive and efficient policy structure. To the best of our knowledge, it is the first work that integrates privacy-preserving *LSSS* access structure with immediate attribute revocation.

(***ii***). We propose an extended path Oblivious RAM access (ePath-ORAM-Access) protocol to prevent disclosure of access patterns and provide advanced access control features (e.g., *write access* and *policy updates*) that are not supported by existing ABE schemes. This allows a client to hide its access patterns from a curious server in cloud storage applications. In addition, *ePath-ORAM-Access* also supports updating both encrypted data and access policies, i.e., data read/write and policy operations issues that have not been adequately addressed in the existing literature.

(***iii***). We also propose a secure deduplication solution based on the proposed access control scheme to satisfy the storage requirements for CSPs. To support this, we propose two *proof of ownership* mechanisms, a *proof of write* mechanism and a scheme to achieve secure deduplication.

(***iv***). We present an evaluation of our proposed framework with regards to both security guarantees and performance, and compare these with other existing approaches. We also implement a prototype of the proposed framework to evaluate the efficiency of the proposed PR-CP-ABE approach and the performance of secure deduplication mechanism in terms of processing time and storage requirements.

The rest of this paper is organized as follows. We overview the proposed access control framework in Section 2. We review some preliminaries and introduce our PR-CP-ABE construction in Section 3. The proposed outsourced data model and the secure deduplication solution are introduced in Section 4 and Section 5, respectively. We present advanced access control features in Section 6. The security and privacy analysis of our framework is presented in Section 7. We also present the performance analysis in Section 8. In Section 9, we discuss related work. Finally, we conclude this paper in Section 10.

2 Overview of Access Control Framework

2.1 A Motivating Example

Here, we present Example 1 to motivate an application scenario from the healthcare domain, and then overview the proposed access control framework.

Example 1. *A healthcare application scenario. We assume a patient/user-centric health application that allows a patient/user to store and manage all his Electronic Health Records (EHRs) by storing them in a CSP. The CSP is assumed to be honest-but-curious. Using our proposed framework, a patient stores his EHRs*

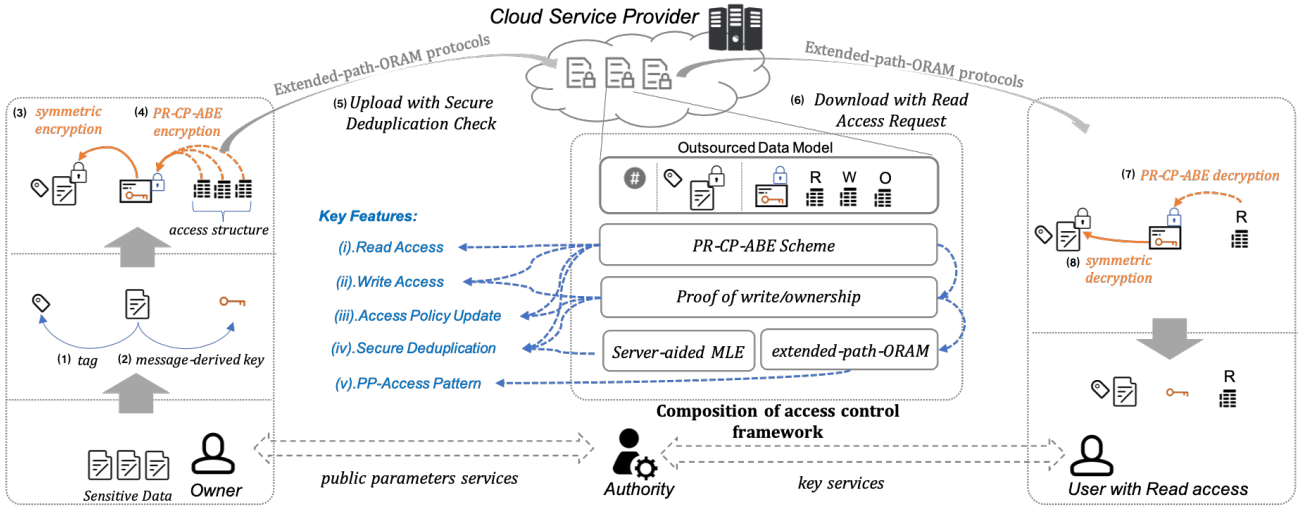


Fig. 1. An illustration of our proposed access control framework. Note that the figure only presents the procedure for the *read* access. The procedure for *write* and *owner* accesses are similar to *read* access except for the access structure in step (7).

in a cloud storage. Suppose that he lives in state Y and usually goes to hospital B. One day he travels to state X and needs to receive healthcare services in a different hospital A. He should be able to authorize read/write permissions to physician M in state X. When he comes back to state Y, he needs to revoke physician M's permissions immediately to prevent future accesses of his sensitive data by M. Moreover, he may need to provide/revoke the read permission to/from a pharmacist for buying medicine in a pharmacy at any place when he is traveling.

While the example is from the healthcare domain, it is easy to generalize our proposed approach to any user or organization-centric application that employs cloud storage services. Such a user-centric sensitive data management scenario can be supported by a cryptography-based access control scheme to provide authorized accesses to sensitive encrypted data based on fine-grained access control policies and privacy requirements.

2.2 Overview of the Framework

Our proposed access control framework has entities with the following roles: *data owner*, *data users*, *CSPs* and *Third Party Authority(TPA)*. The *data owner* specifies the access control policy and encrypts the data using our proposed approach, while the *data user* can access the shared encrypted data by decrypting it using the corresponding private key that is bound to the user's attribute identities that are generated by the *TPA*; the *TPA* sets up the public key and provides private key service for the entire system. Note that a user can decrypt the encrypted data successfully if and only if the user's attribute identities satisfy the associated access control policy.

In the real scenario, both *data owners* and *users* are customers of a cloud service. The CSPs could be any cloud storage service provider such as Amazon Drive, Google Drive, Microsoft OneDrive. Any independent entity that is widely trusted by other entities/roles in the framework can assume the role of a TPA, i.e., the providers of the certificates such as nonprofit organization Let's Encrypt and profit company Symantec.

As illustrated in Fig.1, we assume that a client (on behalf of a data owner) has several pieces of sensitive data to be outsourced to a CSP. First, it generates a tag using a collision-resistant hash function over the data, which is used for a duplication check (step 1). If there is no data duplication, the client encrypts the sensitive data using the sever-aided MLE (steps 2-3). In particular, the client generates a symmetric key with the help of the TPA using the message-derived key generation approach (step 2, see Section 3.1.3). Then the sensitive data is encrypted by using the message-derived key (step 3), while the key is protected by our proposed PR-CP-ABE scheme with three access control policies that are associated with the *read*, *write* and *ownership* permissions, respectively; these are specified by the data owner (step 4, see Section 3). Finally, the *outsourced data model* is used to upload the data after making a secure deduplication check (step 5). In our framework, the proposed *PoW/PoO* mechanisms are used to validate users' privileges to support the advanced access control features (i.e., *write* and *ownership* permissions) and the associated secure deduplication scheme. The *PoW/PoO* mechanisms are integrated in the proposed extended-Path-ORAM protocols to ensure access pattern privacy. For *read access* (step 6), an authorized user first decrypts the key related component to acquire the message-derived key using the PR-CP-ABE scheme (steps 7). Then the client decrypts the data related component using the message-derived key to access the original data (steps 8).

2.3 Adversary Model

The framework provides attribute based access control and protects against the following adversaries:

(i) *Any user* whose attributes do not satisfy the attribute based access policy encoded in the access structure for a requested access (e.g., read and/or write); this includes a user who may have had a valid authorization in the past but currently cannot be authorized because either the *policy has changed* or the user's *attributes have been revoked*; In addition, data users who have invalid authorization may also collude with each other to decrypt the data.

(ii) *Adversaries* who try to attack the cryptographic mechanisms including our proposed PR-CP-ABE scheme, as well as the server-aided MLE employed;

(iii) *Adversaries* who try to obtain privacy sensitive information about users; such an adversary may include an *honest-but-curious* CSP or any other entity that can (a) capture and analyze the access structures to gain information about users' attribute information, and/or (b) observe the physical storage locations accessed by users to learn about their access patterns.

We assume that the trusted third-party authority (TPA) that is responsible for providing the key service for PR-CP-ABE and sever-aided MLE does not collude with other entities. The client application that the users use to execute protocols is also trusted.

3 PR-CP-ABE Scheme

3.1 Preliminaries

3.1.1 Access Structures

We adopt the definitions of *access structure* and *linear secret sharing scheme* formalized in [21].

Definition 1. *Access Structures. Let $\mathcal{U}$ be the attribute universe. An access structure on $\mathcal{U}$ is a collection $\mathcal{AS}$ of non-empty sets of attributes, i.e., $\mathcal{AS} \subseteq 2^{\mathcal{U}} \backslash \emptyset$. The sets in $\mathcal{AS}$ are called the authorized sets and the sets not in $\mathcal{AS}$ are called the unauthorized sets. Additionally, an access structure is called monotone if $\forall B, C$: if $B \in \mathcal{AS}$ and $B \subseteq C$, then $C \in \mathcal{AS}$.*

The access structure encodes an access policy by specifying the set of attributes required to gain an access. Here, we only consider monotonic access structures. Monotonicity indicates that a user does not lose her authorized accesses if she acquires more attributes, as specified in [21].

Privacy-preserving Access Structure. An access structure $\mathcal{AS} = (M, \rho, \tau)$ describes an access policy. Here M is an $l \times n$ share-generating matrix associated with a secret sharing scheme, ρ is a function mapping each row of M to an attribute name and $\tau = \{t_{\rho(i)}\}_{1 \le i \le l}$ is a set of values of associated attributes. In our construction, τ is hidden and the other two parts are attached to the ciphertext; i.e., we use $\mathcal{AS} = (M, \rho)$.

Definition 2. *Linear Secret Sharing Schemes (LSSS). If a secret-sharing scheme Π, is linear, it should satisfy the following two conditions:*
(i). For each party, the generated share should be a vector.
(ii). There should be a share-generating matrix, $M_{l \times n}$, for the scheme. For each row in the matrix, we define function $\rho(\cdot)$ such that $\rho(i)$ maps to the i-th party. We generate n-1 random numbers (over $\mathbb{Z}_p$) and combine them with the secret s to get the column vector v. Then we define $\lambda = Mv^T$ to be the sharing vector for secret s such that share λ_i is for party $\rho(i)$.

Let Π be the corresponding LSSS for an access structure $\mathcal{AS}$. Let $S \in \mathcal{AS}$ be an authorized set, and $I = \{i : \rho(i) \in S\}$. Then we have constant set $\{\omega_i \in \mathbb{Z}_p\}_{i \in I}$ that satisfies $\sum_{i \in I} \omega_i \lambda_i^{'} = s$, where $\lambda_i^{'}$ are valid shares of the secret s generated by Π. According to [21], constant set $\{\omega_i\}$ can be generated in polynomial time in the size of M.

The proposed privacy-preserving access structure with τ hidden can ensure users' attribute privacy effectively, as illustrated in Example 2. In *"SSN: 123-45-6789," SSN* is the attribute name and *123-45-6789* is the attribute value. *SSN* is a privacy sensitive attribute.

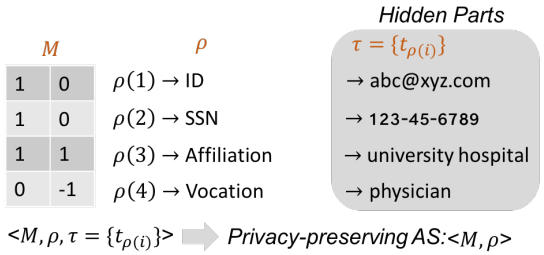


Fig. 2. An example of privacy-preserving access structure.

Example 2. *As shown in Fig.2, suppose that a patient's EHRs are encrypted with an access policy as follows:*

(ID: abc@xyz.com OR *SSN: 123-45-6789)* OR
(Affiliation: university hospital AND *Vocation: physician).*

It means that either the owner with the given ID or SSN can access the EHRs, or the physician in University Hospital can access the EHRs. After the encryption, the access policy that is attached to the outsourced EHRs will be as follows:

*(ID: ** OR *SSN: *)* OR *(Affiliation: ** AND *Vocation: *).*

Thus, even though an attacker may see the attribute name, he does not have the attribute value.

3.1.2 Bilinear Maps

Bilinear maps are used by most of the pairing-based crypto schemes such as identity-based encryption, attribute based encryption and its variants [22]. Bilinear groups of composite order are groups with an efficient bilinear map where the group order is a product of two large primes [23], [24]. Suppose we have cyclic groups $\mathbb{G}$ and $\mathbb{G}_T$ with order $N = pr$ (p, r are distinct primes) and a map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$. Then it should have the following properties [22]: *(i) Bilinear property:* $\forall m, n \in \mathbb{G}, x, y \in Z_N, e(m^x, n^y) = e(m, n)^{xy}$; *(ii) Non-degenerate property:* $\exists m \in \mathbb{G}$ such that $e(m, n)$ has order N in $\mathbb{G}_T$; *(iii) Symmetric property:* $e(g^x, g^y) = e(g, g)^{xy} = e(g^y, g^x)$; *(iv) Orthogonal property:* Let $\mathbb{G}_r$ and $\mathbb{G}_p$ be the subgroups of $\mathbb{G}$ with order r and p, respectively, such that $e(g_r, g_p) = 1$, where $g_p \in \mathbb{G}_p$ and $g_r \in \mathbb{G}_r$.

Our proposed PR-CP-ABE is constructed using composite order bilinear groups and hence has the benefits of the properties described above.

3.1.3 Message-derived Key Generation

We employ RSA based *oblivious pseudo-random function* (RSA-OPRF) [20], [25] to help generate a message-derived key. In the RSA-OPRF protocol, the server can not learn the client's inputs and resulting PRF outputs, while the client learns nothing about the key server's private key.

The RSA-OPRF protocol works as follows. Before the protocol starts, the server sets up the RSA parameters to generate a public/private key pair $(pk, sk) = (N, (N, d))$ with input e, where $ed \equiv 1 \mod \phi(N)$. In our setting, the authority will setup the RSA parameters. The client selects a random number $r \in \mathbb{Z}_N^*$ and two hash functions $H_1 : \{0,1\}^* \to \mathbb{Z}_N^*$ and $H_2 : \mathbb{Z}_N^* \to \{0,1\}^k$. Then the client calculates $h = H_1(M), x = hr^e \mod N$ and sends x to the

TABLE 1
Some notations and symbols in this paper.

Symbols	Description
PK	The public key used by data owners/users.
MSK	The master secret key used by the authority to generate a data user's private key.
$\mathcal{AS}:(M_{l\times n},\rho,\tau)$	The access structure includes a share-generating matrix $M_{l\times n}$, a conjunctive function ρ and related attribute value set τ .
$\mathcal{AS}:(M_{l\times n},\rho)$	The privacy-preserving access structure.
$\mathcal{AS}_r,\mathcal{AS}_w,\mathcal{AS}_o$	The access structure components associated with *read*, *wirte* and *owner* permissions, respectively.
$E_\psi(\cdot),D_\psi(\cdot)$	The encryption and decryption functions of the PR-CP-ABE scheme, respectively.
$E_k(\cdot),D_k(\cdot)$	The encryption and decryption functions of a symmetric encryption scheme using key k, respectively.
$f_{tag}(\cdot)$	The collision resistant hash function.

server. The server calculates $y = x^d \mod N$ and sends y to the client. The client computes $z = yr^{-1}$. If $z^e \mod N = h$, the client gets the message-derived key $k_m = H_2(z)$.

3.1.4 Secure Oblivious RAM

The goal of oblivious RAM is to hide a client's pattern of access to the stored data in the server.

Definition 3. *Secure Oblivious RAM [26]. Let access pattern $\mathcal{AP}(y)$ be the sequence of accesses to the remote storage system, where y is a sequence of data items and corresponding sequence of operations, namely, read and write operations. An oblivious RAM system is considered secure if for any two inputs $y, y^{'}$ of the client, of equal length, the access patterns $\mathcal{AP}(y)$ and $\mathcal{AP}(y^{'})$ are computationally indistinguishable for anyone but the client.*

Our *ePath-ORAM-Access* protocols extend the Path-ORAM proposed in [12]. Besides, as our extended protocols build on the structure of Path-ORAM, they provide the same security guarantee.

3.2 Proposed PR-CP-ABE Model

The proposed PR-CP-ABE scheme has five algorithmic components. We overview them here and present their detailed construction later. Table 1 lists the key notations used.

Setup. The authority runs the *setup* algorithm; it takes a security parameter 1^λ as input and produces a set of public parameters PK and the master key MSK as output.

Encrypt. The data owner runs the *encrypt* algorithm. It takes as inputs the public parameters in PK, a message m, and an access structure $\mathcal{AS} = (M, \rho, \tau)$ over the universe of attributes, and outputs the corresponding ciphertext.

KeyGen. The authority runs the *key generator* algorithm that takes as input the master key MSK and a set of attributes S; it outputs the secret key sk_1 for the user, and the delegation key sk_2 for the CSP.

Re-encrypt. The CSP runs the *re-encrypt* algorithm that takes as input the ciphertext and the delegation key, sk_2. It then re-encrypts the ciphertext and adds a new random element into the ciphertext component, which is associated with a set of revoked attributes. In our PR-CP-ABE scheme, users can only get the re-encrypted ciphertext from the CSP.

Decrypt. A user accessing the data runs the *decrypt* algorithm, which takes as input the re-encrypted ciphertext containing a partial access structure (M, ρ) and a secret key sk_1 for the user's attribute set S. If S satisfies the access structure, it will output m; else, it outputs a stop sign $\perp$.

Feature Comparison. Although *CP-ABE* schemes can support user-centric access control scenarios [27], and access policy privacy and revocation issues have been addressed separately in [28], [29], [30], [31], [32], integration of the two features is still a challenge and has not been addressed [33]. As shown in Table 2, although other existing schemes address some issues, to the best of our knowledge, our proposed framework is the first to integrate these features within one framework.

3.3 Construction Details

We now present details of the proposed PR-CP-ABE scheme:

Setup$(1^\lambda, U)$. The algorithm first generates initial parameters $(p_1, p_2, \mathbb{G}, \mathbb{G}_T, e)$, where p_1, p_2 are primes and $\mathbb{G}, \mathbb{G}_T$ are cyclic groups with order $N = p_1p_2$. Thus, $\mathbb{G}_{p_1}, \mathbb{G}_{p_2}$ are the subgroups of $\mathbb{G}$, $\mathbb{G} = \mathbb{G}_{p_1} \times \mathbb{G}_{p_2}$.

Next, the algorithm randomly chooses $g, h, \{u_i\}_{1\le i\le n}$ from $\mathbb{G}_{p_1}$ and α_1, α_2, a from $\mathbb{Z}_N^*$ such that $\alpha = (\alpha_1 + \alpha_2) \bmod N$, and $Z \in \mathbb{G}_{p_2}$. The public key generated is:

$$PK = (N, g, g^a, g^\alpha, e(g,g)^\alpha, \{u_i\}_{1\le i\le n}, H = h \cdot Z).$$

The master key is $MSK = (h, \alpha_1, \alpha_2)$.

Encrypt$(PK, m,$ $(M_{l\times n}, \rho, \tau))$. The encryption algorithm takes the public key PK, a message m and a LSSS access structure $(M_{l\times n}, \rho, \tau)$ as input.

The algorithm first generates two secrets $s_1, s_2 \in \mathbb{Z}_N^*$ randomly, and then calculates ciphertext components $\widetilde{C}_1 = m \cdot e(g,g)^{\alpha s_1}, \widetilde{C}_2 = e(g,g)^{\alpha s_2}$ and $C_j = \{g^{s_j}\}_{1\le j\le 2}$.

To help share the secrets among attributes (see *Definition 2*), it chooses two random column vectors $\vec{v_1}, \vec{v_2}$ from $\mathbb{Z}_N^*$, where $\vec{v_j}^T = (s_j, v_{j,2}, ..., v_{j,n})_{1\le j\le 2}$. Let $\{Z_{j,1,i}, Z_{j,2,i}\}_{1\le i\le l, 1\le j\le 2}$ and $\{r_{1,i}, r_{2,i}\}_{1\le i\le l}$ be chosen uniformly at random from $\mathbb{G}_{p_2}$ and $\mathbb{Z}_N^*$, respectively. The attribute-related components computed are:

$$C_{j,i} = \{g^{a\vec{M}_i\vec{v_j}^T} \cdot Z_{j,1,i}, (u_{\rho(i)}^{t_{\rho(i)}} H)^{r_{j,i}}\}_{1\le i\le l, 1\le j\le 2},$$
$$D_{j,i} = \{g^{r_{j,i}} \cdot Z_{j,2,i}\}_{1\le i\le l, 1\le j\le 2},$$

where $\vec{M}_i$ is the vector corresponding to the i-th row of M, and $u_{\rho(i)}$ is the corresponding components from public key. Finally, the output is the ciphertext CT as follows:

$$CT = (\{(M,\rho)\}, \{\widetilde{C}_j, C_j, \{C_{j,i}, D_{j,i}\}_{1\le i\le l}\}_{1\le j\le 2}.$$

KeyGen(PK, MSK, S). The KeyGen algorithm takes public key PK, master key MSK and a user's attribute set S=$\{s_i \to a_i : t_i\}_{1\le i\le n}$ as inputs and returns two secret keys: user's private key sk_1 and delegation key sk_2 for the CSP. Here, s_i is the attribute information that includes an attribute name a_i and an attribute value t_i. It randomly chooses $t \in \mathbb{Z}_N^*$ and $R, R', \{R_i\}_{1\le i\le n}$ from $\mathbb{G}_{p_2}$. Then users' secret keys are generated as $sk_1 = (k, k', \{k_i\}_{1\le i\le n})$, where $k = R \cdot g^{\alpha_1} \cdot g^{at}, k' = R' \cdot g^t, k_i = R_i \cdot \{(u_i^{t_i} h)^t\}_{1\le i\le n}$. The delegation key for CSP is generated as $sk_2 = (g^{\alpha_2})$.

TABLE 2
Comparison of key features

Schemes	$\mathcal{AS}^{\dagger}$Type	Immediate Revocation	Privacy Preserving $\mathcal{AS}^{\dagger}$	Privacy Preserving $\mathcal{AP}^{\ddagger}$
[10]	LSSS Matrix	●	○	○
[29]	And-gate	●	○	○
[30]	Tree-based	●	○	○
[31]	LSSS Matrix	○	●	○
[32]	And-gate	○	●	○
Ours	LSSS Matrix	●	●	●

† $\mathcal{AS}$ represents *access structure*. ‡ $\mathcal{AP}$ represents *access pattern*.

Re-encrypt(CT, sk_2). Re-encryption algorithm takes the initial ciphertext CT and the delegation key as input to re-encrypt and returns the new ciphertext $\widetilde{CT}$. There are two cases to consider:

Suppose that there are no revoked attributes. The CSP selects an element $\theta \in \mathbb{Z}_N^*$ randomly. Then the CSP calculates the ciphertext as follows:

$$\begin{aligned} D =&(sk_2)^{\theta} = g^{\alpha_2 \theta}, \\ C_j^{'} =&\{C_j^{(1/\theta)}\}_{1\le j\le 2}, \\ C_{j,i}^{'} =&\{g^{a\vec{M}_i \vec{v_j}^T} \cdot Z_{j,1,i} \cdot ((u_{\rho(i)}^{t_{\rho(i)}} H)^{r_{j,i}})^{\theta}\}_{1\le i\le l, 1\le j\le 2}, \\ D_{j,i}^{'} =&\{(D_{j,i})^{\theta}\}_{1\le i\le l, 1\le j\le 2}. \end{aligned}$$

Then the re-encrypted ciphertext is generated as

$$\widetilde{CT} = \{D, \{\widetilde{C}_j, C_j, C_j^{'}, \{C_{j,i}^{'}, D_{j,i}^{'}\}_{1\le i\le l}\}_{1\le j\le 2}\}.$$

Suppose that there is a revoked attribute $\mathbf{att}_x$. As in the previous case, it will select random elements $\theta, \theta_x \in \mathbb{Z}_N^*$ to encrypt the ciphertext and the delegation keys $D, C_j^{'}, C_{j,i}^{'}$ as before. The components $D_{j,i}^{'}$ are computed as follows:

$$D_{j,i}^{'} = \left\{ \begin{array}{ll} D_{j,i}^{\theta} & \text{if } \rho(i) \neq \text{att}_x \\ D_{j,i}^{\theta/\theta_x} & \text{if } \rho(i) = \text{att}_x \end{array} \right\}_{1\le j\le 2, 1\le i\le l}.$$

The re-encrypted ciphertext is generated as

$$\widetilde{CT} = \{D, \{\widetilde{C}_j, C_j, C_j^{'}, \{C_{j,i}^{'}, D_{j,i}^{'}\}_{1\le i\le l}\}_{1\le j\le 2}\}.$$

Decrypt$(\widetilde{CT}, sk_1)$. The decryption algorithm takes a ciphertext $\widetilde{CT}$ and a secret key sk_1 for a set of attributes S as input. It first calculates $I_{M,\rho}$ from (M, ρ), where $I_{M,\rho}$ denotes one of subsets of $\{1, ..., l\}$ that satisfies (M, ρ). Then it checks if there exists an $\mathcal{I} \in I_{M,\rho}$ that satisfies the following equation:

$$\frac{\widetilde{C}_2 \cdot \prod_{i\in\mathcal{I}} e(C_{2,i}^{'}, k^{'})^{\omega_i}}{e(C_2^{'}, D) \cdot e(C_2, K) \cdot \prod_{i\in\mathcal{I}} e(D_{2,i}^{'}, k_i)^{\omega_i}} = 1,$$

where $\sum_{i\in\mathcal{I}} \omega_i \vec{M}_i = (1, 0, ..., 0)$. If the above test is not passed, it outputs $\perp$. Otherwise, it computes:

$$T = \frac{\prod_{i\in\mathcal{I}} e(C_{1,i}^{'}, k^{'})^{\omega_i}}{\prod_{i\in\mathcal{I}} e(D_{1,i}^{'}, k_i)^{\omega_i}} = e(g, g)^{ats_1}.$$

Message m is recovered as follows:

$$m = \frac{\widetilde{C}_1 \cdot T}{e(C_1^{'}, D) \cdot e(C_1, k)}.$$

We refer the readers to our conference version [1] for the correctness proof of PR-CP-ABE decryption.

4 Outsourced Data Model

To support secure deduplication and advanced access control features described earlier, we propose an *outsourced data model* $\mathcal{C}$ illustrated in Fig.3.

Definition 4. *Let k_{data} be the message-derived symmetric key generated from data d. Let $E_{k_{data}}(d)$ be the ciphertext of d encrypted by a symmetric encryption $E(\cdot)$ using key k_{data}. Let $f_{tag}(\cdot)$ be a collision resistant hash function. Let $E_{\psi}(k_{data})$ be the ciphertext of the message-derived key k_{data} encrypted by our proposed PR-CP-ABE scheme represented as ψ. Then, we denote the outsourced data $\mathcal{C}$ as a 2-tuple $(\mathcal{C}_{meta}, \mathcal{C}_{data})$, where*

$$\begin{aligned} \mathcal{C}_{meta} =&(id, \mathcal{AS}_r, \mathcal{AS}_w, \mathcal{AS}_o, P_{\mathcal{C}_{data}}), \\ \mathcal{C}_{data} =&(f_{tag}(d), E_{k_{data}}(d)). \end{aligned}$$

Here, $\mathcal{AS}_r, \mathcal{AS}_w, \mathcal{AS}_o$ are defined as follows, respectively.

$$\begin{aligned} \mathcal{AS}_r =&(\langle M_r, \rho_r\rangle, E_{\psi}(k_{data})), \\ \mathcal{AS}_w =&(\langle M_w, \rho_w\rangle, E_{\psi}(r_w), H(r_w)), \\ \mathcal{AS}_o =&(\langle M_o, \rho_o\rangle, E_{\psi}(r_o), H(r_o)), \end{aligned}$$

where r_w and r_o are random nonces.

The outsourced data includes two parts: $\mathcal{C}_{data}$ contains encrypted data and its tag information; $\mathcal{C}_{meta}$ includes access policies associated with the encrypted data. In $\mathcal{C}_{meta}$, id is the unique global identifier to represent the outsourced data and $P_{\mathcal{C}_{data}}$ is the position of the encrypted data, $\mathcal{C}_{data}$. $\mathcal{AS}_r, \mathcal{AS}_w, \mathcal{AS}_o$ are access structure components that are associated with *read*, *write* and *owner* permissions, respectively. $\langle M_r, \rho_r\rangle, \langle M_w, \rho_w\rangle, \langle M_o, \rho_o\rangle$ are privacy-preserving access structures associated with each permission type, *read*, *write*, and *owner*, respectively. For example, if an attribute set S_u of user u satisfies the privacy-preserving access structure $\langle M_r, \rho_r\rangle$, u is able to read the encrypted data. If u needs to write to the encrypted data, u should prove *write* permission to the CSP by running a *proof of write* access protocol that is based on the component $\mathcal{AS}_w$. Similarly, the component $\mathcal{AS}_o$ is the basis for the *proof of ownership* protocol. Here $H(r_w), H(r_o)$ are the hash values of the random seeds related to users' *write* and *ownership* permissions, respectively. Note that a user who has a ownership permission is able to update the access policies in $\mathcal{C}_{meta}$.

To store outsourced data, three storage components, as shown in Fig.3, are needed as per Definition 4. *Meta-data storage* is used to store component $\mathcal{C}_{meta}$. Component $\mathcal{C}_{data}$ is divided into two parts: tag information is stored in *rapid storage* where the data can be checked quickly, while the *encrypted data* is constructed in Merkle-based path ORAM tree blocks in normal data storage.

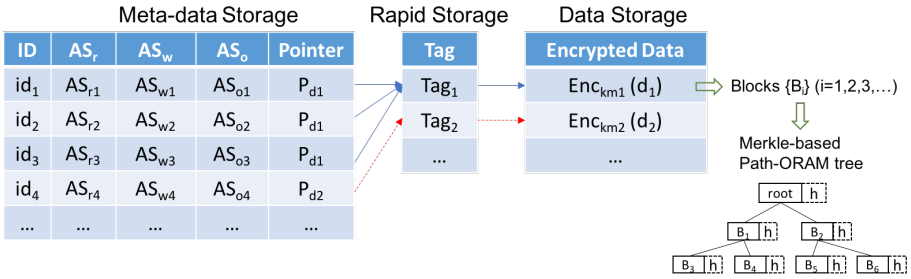


Fig. 3. Representation of outsourced data model supporting secure deduplication

5 Proposed Secure Deduplication Scheme

Although several secure deduplication schemes such as MLE and its variants [18], [19], [20], [34] have been proposed in the literature to address the deduplication challenge for the encrypted data, they do not support cryptography-based access control mechanisms (i.e., ABE related schemes).

The key issues are: (i) how can a data owner prove his *ownership*, and (ii) how can an authorized user prove his *read* and *write* rights over the encrypted data without leaking any part of the original data. To address these issues, we propose a new secure deduplication mechanism that includes following methods: *proof of ownership*, *proof of write*, and *message-derived key generation* (see Section 3.1.3). We note that MLE and its variants can support deduplication on encrypted data independently. Here we emphasize the need for *PoW/PoO*, as compared to those in existing secure deduplication mechanisms, as follows:

(i) Even though the duplicate of encrypted data $\mathsf{Enc}(d)$ can be found by checking the hash value of $\mathsf{Enc}(d)$, the *ownership* permission cannot be granted based on such a hash value, because of known attacks. Even if the adversary cannot break the confidentiality of $\mathsf{Enc}(d)$ immediately, the adversary can claim the ownership of such encrypted data and download it locally; it will increase the possibility of leakage in future through the use of additional approaches such as brute-force or side-channel attacks [35].

(ii) Consider the access scenario in Example 1; it is not hard for an adversary to claim the *ownership/write* privilege without *PoO/PoW* based on the existing schemes. Even if the adversary cannot break the encrypted data, it is possible to update the access structure, which will also result in data leakage [35].

5.1 Proof of Write/Ownership Permissions

To support *write* operation on the encrypted data and the secure deduplication operations, it is necessary to prove that a user is authorized for *write* access on the encrypted data. We focus on *write* access because the deduplication operation is associated only with the data upload phase [34]. The generalization of the *Proof of Write/Ownership* protocol is presented in Fig.4. We assume that a random one-time session identifier s_{id} is established before the proof protocol is executed. The integrity of the message is guaranteed by a signature algorithm $\mathsf{Sig}_{\mathsf{sk}}^{\mathsf{msg},s_{id}}$, where the subscript sk denotes the private key of the signer, and superscript msg, s_{id} represents the content to be signed. The key phases of the protocol include three steps: (i) **Challenge**, where the verifier prepares the challenge "question" for the prover;

Protocol ***Proof of Write/Ownership*** $^{\mathcal{CSP},\mathcal{CL}}$

$\mathcal{CL}_{(\mathsf{pk},\mathsf{sk})_{\mathcal{CL}}}$ $\qquad$ $\mathcal{CSP}_{(\mathsf{pk},\mathsf{sk})_{\mathcal{CSP}}}$

$\leftarrow \langle$establish session id, $s_{id}\rangle \rightarrow$

1 : $\mathsf{msg}_{\mathsf{req}} \leftarrow f_{id}$

2 : $\mathsf{msg}_{\mathsf{req}}, \mathsf{Sig}_{\mathsf{sk}_{\mathcal{CL}}}^{\mathsf{msg}_{\mathsf{req}},s_{id}} \longrightarrow$

3 : $\mathsf{msg}_{\mathsf{challenge}} \leftarrow \mathbf{Challenge}(\mathsf{msg}_{\mathsf{req}}, s_{id}, \mathsf{Sig}_{\mathsf{sk}_{\mathcal{CL}}}^{\mathsf{msg}_{\mathsf{req}},s_{id}})$

4 : $\mathsf{msg}_{\mathsf{challenge}}, \mathsf{Sig}_{\mathsf{sk}_{\mathcal{CSP}}}^{\mathsf{msg}_{\mathsf{challenge}},s_{id}} \longleftarrow$

5 : $\mathsf{msg}_{\mathsf{proof}} \leftarrow \mathbf{GenProof}(\mathsf{msg}_{\mathsf{challenge}}, s_{id}, \mathsf{Sig}_{\mathsf{sk}_{\mathcal{CL}}}^{\mathsf{msg}_{\mathsf{challenge}},s_{id}})$

6 : $\mathsf{msg}_{\mathsf{proof}}, \mathsf{Sig}_{\mathsf{sk}_{\mathcal{CL}}}^{\mathsf{msg}_{\mathsf{proof}},s_{id}} \longrightarrow$

7 : $\mathbf{VerifyProof}(\mathsf{msg}_{\mathsf{proof}}, s_{id}, \mathsf{Sig}_{\mathsf{sk}_{\mathcal{CL}}}^{\mathsf{msg}_{\mathsf{proof}},s_{id}})$

Fig. 4. Generalization of *Proof of Write/Ownership* $^{\mathcal{CSP},\mathcal{CL}}$ protocols. Note that the subscript of $\mathcal{CSP}$ and $\mathcal{CL}$ indicates its public and private key pairs, where the $\mathcal{CSP}$ and $\mathcal{CL}$ represent the cloud service provider and the user's client, respectively.

(ii) **GenProof**, where the prover generates the evidence to "answer" the challenge "question"; (iii) **VerifyProof**, where the verifier verifies the "answer" provided by the prover. The specification of each operation is presented below.

5.1.1 Proof of Write (PoW)

The *PoW* protocol is used by a client to prove to a server without leaking any information of a file that it indeed has the *write* permission for that encrypted file.

The key idea in the *PoW* protocol is to verify the decryption ability based on the PR-CP-ABE scheme. To prove that a user has the *write* permission on the encrypted data, the user's client first sends the identifier f_{id} of the data with generalized signature, including the random one-time session identifier, to the CSP, as shown in Fig.4. The specification of each operation for $PoW^{\mathcal{CSP},\mathcal{CL}}$ protocol is presented in Fig.5. Based on f_{id}, the CSP verifies the message and queries the meta-data storage to find the corresponding $\mathcal{C}_{meta}$. Then, the CSP extracts the access structure component $\mathcal{AS}_w$ and sends the challenge (i.e., the component $\langle M_w, \rho_w\rangle$ and $E_\psi(r_w)$) to the client. The client also verifies the message's signature first and generates the proof for its *write* permission by showing the decryption ability on the write permission related component $\mathcal{AS}_w$. Then, the client sends the hash of the decrypted random seed and random nonce session id with signature to the CSP. Finally, the CSP verifies the signature and compares the two hash values of the random seeds: the originally stored $\mathcal{AS}_w.\mathsf{H}(r_w)$ with random nonce session id, and the received $\mathsf{msg}_{\mathsf{proof}}.\mathsf{H}(r_w')$. If they match, it confirms that the user has the *write* permission

PoW-Challenge$(\text{msg}_{\text{req}}, s_{id}, \text{Sig}_{\text{sk}_{\mathcal{CL}}}^{\text{msg}_{\text{req}}, s_{id}})$

1 : **if** $\text{Vf}_{\text{pk}_{\mathcal{CL}}}(\text{msg}_{\text{req}}, s_{id}, \text{Sig}_{\text{sk}_{\mathcal{CL}}}^{\text{msg}_{\text{req}}, s_{id}})$ **is true then**
2 : $\quad \mathcal{AS}_w \leftarrow S_{\mathcal{C}_{meta}}[\text{msg}_{\text{req}}.f_{id}]$
3 : $\quad \text{msg}_{\text{challenge}} \leftarrow \mathcal{AS}_w.\langle M_w, \rho_w\rangle, \mathcal{AS}_w.E_\psi(r_w)$
4 : **else** $\text{msg}_{\text{challenge}} \leftarrow$ failure **and** abort
5 : **return** $\text{msg}_{\text{challenge}}$

PoW-GenProof$(\text{msg}_{\text{challenge}}, s_{id}, \text{Sig}_{\text{sk}_{\mathcal{CSP}}}^{\text{msg}_{\text{challenge}}, s_{id}})$

1 : $b \leftarrow \text{Vf}_{\text{pk}_{\mathcal{CSP}}}(\text{msg}_{\text{challenge}}, s_{id}, \text{Sig}_{\text{sk}_{\mathcal{CSP}}}^{\text{msg}_{\text{challenge}}, s_{id}})$
2 : **if** $b \doteq$ **true and** $\text{msg}_{\text{challenge}}$ **is not** failure **then**
3 : $\quad r_w^{'} \leftarrow D_{\psi, \text{msg}_{\text{challenge}}.\langle M_w, \rho_w\rangle}(\text{msg}_{\text{challenge}}.E_\psi(r_w))$
4 : $\quad \text{msg}_{\text{proof}} \leftarrow \text{Enc}_{\text{pk}_{\mathcal{CSP}}}(\text{H}(\text{H}(r_w^{'})|s_{id}))$
5 : **else** abort
6 : **return** $\text{msg}_{\text{proof}}$

PoW-VerifyProof$(\text{msg}_{\text{proof}}, s_{id}, \text{Sig}_{\text{sk}_{\mathcal{CL}}}^{\text{msg}_{\text{proof}}, s_{id}})$

1 : $b \leftarrow \text{Vf}_{\text{pk}_{\mathcal{CL}}}(\text{msg}_{\text{proof}}, s_{id}, \text{Sig}_{\text{sk}_{\mathcal{CL}}}^{\text{msg}_{\text{proof}}, s_{id}})$
2 : $\sigma \leftarrow \text{Dec}_{\text{sk}_{\mathcal{CSP}}}(\text{msg}_{\text{proof}})$
3 : **if** $b \doteq$ **true and** $\text{H}(\mathcal{AS}_w.\text{H}(r_w)|s_{id}) \doteq \sigma$ **then** accept
4 : **else** reject

Fig. 5. The *Proof of Write* $^{\mathcal{CSP},\mathcal{CL}}$ specification. Note that Vf_{pk} represents the corresponding verification algorithm of the signature algorithm Sig, where the subscript pk denotes the corresponding public key.

on the encrypted data even though the user does not have the original encrypted data.

5.1.2 Proof of Ownership (PoO)

The *PoO* protocol is used by a client to prove to a CSP that it owns the encrypted file without leaking any information about that file. We propose two ways to implement the *PoO* protocol: one is an enhancement of the traditional method, which we refer to as path-ORAM-Merkle-tree based PoO (*pom-PoO*); the other is a *PR-CP-ABE* based method that we refer to as decryption-based PoO (*d-PoO*).

Decryption-based proof of ownership. The *d-PoO* protocol is similar to the *PoW* protocol described in Section 5.1.1. The difference is that in *d-PoO*, the CSP verifies the client's ownership by checking its decryption ability on the ownership component $\mathcal{AS}_o$. In access structure $\langle M_o, \rho_o\rangle$, only data owners' attribute identities can satisfy the access policy. This protocol is similar to *PoW* $^{\mathcal{CSP},\mathcal{CL}}$ presented in Fig.5, so we do not present the details here.

Merkle-tree based proof of ownership. Since our framework employs the path-ORAM to prevent the leakage of users' access pattern, we propose a *PoO* protocol that is built on the path ORAM tree by integrating it with a Merkle-tree. A hash tag H is attached to each bucket (node) in the path ORAM tree, which is defined as follows:

$$H = Hash(B||H_{\text{left-child}}||H_{\text{right-child}}),$$

where B contains all the blocks in the bucket, and $H_{\text{left-child}}, H_{\text{right-child}}$ are the hash values of the left and right children.

Here, let $T_{h,z}$ be a binary Merkle based path ORAM tree with z blocks in each bucket and let h be a collision-resistant hash function. In the tree $T_{h,z}$, let the tuple $(v_{b_i,n}, v_{h,n})$ be the value of the node, where $v_{b_i,n}$ is the value of the block in the bucket and $v_{h,n} = h(v_{b_i,n}||v_{h,right}||v_{h,left})$ is the hash

pom-PoO-Challenge$(\text{msg}_{\text{req}}, s_{id}, \text{Sig}_{\text{sk}_{\mathcal{CL}}}^{\text{msg}_{\text{req}}, s_{id}})$

1 : **if** $\text{Vf}_{\text{pk}_{\mathcal{CL}}}(\text{msg}_{\text{req}}, s_{id}, \text{Sig}_{\text{sk}_{\mathcal{CL}}}^{\text{msg}_{\text{req}}, s_{id}})$ **is true then**
2 : $\quad \{i_0, i_1, ..., i_u\} \leftarrow_{\$} \{0, 1, 2, ..., N_{\text{leaf},\text{msg}_{\text{req}}.f_{id}}\}$
3 : $\quad \text{msg}_{\text{challenge}} \leftarrow \{i_j\}_{1\le j\le u}$
4 : **else** $\text{msg}_{\text{challenge}} \leftarrow$ failure **and** abort
5 : **return** $\text{msg}_{\text{challenge}}$

pom-PoO-GenProof$(\text{msg}_{\text{challenge}}, s_{id}, \text{Sig}_{\text{sk}_{\mathcal{CSP}}}^{\text{msg}_{\text{challenge}}, s_{id}})$

1 : $b \leftarrow \text{Vf}_{\text{pk}_{\mathcal{CSP}}}(\text{msg}_{\text{challenge}}, s_{id}, \text{Sig}_{\text{sk}_{\mathcal{CSP}}}^{\text{msg}_{\text{challenge}}, s_{id}})$
2 : **if** $b =$ **true and** $\text{msg}_{\text{challenge}}$ **is not** failure **then**
3 : $\quad S_P \leftarrow \{\emptyset\}$
4 : $\quad$ **foreach** index i **in**$\{i_j\}_{1\le j\le u}$ **do**
5 : $\quad\quad S_P \leftarrow$ add the sibling path P_i of leaf i from $T_{h,z,f_{id}}$
6 : $\quad$ **endforeach**
7 : $\quad \text{msg}_{\text{proof}} \leftarrow S_P$
8 : **else** abort
9 : **return** $\text{msg}_{\text{proof}}$

pom-PoO-VerifyProof$(\text{msg}_{\text{proof}}, s_{id}, \text{Sig}_{\text{sk}_{\mathcal{CL}}}^{\text{msg}_{\text{proof}}, s_{id}})$

1 : **if** $\text{Vf}_{\text{pk}_{\mathcal{CL}}}(\text{msg}_{\text{proof}}, s_{id}, \text{Sig}_{\text{sk}_{\mathcal{CL}}}^{\text{msg}_{\text{proof}}, s_{id}})$ is **true then**
2 : $\quad b_{\text{verify}} \leftarrow$ **false**
3 : $\quad$ **foreach** P_i **in** S_P **do**
4 : $\quad\quad v_{c,root} \leftarrow$ calculate the root value on sibling path P_i
5 : $\quad\quad$ **if** $v_{c,root} \neq v_{o,root}$ **then**
6 : $\quad\quad\quad b_{\text{verify}} \leftarrow$ **true**
7 : $\quad\quad\quad$ **break**
8 : $\quad$ **endforeach**
9 : $\quad$ **if** b_{verify} **then** reject **else** accept
10 : **else** reject

Fig. 6. The *Proof of Ownership* $^{\mathcal{CSP},\mathcal{CL}}$ protocol specification.

value of the node. If the node is a leaf node, the hash value components are set to empty strings. For a node x of $T_{h,z}$, its sibling path includes x and all the sibling nodes on the path from the root to x.

The *pom-PoO* protocol is presented in Fig.6 that also inherits from the protocol template in Fig.4. Before it starts, the CSP already has $T_{h,z}$ for each file, and keeps the following information of $T_{h,z}$ for verification: the number of leaves N_{leaf} and the value of the root $v_{o,root}$. Then the CSP randomly selects u leaf indices $\{i_0, i_1, ..., i_u\}$ and sends them to the client. For each leaf index, the client finds the sibling path and sends all sibling paths back to the verifier. The CSP will accept it if and only if all sibling paths are valid on $T_{h,z}$. Note that a claimed sibling path P_i is valid on $T_{h,z}$ if and only if the value of the root is equal to the computed value based on path P_i.

5.2 Proposed Secure Deduplication Mechanism

In this section, we present our proposed secure deduplication solution.

Setup.The setup phase initializes the necessary parameters, including those for the PR-CP-ABE scheme and the message-derived key generation method. The authority generates the public/private key pair as follows: $\text{pk} = (\text{pk}_{\text{PR-CP-ABE}}, \text{pk}_{\text{RSA-OPRF}})$, $\text{sk} = (\text{msk}_{\text{PR-CP-ABE}}, \text{sk}_{\text{RSA-OPRF}})$, and delivers the public keys to the corresponding parties. Also, the system shares a tag generation function: $f_{tag}(d)$, a collision-resistant hash function for the outsourced data d.

File Upload. Here, we only consider file deduplication. It is similar to the case of block deduplication. Suppose we have data d to be outsourced. The client first generates the tag $T = f_{tag}(d)$ and sends it to the CSP. After receiving T, the CSP checks if there exists the same tag in the rapid storage system. If it exists, the CSP sends a "*duplicate*" message indicating that the file already exists; otherwise, it responds with a "*non-duplicate*" message.

Case of Non-Duplicate Message. The client first generates the message-derived key k_d with the input data d by running the *message-derived key generation protocol* (see Section 3.1.3) with the *key server*, namely, the *the authority* in our setting. Then the client specifies the read/write/owner access policies and uses the PR-CP-ABE scheme to encrypt the message-derived key k_d and random key seeds r_w, and r_o. Finally, the client prepares $\mathcal{C}_{meta}$ and $\mathcal{C}_{data}$ and sends them to the CSP. Note that the position parameter $P_{\mathcal{C}_{data}}$ is generated by the CSP.

Case of Duplicate Message. If the user is the owner of the data, he runs the *PoO* protocol on the encrypted data $E_{k_d}(d)$ to prove his *ownership* right on it; if the user is authorized for an update, the client runs the *PoW* protocol. If *PoO/PoW* is accepted, the CSP only requires the client to send $\mathcal{C}_{meta}$ without the position component. Then the CSP will add the position $P_{\mathcal{C}_{data}}$ and store it into the database.

File Download. To download a file, the client first requests the access structure component $\mathcal{AS}_r$ from the CSP and decrypts the message-derived key k_d using PR-CP-ABE scheme. If successful, the user downloads the encrypted data and decrypts it with key k_d.

6 Advanced Access Control

Existing CP-ABE schemes only support basic *read* access for cloud scenario. We propose *extended path-ORAM access (ePath-ORAM-Access)* protocol by extending the existing *Path ORAM* that has been known to be the most practical ORAM scheme known-to-date that uses a small amount of client storage [12]. The extension is to facilitate integration of *Path ORAM* with our proposed schemes in Section 3 and Section 5 so that it can support advanced access control features such as *write access* on encrypted data, and *policy update*. The *ePath-ORAM-Access* protocol only employs the underlying storage-related operations of *Path-ORAM* that can continuously shuffle the data storage locations per access. As a result, the proposed *ePath-ORAM-Access* protocol supports both the advanced access control and the access pattern privacy. The details of the *ePath-ORAM-Access* is presented in Fig.7. Note that these protocols work under the assumption of an *honest-but-curious* CSP. More specifically, the CSP will follow the protocol but will try to gather or infer additional information.

For the *read* access request in *ePath-ORAM-Access* protocol, the CSP first finds $\mathcal{C}_{meta}$ for a given f_{id}, and extracts the access policy component $\mathcal{AS}_r$. Then the server executes the *Path-ORAM* algorithm to find $E_{k_d}(d)$ according to $f_{tag}(d)$. Note that we use $f_{tag}(d)$ as the identifier in the original *Path-ORAM* [12]. For the *write* access request in *ePath-ORAM-Access* protocol, the CSP first does the *PoW* check. If the client passes the check, the CSP generates a new random seed $r_w^{'}$ to update $\mathcal{AS}_w$ and let the client generate corresponding components as shown in Fig.7. Then the server

Protocol *Extended Path-ORAM Access* $^{\mathcal{CSP},\mathcal{CL}}$

$\mathcal{CL}_{(\mathsf{pk},\mathsf{sk})_{\mathcal{CL}}}$ $\qquad$ $\mathcal{CSP}_{(\mathsf{pk},\mathsf{sk})_{\mathcal{CSP}}}$

$\leftarrow \langle$establish session id, $s_{id}\rangle \rightarrow$

1 : $\mathrm{msg}_{\mathrm{req}} \leftarrow f_{id}, op$

2 : $\xrightarrow{\mathrm{msg}_{\mathrm{req}}, \mathsf{Sig}_{\mathsf{sk}_{\mathcal{CL}}}^{\mathrm{msg}_{\mathrm{req}}, s_{id}}}$

3 : $\mathrm{msg}_{\mathrm{resp}} \leftarrow \mathbf{Access}(\mathrm{msg}_{\mathrm{req}}, s_{id}, \mathsf{Sig}_{\mathsf{sk}_{\mathcal{CL}}}^{\mathrm{msg}_{\mathrm{req}}, s_{id}})$

4 : $\xleftarrow{\mathrm{msg}_{\mathrm{resp}}, \mathsf{Sig}_{\mathsf{sk}_{\mathcal{CSP}}}^{\mathrm{msg}_{\mathrm{resp}}, s_{id}}}$

(**if write request**) .....

5 : $\mathrm{msg}_{\mathrm{update}} \leftarrow \mathbf{Update\text{-}Client}(\mathrm{msg}_{\mathrm{resp}}, s_{id}, \mathsf{Sig}_{\mathsf{sk}_{\mathcal{CL}}}^{\mathrm{msg}_{\mathrm{resp}}, s_{id}})$

6 : $\xrightarrow{\mathrm{msg}_{\mathrm{update}}, \mathsf{Sig}_{\mathsf{sk}_{\mathcal{CL}}}^{\mathrm{msg}_{\mathrm{update}}, s_{id}}}$

7 : $\mathbf{Update\text{-}CSP}(\mathrm{msg}_{\mathrm{update}}, s_{id}, \mathsf{Sig}_{\mathsf{sk}_{\mathcal{CL}}}^{\mathrm{msg}_{\mathrm{update}}, s_{id}})$

Access$(\mathrm{msg}_{\mathrm{req}}, s_{id}, \mathsf{Sig}_{\mathsf{sk}_{\mathcal{CL}}}^{\mathrm{msg}_{\mathrm{req}}, s_{id}})$

1 : **if** $\mathsf{Vf}_{\mathsf{pk}_{\mathcal{CL}}}(\mathrm{msg}_{\mathrm{req}}, s_{id}, \mathsf{Sig}_{\mathsf{sk}_{\mathcal{CL}}}^{\mathrm{msg}_{\mathrm{req}}, s_{id}})$ **is true then**

2 : $\quad \mathcal{C}_{meta} \leftarrow S_{\mathcal{C}_{meta}}[\mathrm{msg}_{\mathrm{req}}.f_{id}]$

3 : $\quad E_{k_d}(d) \leftarrow$ path-ORAM(READ, $\mathcal{C}_{meta}.P_{\mathcal{C}_{data}}$, NULL)

4 : $\quad$ **if** $\mathrm{msg}_{\mathrm{req}}.op = \mathrm{READ}$ **then**

5 : $\quad\quad \mathrm{msg}_{\mathrm{resp}} \leftarrow \mathcal{C}_{meta}.\mathcal{AS}_r, E_{k_d}(d)$

6 : $\quad$ **elseif** $\mathrm{msg}_{\mathrm{req}}.op = \mathrm{WRITE}$ **then**

7 : $\quad\quad$ **if** PoW $^{\mathcal{CSP},\mathcal{CL}}$ is reject **then** abort

8 : $\quad\quad \mathrm{msg}_{\mathrm{resp}} \leftarrow \mathcal{C}_{meta}.\mathcal{AS}_r, E_{k_d}(d), \mathsf{Enc}_{\mathsf{pk}_{\mathcal{CL}}}(r_w^{'})_{r_w^{'} \in_R \mathbb{Z}}$

9 : **return** $\mathrm{msg}_{\mathrm{resp}}$

Update-Client$(\mathrm{msg}_{\mathrm{resp}}, s_{id}, \mathsf{Sig}_{\mathsf{sk}_{\mathcal{CSP}}}^{\mathrm{msg}_{\mathrm{resp}}, s_{id}})$

1 : **if** $\mathsf{Vf}_{\mathsf{pk}_{\mathcal{CSP}}}(\mathrm{msg}_{\mathrm{resp}}, s_{id}, \mathsf{Sig}_{\mathsf{sk}_{\mathcal{CSP}}}^{\mathrm{msg}_{\mathrm{resp}}, s_{id}})$ **is true then**

2 : $\quad d \leftarrow$ decrypt $E_{\psi}(k_d)$ and then $E_{k_d}(d)$

3 : $\quad r_w^{'} \leftarrow \mathsf{Dec}_{\mathsf{sk}_{\mathcal{CL}}}(\mathsf{Enc}_{\mathsf{pk}_{\mathcal{CL}}}(r_w^{'}))$

4 : $\quad$ update d **to** $d^{'}$ and generate $f_{tag}(d^{'})$

5 : $\quad \mathrm{msg}_{\mathrm{update}} \leftarrow \mathsf{Enc}_{\mathsf{pk}_{\mathcal{CSP}}}(E_{\psi}(r_w^{'}) || H(r_w^{'}))$

6 : $\quad$ **if** $f_{tag}(d^{'})$ is not duplicate **then**

7 : $\quad\quad \mathrm{msg}_{\mathrm{update}} \leftarrow$ append $E_{k_{d^{'}}}(d^{'}), E_{\psi}(k_{d^{'}})$

8 : **return** $\mathrm{msg}_{\mathrm{update}}$

Update-CSP$(\mathrm{msg}_{\mathrm{update}}, s_{id}, \mathsf{Sig}_{\mathsf{sk}_{\mathcal{CL}}}^{\mathrm{msg}_{\mathrm{update}}, s_{id}})$

1 : **if** $\mathsf{Vf}_{\mathsf{pk}_{\mathcal{CL}}}(\mathrm{msg}_{\mathrm{update}}, s_{id}, \mathsf{Sig}_{\mathsf{sk}_{\mathcal{CL}}}^{\mathrm{msg}_{\mathrm{update}}, s_{id}})$ **is true then**

2 : $\quad E_{\psi}(r_w^{'}), H(r_w^{'})) \leftarrow \mathsf{Dec}_{\mathsf{sk}_{\mathcal{CSP}}}(\mathsf{Enc}_{\mathsf{pk}_{\mathcal{CSP}}}(E_{\psi}(r_w^{'}) || H(r_w^{'})))$

3 : $\quad$ update $\mathcal{AS}_w$ with $E_{\psi}(r_w^{'}), H(r_w^{'}))$

4 : $\quad$ **if** $f_{tag}(d^{'})$ is not duplicate **then**

5 : $\quad\quad$ update $\mathcal{C}_{data}$ with $f_{tag}(d^{'}), E_{k_{d^{'}}}(d^{'}), E_{\psi}(k_{d^{'}})$

6 : $\quad\quad$ path-ORAM(WRITE, $f_{tag}(d^{'}), E_{k_{d^{'}}}(d^{'})$)

Fig. 7. The *Extended Path-ORAM Access* $^{\mathcal{CSP},\mathcal{CL}}$ protocol.

updates the stored $\mathcal{AS}_w$. Moreover, if the updated data is not duplicated, $\mathcal{C}_{data}$ will also be updated. Note that the *owner* related operations such as *policy update* in *ePath-ORAM-Access* protocol focuses on component $\mathcal{AS}_o$ and the process is similar to *write* access hence we do not show the details of the protocol.

Attribute Revocation. The immediate revocation feature is achieved by dividing the initial secret component into two parts. One is used for constructing a user's private key;

the other is used for constructing the CSP's delegation key. The delegation key makes immediate attribute revocation possible due to the destruction of the attribute-related components revoked by the CSP. Specifically, if an attribute att_x is revoked, the exponentiation on the associated ciphertext component $D_{j,i}$ will introduce a new random element θ_x (see Section 3.3 for detail). Then the users with attribute att_x can no longer decode component $D_{j,i}$, unless they satisfy the changed set of attributes in the policy for that component.

7 Security and Privacy Analysis

7.1 Privacy Analysis

Even though the confidentiality of the data is guaranteed by cryptography schemes, the meta information of the data is not confidential, which does not leak users' privacy information. Here, we analyze the privacy-preserving features of our proposed work based on the adversary model in Section 2.3.

The privacy protection in our proposed *PR-CP-ABE* includes two parts: a privacy-preserving access structure and privacy-preserving attribute-related components. The former one can prevent privacy leakage effectively by striping the attribute value from the access structure when outsourcing the data (see Section 3.1.1). We employ *composite order bilinear group* to construct the initial parameters instead of *prime order bilinear group* used in most of the existing approaches [36]. By applying orthogonal property of *composite order bilinear group*, we perturb the attribute-related components by multiplying random elements (e.g., $Z_{j,1,i}$ in Section 3.3), and the rest of the components in privacy-preserving access structure, i.e., (M, ρ), can ensure subset generation in the decryption phase as described in Section 3.3. Thus, without losing usability, the access policy does not leak the privacy information.

Moreover, our framework addresses access pattern privacy issue by using the *ePath-ORAM-Access* protocol, where path-ORAM is integrated with the basic read/write infrastructure to access encrypted data in the cloud storage server (see Section 6). The ORAM mechanism has been shown to be practical in ensuring that an adversary who can observe the physical storage locations accessed by a user has negligible advantage of learning anything about the true (logical) access patterns [12], and hence our *ePath-ORAM-Access* protocol can also prevent an adversary from inferring privacy sensitive access pattern information.

7.2 Security Analysis

7.2.1 Attribute-based Access Control Framework

Security proof of PR-CP-ABE. Our proposed PR-CP-ABE has been proved to be secure against Chosen-plaintext Attack (CPA) using proof approach that is similar to that used in [37], more specifically, as presented in Theorem 1.

Theorem 1. *Under the decisional q-parallel BDHE assumption, PR-CP-ABE scheme is secure if an adversary does not have non-negligible advantage in polynomial time in the simulation game.*

In addition, PR-CP-ABE also provides collusion-resistant property, forward and backward secrecy property. We refer the readers to our conference version [1] for the details of the proof and analysis.

Security analysis of *ePath-ORAM-Access*. The security of *ePath-ORAM-Access* relies on the basic, secure infrastructure, namely, the underlying public key crypto-system and collision resistant hash function. In the access protocol, as shown in Fig.7, each entity has a public/private key pair. To resist the man-in-the-middle attack, the protocol requires each sender to sign each message she sends with her private key to ensure the integrity of the message. In addition, each conversation starts with establishing a secure random one-time session identifier. Such an identifier is also included in the signature to resist replay attacks, where the adversary may replay the authentication fragment collected in an older session to claim its privilege. Besides, critical components such as $r_w^{'}$ and $r_o^{'}$ are used for verifying that a user's access privilege is encrypted using the receiver's public key (see **Access** function in Fig.7). As the underlying basic primitives of the protocols are secure and the security of PR-CP-ABE has been proved, we see that the *ePath-ORAM-Access* protocol is also secure.

7.2.2 Secure Deduplication Mechanism

Security analysis of sever-aided MLE. Unlike convergent encryption (CE), which is the most prominent instantiation of MLE, the sever-aided MLE employs a key server - a trusted TPA - to help generate message-derived key (see Section 3.1.3). As the trusted TPA is assumed to be not compromised in our adversary model, we only need to consider two cases: (i) offline brute-force dictionary attacks and (ii) compromised authorized user attacks.

As discussed in [20], CE is susceptible to an offline brute-force dictionary attack as the key is derived from the message using an approach such as a publicly known collision resistant hash function. It is possible for an adversary to launch the offline brute-force attack. In our secure deduplication mechanism, the server-aided MLE generates the key with the help of a key server and the underlying RSA-OPRF protocol. The message-derived key is random. Even if the adversary has plaintext and corresponding ciphertext, without the access to the key server, he cannot establish the link. Thus, the sever-aided MLE can achieve semantic security.

Another brute-force attack is when an adversary has compromised one or more authorized users, which means that the adversary has remote access to the key server but does not have the key server's secret key. Note that the client application is assumed to be trusted. As presented in [20], the sever-aided MLE also makes it much more difficult to do brute-force attacks. In addition, such brute-force attacks cannot be launched as there is a trusted TPA; such attacks are easy to detect from the key server side.

Security analysis of PoW/PoO. Essentially, the proposed PoW/PoO protocol is an interactive proof system, namely, the client (i.e., prover) tries to prove its *write/ownership* privilege to the CSP (i.e., verifier). Hence, we analyze the security proprieties as follows: (i) *completeness*, where for all the inputs that correspond to the prover's valid authorized privileges, the probability of the verifier accepting should be 1; and (ii) *soundness*, where for all the inputs that do not correspond to the prover's valid authorized privileges, the probability of the verifier rejecting it should be negligible.

Completeness. In the *PoW* and *d-PoO* mechanism (see Section 5.1.1), the proofs of a client's *write* and *ownership* privileges are based on the verification of the decryption ability of the PR-CP-ABE scheme. As presented in Section 3, any authorized client has the ability to decrypt the corresponding components defined in our proposed outsourced data model according to their attribute identities and the access policy. Thus, any authorized user can prove their corresponding privilege to the CSP. In the *pom-PoO* mechanism, the proof of a client's *ownership* privilege is based on the verification of the Merkle-based path-ORAM tree. The existing work in [38] provides the proof of the completeness property of the Merkle tree based solution. In summary, the completeness property of PoW/PoO is guaranteed.

Soundness. As the security of the PR-CP-ABE scheme has been proved, the only way to compromise the *PoW* and *d-PoO* mechanism is to forge a hash value (i.e., $H(r_w)$) rather than compromising the PR-CP-ABE scheme. If the adversary has the negligible advantage ϵ to forge $H(r_w)$, it will have the advantage ϵ to compromise the collision-resistant hash function. According to our assumption, however, it is impossible to break the collision-resistant hash function; hence, the adversary does not have negligible advantage to compromise the *PoW* and *d-PoO*. Similar to the analysis in [38], the adversary also has no advantage to break the Merkle-tree based proof. As a result, the soundness property of PoW/PoO is guaranteed.

Threats in multi-owner cross-user scenario. The proposed framework supports multi-owner cross-user scenario, where one physical copy of data may have multiple owners, and each owner can share the data with a group of authorized users with different permissions. Thus, two threats need to be considered: (i) deletion across multi-owner scenario, where one owner deletes the data that is also owned by another owner, and (ii) side-channel information leakage in cross-user scenario, where one authorized user may acquire additional information related to other users.

In our proposed outsourced data model (see Section 4), the meta-data $\mathcal{C}_{meta}$ and the secure-data $\mathcal{C}_{data}$ are separated. The access related operation focuses on $\mathcal{C}_{meta}$, while the deduplication related operation is executed on $\mathcal{C}_{data}$. This means multiple owners will have different meta-data items but the same secure-data component, as shown in Fig.3. If one owner deletes the data, what it actually deletes is the meta-data item including the authorized access permissions that the owner has specified. In addition, the owner also cannot learn the information about other owners who have the same data because the link (i.e., $P_{\mathcal{C}_{data}}$) is only available to the CSP. Besides, in the cross-user scenario, the only side-channel information that each authorized user can acquire is the privacy-preserving access policy, where the attribute value is hidden. Thus, they cannot infer the information related to other authorized users who have access to the same data. Furthermore, the CSP can certainly relate a set of attributes to an encrypted file. But note that each user can have several attributes, and no one other than the TPA can relate his attributes to the related identities. Thus, the CSP will not be able to count the number of users associated with a given set of attributes that relates to a particular encrypted file. Hence, identifying a popular file is not possible.

7.3 Universally Composable Security Analysis

Universally Composable (UC) security is a framework proposed in [39] as an approach to define security for protocols such that security-preserving composition is possible. The UC framework allows for a modular design and analysis of protocols. Informally, if a protocol UC securely realizes some ideal functionality $\mathcal{F}$, then the protocol will behave as $\mathcal{F}$ in any arbitrary computational environment it is run.

Theorem 2. *The attribute-based access control with secure deduplication framework π_{ABAC} securely realizes $\mathcal{F}_{ABAC}$ in the ($\mathcal{F}_{ABE}$, $\mathcal{F}_{MLE}$) model, where $\mathcal{F}_{ABAC}$ is the defined ideal functionality for the framework, and $\mathcal{F}_{ABE}$, $\mathcal{F}_{MLE}$ are the ideal functionality for the PR-CP-ABE scheme and the server-aided MLE scheme, respectively.*

$\mathcal{F}_{ABE}$ and $\mathcal{F}_{MLE}$ are ideal functionalities that capture the secure requirements of PR-CP-ABE scheme and server-aided MLE scheme, respectively. All involved parties have secure access to the ideal functionalities. Our framework is implemented by replacing the ideal functionality with a protocol securely realizing the functionality. As the *composition theorem* presented in [39], the security can be guaranteed. The specific proof procedures are similar to the proof presented in [40], where a signcryption scheme and a public key infrastructure scheme are integrated to construct a secure messaging scheme. Due to space limitations, we present the proof details in online supplementary materials.

8 Performance Analysis

8.1 Implementation of Prototype Framework

We have implemented a prototype of our framework using Python. The implementation of the core modules (e.g. PR-CP-ABE, Symmetric encryption, and Message-derived key generation) are based on Charm [41], which is a Python-based framework for rapidly prototyping advanced cryptographic systems. In Charm, the performance intensive mathematical operations rely on native C libraries of GMP (GNU Multiple Precision Arithmetic Library) and PBC (Pairing-based Crypto Library), while cryptographic modules themselves are written in Python [41].

We use the Hierarchical File System (HFS), the native file system of Mac OS, as *Data Storage* to store the encrypted file. In our prototype, the *Meta-data Storage* and *Rapid Storage* are represented by two hash table structures that are loaded in the RAM. As we mentioned before, from the perspective of the CSPs, the main concern is the storage space. Thus, the focus of performance analysis on storage is measuring the storage space savings. Our simulated outsourced files are based on surveys from *Gartner, Inc* and *Nasuni, Inc*. The benchmarks environment is Mac OS with 4 2.5 GHz Intel Core i7 processors and 16 GB DDR3 memory.

8.2 Efficiency of PR-CP-ABE

The PR-CP-ABE is the core module, so we evaluate its performance independently. As the authority and the CSPs typically have powerful and distributed server resources in the real world, we only consider the performance of PR-CP-ABE from the perspective of users, e.g., the cost of *encryption*, *decryption* and *key application*. Here the *key application* refers to the process where a user sends her attribute sets to

TABLE 3
Comparison of elements size in the network transmission

Entities	Our scheme	[10]	[29]	[30]
Authority ↔ User	$(2+n_i)\|\mathbb{G}\|$	$(2+n_i)\|\mathbb{G}\|$	$(1+2n_i)\|\mathbb{G}\|$	$(1+2n_i)\|\mathbb{G}\|$
Authority ↔ Owner	$(2+n_a)\|\mathbb{G}\|+\|\mathbb{G}_T\|$	$2\|\mathbb{G}\|+\|\mathbb{G}_T\|$	$(1+3n_a)\|\mathbb{G}\|+\|\mathbb{G}_T\|$	$2\|\mathbb{G}\|+\|\mathbb{G}_T\|$
CSP ↔ Owner	$2((2m+1)\|\mathbb{G}\|+\|\mathbb{G}_T\|)$	$(2m+1)\|\mathbb{G}\|+\|\mathbb{G}_T\|$	$3m\|\mathbb{G}\|+\|\mathbb{G}_T\|$	$2m\|\mathbb{G}\|+(m+1)\|\mathbb{G}_T\|$
CSP ↔ User	$(4m+3)\|\mathbb{G}\|+2\|\mathbb{G}_T\|$	$(2m+3)\|\mathbb{G}\|+\|\mathbb{G}_T\|$ $+m\|\mathbb{Z}_p\|$	$(3m+2n_i)\|\mathbb{G}\|+\|\mathbb{G}_T\|$	$(3m+2n_i)\|\mathbb{G}\|+\|\mathbb{G}_T\|$ $+(m/2\|n_u\|+log(n_u+1))\|\mathbb{Z}_p\|$

[1] Let $|\mathbb{G}|$, $|\mathbb{G}_T|$ and $|\mathbb{Z}_p|$ be the elements size in $\mathbb{G}, \mathbb{G}_T$ and $\mathbb{Z}_p$, respectively.
[2] Let n_i, n_u, n_a be the number of attributes of user i, number of users, and universal attributes number, respectively.
[3] Let m represent the number of attributes in access policy.

the authority and gets back the generated private keys from the authority.

8.2.1 Encryption and Decryption

The computation time of encryption and decryption is related to the number of attributes in the policy. Thus, we design 11 test cases of access policy (all *AND* logical conjunction) with varying numbers of attributes: 2-11, 15, and 20. For each case, we encrypt and decrypt the message-derived key using PR-CP-ABE and compute the cost, as shown in Fig.8.

We see that both the encryption and decryption times are approximately linear to the number of attributes. The decryption time is much lower than the encryption time. Even with 20 attributes in the access policy, the encryption time is under 0.35 seconds. We also compare our proposed PR-CP-ABE with similar other schemes [10], [29], [30] which support attribute revocation. Even though our scheme does not have the best performance compared to these schemes, it has comparable encryption times as that of [10] and comparable decryption times as that of [10] and [30]. With regards to the trade-offs between the features (i.e., revocation and privacy preservation) and efficiency, our framework provides a more complete and privacy-preserving access features; while not the best, our scheme provides an acceptable level of efficiency.

8.2.2 Key application

While the cost of encryption/decryption is in milliseconds, the cost of the authority related activities, e.g., the *key application*, is in seconds [42]. In the key application process, the network transmission adds the major cost. However, the network transmission time depends on many factors like distance, network quality, etc., which are beyond the scope of our paper. Here, we focus on the theoretical analysis in terms of the sizes of the transferred elements in the network. In Table 3, we compare our scheme with the existing schemes [10], [29], [30]. According to the analysis in Section 3.2, the private keys and public parameters contribute to the elements that are transfered between authority and a user and between the authority and the owner, respectively. The major communication cost between CSP and user/owner is the transmission of encrypted and re-encrypted ciphertexts. The scheme in [10] has the smallest element size. However, our scheme is better than other two schemes with regards to the communication cost and provides more protection features overall, including that of access structure when compared to the scheme in [10].

8.3 Impact of File Size

Next, we evaluate how file sizes impact performance.

Setting: We generate seven random files with file sizes: 1KB, 10KB, 100KB, 1MB, 10MB, 100MB, and 1GB. Then we simulate and record the time of a user's process on the test data including message-derived key generation, encryption/decryption, and data serialization (read/write/format). Note that the time measurement is based on *timeit* module built in python library. And the result is the average value based on ten experimental runs.

As shown in Fig.9a, the time of message-derived key generation is constant (about 20 ms). For file sizes less than 100MB, the encryption/decryption time, including PR-CP-ABE and AES symmetric encryption, is lower than 2 seconds. When it includes file I/O processes (read/write), under the same situation, the processing time is lower than 9 seconds (Fig.9b). Thus, it is necessary to compare times for sub-processes. The file I/O process has the primary impact on the performance when the file size is more than 10MB (Fig.9c and Fig.9d).

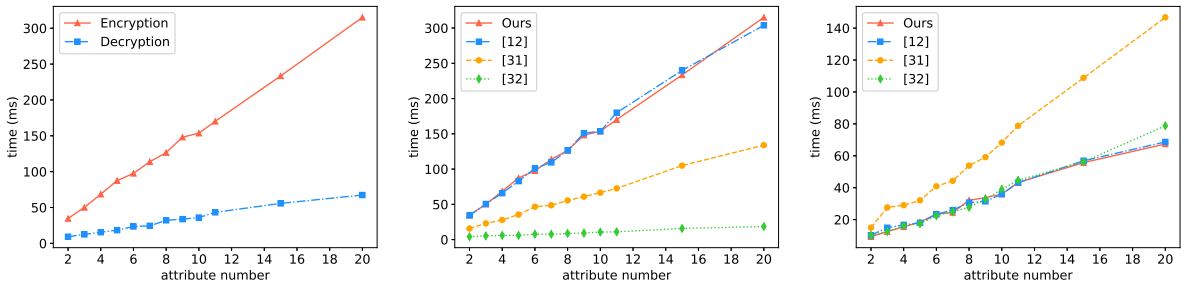
Overall Performance. To evaluate the overall performance, we simulated a CSP using a virtual private server that is located in San Francisco. Specifically, we measure the time taken against the file size from both client and CSP sides, respectively. From the perspective of client, the experiment includes three test cases: initial upload, read access, and update access. From the CSP side also, we include three test cases: processing client's read access, and processing client's upload/update access with or without duplication. The processing time taken against the file size is depicted in Fig.10. Note that the time of client's operations includes data transmission time and security related processing time. In the simulated CSP server, the block size and block count for the extended Path-ORAM scheme are set as 100KB and 2^9 - 1, respectively, in our simulation.

8.4 Storage size analysis

8.4.1 Simulation of outsourced files

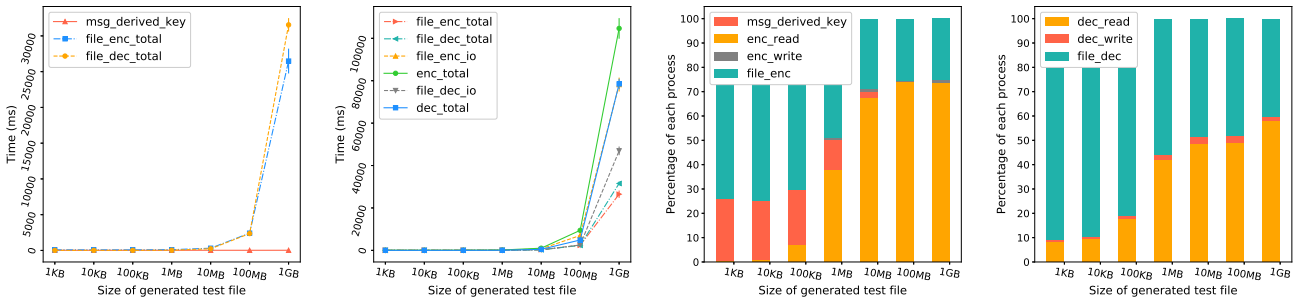
As it is difficult to check the users' outsourced files in the real scenarios, we simulate the possible distribution of outsourced files. First, we prepare a file set used to simulate the outsourced files. Specifically, in our setting, we generate 600 random files with following size distributions: 500 files from 1KB to 1000KB with step 2KB, 100 files from 1MB to 200MB with step 2MB. Then we select the users' outsourced files using the following three methods to simulate the real scenarios: (i) *Random*. Uniformly select the outsourced files from the file set. (ii) *Gartner*. Based on Gartner's report [1] that shows that 75% of files in the cloud are less than 10MB. (iii)

1. Magic Quadrant for Public Cloud Storage Services. Gartner, Inc.



(a) Efficiency of PR-CP-ABE (b) Comparison of encryption (c) Comparison of decryption

Fig. 8. Efficiency and comparison of PR-CP-ABE



(a) Time of crypto-process (b) Time of all components (c) Proportion of encryption (d) Proportion of decryption

Fig. 9. The effect of file size on the processing time, and proportion of each processing component.

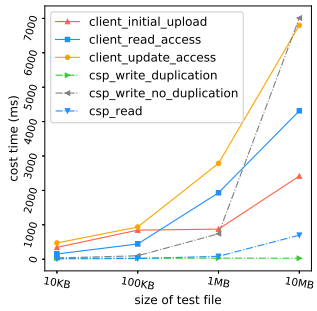


Fig. 10. Overall performance based on file sizes.

Nasuni. Based on Nasuni's survey [2] of their actual Nasuni enterprise customers.

We simulate 1000 times the file outsourcing process under each simulation strategy. As we can see in the figure, most of the outsourced files are located in the interval where the file size is less than 10MB in the *Gartner* simulation. Moreover, in the *Nasuni* simulation (the closest methods to the real scenario), the distribution is similar to *Random* simulation except for its narrower fluctuation for each file size.

8.4.2 *Effect of secure deduplication*

To evaluate storage savings of the secure deduplication mechanism, we use as baseline the outsourced encrypted files without deduplication under three simulation scenarios described in Section 8.4.1. The result is shown in Fig.11.

The left two figures in Fig.11 show the total storage for each file under scenarios of baseline (*Random*, *Gartner* and *Nasuni*) and our proposed secure deduplication approach, respectively. As a result of deduplication, the storage size of applying secure deduplication is linear to the number of file identifiers instead of the actual total number of files. The right figure in Fig.11 shows the total storage for each file. Compared to the three baselines, our proposed secure deduplication mechanism could save about 94%, 77% and 92% storage, respectively.

9 Related Work

Sahai and Waters propose the first Attribute-based Encryption scheme in [43] that combines access control function and encryption, and allows specification of access policies based on users' attributes to support dynamic, attribute based access control. Bethencourt et al. propose CP-ABE in [8] where an access structure and the users' keys are associated with the ciphertexts and users' attributes, respectively. Here, the data owner who encrypts the data determines the policy. The CP-ABE scheme hence introduces a new way to protect outsourced data. Several researchers have attempted to make access policy flexible. There are three types of access structures proposed in the literature: *AND-gates*, *LSSS* matrix and *tree*. In [9], Waters proposes the first LSSS matrix based CP-ABE. He shows that its expressiveness is not lower than that of the tree structure.

Two key privacy issues have been addressed by researchers. In [44], Hur addresses the issue of a private key generator that discloses users' privacy as it is in full control of the private keys. Another issue is that of leakage of users' sensitive attribute information through the access structure. More recently, various CP-ABE schemes, such as [31], [32], [45], that support hiding policies have been proposed. However, the key issue with these schemes is the limited expressiveness of policies through an *AND-gate* access structure. Lai et al. present a CP-ABE scheme that supports policy hiding by inner product predicate encryption, which is proven to be *fully* secure [45]. They also propose a CP-ABE scheme in [31] that uses a LSSS matrix to support partially hidden policy.

2. The State of Cloud Storage 2015 Industry Report. Nasuni, Inc.

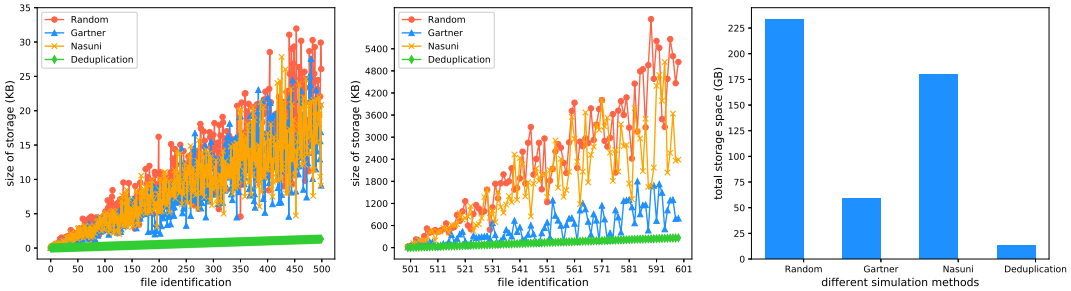


Fig. 11. Analysis result on storage savings.

In existing work addressing the revocation issue, expiration time has been added to each attribute to support revocation; however, immediate revocation is not supported. There still exist the issues of degradation in security and scalability related to forward and backward security. Several researchers propose schemes that support immediate revocation of attributes. For more efficient revocation of attributes, Hur et al. in [28], [30], [44] propose using secure two-party computation to generate users' private keys and Zu et al. in [10] propose a revocable CP-ABE scheme.

Several schemes based on Oblivious RAM [46] have been proposed in the literature to prevent leakage of privacy sensitive information through access patterns [6], [7], [11], [12]. ORAM makes the access patterns independent of the inputs to the algorithm. Goodrich et al. propose practical oblivious storage in [11], but their scheme does not consider a strong attacker model. Nabeel and Bertino in [47] propose an approach that is based on two layers of encryption with broadcast encryption; however, it requires a decomposition of a policy. Maffei et al. in [7] propose a framework that uses ORAM, zero-knowledge proof and predicate encryption; but it does not consider flexible access policy and secure data deduplication issues.

Most of the existing secure depulication solutions utilize convergent encryption (CE) algorithms [17] to enable deduplication over encrypted data. In convergent encryption, a user derives a convergent key by using a cryptographic hash algorithm to get the hash value of a file. Thus, the same ciphertext is deterministically computed from the same plaintext. To formalize the security of CE, Message-locked encryption (MLE) [18], [19] has been proposed, which provides a formal framework for CE. However, both CE and MLE lose semantic security because of their deterministic property. As the entropy of the convergent keys is dependent on the message, the key space is very small. Thus, CE/MLE does not prevent brute-force attacks [20].

Several researchers have tried to strengthen the key generation method to preserve the complexity of convergent key space. Li et al. focus on reliable convergent key management in [48], while Chen et al. focus on reducing metadata set in the case of large file deduplication in [49]. In [20], [50], an additional key server is introduced into the scheme, where the convergent keys are independent of the message. Bellare et al. propose DupLESS, a server-aided encryption scheme against brute-force attacks in [20]. They employ an interactive key generation protocol based on RSA-OPRF [50]. In [51], a secure cross-user deduplication scheme that supports client-side encryption without requiring any additional independent servers has been proposed. As our cryptography-based access control framework relies on a trusted TPA, we prefer the sever-aided MLE that also is adopted in DupLESS. Besides, our secure deduplication also supports cross-user deduplication.

The secure deduplication issue has not been addressed adequately for ABE schemes. To the best of our knowledge, the only two related works are by Tang et al. in [34] and Li et al. in [52]. In [34], Tang et al. propose ciphertext deduplication under ABE by reusing secrets in a complete access tree. Thus, the access policy in their scheme is limited to tree-based structure, and they need a global table to store all the node structures in the access tree. In the scheme proposed in [52], access control is only limited to *read access* with revocation. Neither of these considers the scenario of data sharing and a diverse set of access rights by adopting ABE in the cloud storage.

10 Conclusion

We have proposed an integrated framework for privacy-preserving attribute-based access control to protect sensitive outsourced data. The proposed approach supports user/organization-centric policy management of data oursourced to a cloud storage, immediate privilege revocation, and privacy protection. The proposed framework also supports secure deduplication that helps eliminate redundant encrypted data in the storage. The proposed scheme has been shown to satisfy the security and privacy requirements and demonstrate good performance with regards to the communication cost. The security proof shows that the proposed system achieves CPA security under the decisional parallel Bilinear Diffie-Hellman Exponent assumption. A key future direction includes integrating the proposed solution in an existing cloud storage service.

Acknowledgment

This research work has been supported by the National Science Foundation grant DGE-1438809.

References

[1] R. Xu and J. B. Joshi, "An integrated privacy preserving attribute based access control framework," in *9th CLOUD*. IEEE, 2016, pp. 68–76.

[2] Y. Zhang, M. Qiu, C.-W. Tsai, M. M. Hassan, and A. Alamri, "Health-cps: Healthcare cyber-physical system assisted by cloud and big data," *IEEE Systems Journal*, vol. 11, no. 1, pp. 88–95, 2017.

[3] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica *et al.*, "A view of cloud computing," *Communications of the ACM*, vol. 53, no. 4, pp. 50–58, 2010.

[4] D. J. Abadi, "Data management in the cloud: limitations and opportunities." *IEEE Data Eng. Bull.*, vol. 32, no. 1, pp. 3–12, 2009.

[5] H. Takabi, J. B. Joshi, and G.-J. Ahn, "Security and privacy challenges in cloud computing environments," *S&P (Oakland)*, vol. 8, no. 6, pp. 24–31, 2010.
[6] M. S. Islam, M. Kuzu, and M. Kantarcioglu, "Access pattern disclosure on searchable encryption: Ramification, attack and mitigation." in *NDSS*, vol. 20, 2012, p. 12.
[7] M. Maffei, G. Malavolta, M. Reinert, and D. Schröder, "Privacy and access control for outsourced personal records," in *S&P*. IEEE, 2015, pp. 341–358.
[8] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attribute-based encryption," in *S&P (Oakland)*. IEEE, 2007, pp. 321–334.
[9] B. Waters, "Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization," in *PKC*. Springer, 2011, pp. 53–70.
[10] L. Zu, Z. Liu, and J. Li, "New ciphertext-policy attribute-based encryption with efficient revocation," in *CIT*. IEEE, 2014, pp. 281–287.
[11] M. T. Goodrich, M. Mitzenmacher, O. Ohrimenko, and R. Tamassia, "Practical oblivious storage," in *2ed CODASPY*. ACM, 2012, pp. 13–24.
[12] E. Stefanov, M. Van Dijk, E. Shi, C. Fletcher, L. Ren, X. Yu, and S. Devadas, "Path oram: An extremely simple oblivious ram protocol," in *CCS*. ACM, 2013, pp. 299–310.
[13] D. Apon, J. Katz, E. Shi, and A. Thiruvengadam, "Verifiable oblivious storage," in *PKC*. Springer, 2014, pp. 131–148.
[14] E. Stefanov and E. Shi, "Multi-cloud oblivious storage," in *CCS*. ACM, 2013, pp. 247–258.
[15] D. T. Meyer and W. J. Bolosky, "A study of practical deduplication," *TOS*, vol. 7, no. 4, p. 14, 2012.
[16] N. Mandagere, P. Zhou, M. A. Smith, and S. Uttamchandani, "Demystifying data deduplication," in *Proceedings of the ACM/IFIP/USENIX Middleware'08 Conference Companion*. ACM, 2008, pp. 12–17.
[17] J. R. Douceur, A. Adya, W. J. Bolosky, P. Simon, and M. Theimer, "Reclaiming space from duplicate files in a serverless distributed file system," in *22nd ICDCS*. IEEE, 2002, pp. 617–624.
[18] M. Bellare, S. Keelveedhi, and T. Ristenpart, "Message-locked encryption and secure deduplication," in *EUROCRYPT*. Springer, 2013, pp. 296–312.
[19] M. Abadi, D. Boneh, I. Mironov, A. Raghunathan, and G. Segev, "Message-locked encryption for lock-dependent messages," in *CRYPTO*. Springer, 2013, pp. 374–391.
[20] S. Keelveedhi, M. Bellare, and T. Ristenpart, "Dupless: server-aided encryption for deduplicated storage," in *USENIX Security*, 2013, pp. 179–194.
[21] A. Beimel, "Secure schemes for secret sharing and key distribution," Ph.D. dissertation, Technion-Israel Institute of technology, Faculty of computer science, 1996.
[22] D. Boneh and M. Franklin, "Identity-based encryption from the weil pairing," in *CRYPTO*. Springer, 2001, pp. 213–229.
[23] D. Boneh, E.-J. Goh, and K. Nissim, "Evaluating 2-dnf formulas on ciphertexts," in *Theory of cryptography*. Springer, 2005, pp. 325–341.
[24] D. Boneh, "Bilinear groups of composite order," in *Pairing 2007*. Springer, 2007, pp. 1–1.
[25] A. Boldyreva, "Threshold signatures, multisignatures and blind signatures based on the gap-diffie-hellman-group signature scheme," in *PKC*. Springer, 2003, pp. 31–46.
[26] B. Pinkas and T. Reinman, "Oblivious ram revisited," in *CRYPTO*. Springer, 2010, pp. 502–519.
[27] V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-based encryption for fine-grained access control of encrypted data," in *13th CCS*. Acm, 2006, pp. 89–98.
[28] X. Xie, H. Ma, J. Li, and X. Chen, "New ciphertext-policy attribute-based access control with efficient revocation," in *ICT*. Springer, 2013, pp. 373–382.
[29] S. Yu, C. Wang, K. Ren, and W. Lou, "Attribute based data sharing with attribute revocation," in *5th ASIA CCS*. ACM, 2010, pp. 261–270.
[30] J. Hur and D. K. Noh, "Attribute-based access control with efficient revocation in data outsourcing systems," *TPDS*, vol. 22, no. 7, pp. 1214–1221, 2011.
[31] J. Lai, R. H. Deng, and Y. Li, "Expressive cp-abe with partially hidden access structures," in *7th ASIA CCS*. ACM, 2012, pp. 18–19.
[32] X. Li, D. Gu, Y. Ren, N. Ding, and K. Yuan, "Efficient ciphertext-policy attribute based encryption with hidden policy," in *IDCS*. Springer, 2012, pp. 146–159.
[33] S. Agrawal and M. Chase, "Fame: Fast attribute-based message encryption," in *CCS*. ACM, 2017, pp. 665–682.
[34] H. Tang, Y. Cui, C. Guan, J. Wu, J. Weng, and K. Ren, "Enabling ciphertext deduplication for secure cloud storage and access control," in *11th ASIA CCS*. ACM, 2016, pp. 59–70.
[35] F. Armknecht, C. Boyd, G. T. Davies, K. Gjøsteen, and M. Toorani, "Side channels in deduplication: trade-offs between leakage and efficiency," in *ASIA CCS*. ACM, 2017, pp. 266–274.
[36] D. Freeman, "Converting pairing-based cryptosystems from composite-order groups to prime-order groups," *EUROCRYPT*, pp. 44–61, 2010.
[37] L. Cheung and C. Newport, "Provably secure ciphertext policy abe," in *14th CCS*. ACM, 2007, pp. 456–465.
[38] S. Halevi, D. Harnik, B. Pinkas, and A. Shulman-Peleg, "Proofs of ownership in remote storage systems," in *18th CCS*. ACM, 2011, pp. 491–500.
[39] R. Canetti, "Universally composable security: A new paradigm for cryptographic protocols," in *42nd FOCS*. IEEE, 2001, pp. 136–145.
[40] K. Gjøsteen and L. Kråkmo, "Universally composable signcryption," in *European Public Key Infrastructure Workshop*. Springer, 2007, pp. 346–353.
[41] J. A. Akinyele, C. Garman, I. Miers, M. W. Pagano, M. Rushanan, M. Green, and A. D. Rubin, "Charm: a framework for rapidly prototyping cryptosystems," *Journal of Cryptographic Engineering*, vol. 3, no. 2, pp. 111–128, 2013.
[42] B. Lang, R. Xu, and Y. Duan, "Extending the ciphertext-policy attribute based encryption scheme for supporting flexible access control," in *SECRYPT*. IEEE, 2013, pp. 1–11.
[43] A. Sahai and B. Waters, "Fuzzy identity-based encryption," in *EUROCRYPT*. Springer, 2005, pp. 457–473.
[44] J. Hur, "Improving security and efficiency in attribute-based data sharing," *TKDE*, vol. 25, no. 10, pp. 2271–2282, 2013.
[45] J. Lai, R. H. Deng, and Y. Li, "Fully secure cipertext-policy hiding cp-abe," in *ISPEC*. Springer, 2011, pp. 24–39.
[46] O. Goldreich and R. Ostrovsky, "Software protection and simulation on oblivious rams," *Journal of the ACM*, vol. 43, no. 3, pp. 431–473, 1996.
[47] M. Nabeel and E. Bertino, "Privacy preserving delegated access control in public clouds," *TKDE*, vol. 26, no. 9, pp. 2268–2280, 2014.
[48] J. Li, X. Chen, M. Li, J. Li, P. P. Lee, and W. Lou, "Secure deduplication with efficient and reliable convergent key management," *TPDS*, vol. 25, no. 6, pp. 1615–1625, 2014.
[49] R. Chen, Y. Mu, G. Yang, and F. Guo, "Bl-mle: block-level message-locked encryption for secure large file deduplication," *TIFS*, vol. 10, no. 12, pp. 2643–2652, 2015.
[50] M. Naor and O. Reingold, "Number-theoretic constructions of efficient pseudo-random functions," *Journal of the ACM*, vol. 51, no. 2, pp. 231–262, 2004.
[51] J. Liu, N. Asokan, and B. Pinkas, "Secure deduplication of encrypted data without additional independent servers," in *22nd CCS*. ACM, 2015, pp. 874–885.
[52] J. Li, C. Qin, P. P. Lee, and J. Li, "Rekeying for encrypted deduplication storage," in *46th IEEE/IFIP DSN*. IEEE, 2016, pp. 618–629.

Runhua Xu is a PhD student of School of Computing and Information at the University of Pittsburgh. He received his M.S. in Computer Science and B.E. in Software Engineering degrees from Beihang University and Northwestern Polytechnical University, China, respectively. His research interests include Access Control, Applied Cryptography, Data Security and Privacy.

James Joshi is a professor of SCI at the University of Pittsburgh. He received his MS in Computer Science and PhD in Computer Engineering degrees from Purdue University in 1998 and 2003, respectively. He is an elected Fellow of the SIRI and is a senior member of the IEEE and a distinguished member of the ACM. His research interests include Access Control Models, Security and Privacy of Distributed Systems, Trust Management and Information Survivability.

Prashant Krishnamurthy is a professor of SCI at the University of Pittsburgh. His research interests include wireless network security, wireless data networks, position location in indoor wireless networks, and radio channel modeling for indoor wireless networks. He is a member of the IEEE.