An Efficient Method for Realizing Contractions of Access Structures in Cloud Storage

Shuai Feng and Liang Feng Zhang

Abstract—In single-cloud storage, ciphertext-policy attribute-based encryption (CP-ABE) allows one to encrypt any data under an access structure to a cloud server, specifying what attributes are required to decrypt. In multi-cloud storage, a secret sharing scheme (SSS) allows one to split any data into multiple shares, one to a single server, and specify which subset of the servers are able to recover the data. It is an interesting problem to remove some attributes/servers but still enable the remaining attributes/servers in every authorized set to recover the data. The problem is related to the contraction problem of access structures for SSSs. In this paper, we propose a method that can efficiently transform a given SSS for an access structure to SSSs for contractions of the access structure. We show its applications in solving the attribute removal problem in the CP-ABE based single-cloud storage and the data relocating problem in multi-cloud storage. Our method results in solutions that require either less server storage or even no additional server storage.

Index Terms—Cloud storage, access structure, contraction, linear secret sharing, attribute-based encryption.

I. INTRODUCTION

WITH the rapid development of cloud computing in recent years, *cloud storage* [1], [2] has moved to the mainstream of storage technology. It allows one to lease computing resources from cloud service providers in a pay-per-use manner and remotely store/access important data, without need to build local data centers (including expensive software and hardware infrastructures) from scratch. However, many organizations are still reluctant to use cloud to store sensitive data. The reason is that they may lose control of the data, and information leakage may occur due to unauthorized access [3]. How to ensure the *confidentiality* of cloud data is an important problem.

Storing data with cloud may use two different models: *single-cloud storage* and *multi-cloud storage*. Single-cloud storage means storing data with a single cloud server. In this model, the confidentiality of data may be ensured by the user encrypting the data and uploading the ciphertexts to a cloud server. For example, *ciphertext-policy attribute-based encryption* (CP-ABE) [4], [5] is a commonly used encryption technology for fine-grained access control that allows one to set a *policy* to specify who are eligible to decrypt a ciphertext. More precisely, every CP-ABE ciphertext is associated with a policy; every *user* of the data gets a private key associated with several attributes from a set of n *attributes* and is eligible to decrypt if and only if its attributes satisfy the policy.

S. Feng and L.F. Zhang (Corresponding author) are with the School of Information Science and Technology, ShanghaiTech University, Shanghai, PR China. E-mail: {fengshuai,zhanglf}@shanghaitech.edu.cn

In multi-cloud storage [1], [6]–[8], one may generate n *shares* of the data and store each share with a different cloud server to enforce the following policy: the data can be reconstructed if $\geq t+1$ out of the n shares are available but any $\leq t$ shares leak information about the data. In this model, special techniques for splitting data into shares, such as homomorphic secret sharing (HSS) [9], have found interesting applications in the field of outsourcing computations [10], [11]. Such techniques allow each server to compute a function on its share to produce a *partial result* and finally a user of the data can reconstruct the function's output from all partial results.

A critical technology used in both single-cloud storage and multi-cloud storage is *linear secret sharing schemes* (LSSSs) [12]. A *secret sharing scheme* (SSS) [13], [14] for a set $\mathcal{P} = \{P_1, \ldots, P_n\}$ of n *participants* allows a *dealer* to generate n *shares* for a *secret* s, one for each participant, such that any *authorized subset* of $\mathcal{P}$ can reconstruct s with their shares but any *unauthorized subset* learns no information about s. The set Γ of all authorized subsets is called an *access structure* and the SSS is said to *realize* Γ. In general, a set Γ of subsets of $\mathcal{P}$ is qualified as an access structure if the superset of any set in Γ remains in Γ. SSSs were introduced by Shamir [13] and Blakley [14] for *threshold* access structures and then extended to any *general* access structures by Ito et al. [15]. An SSS is *linear* if both the *share generation* and the *secret reconstruction* can be accomplished with linear operations. Since [13]–[15], SSSs have been one of the most important building blocks of cryptographic protocols [16]–[18]. In particular, the n attributes in the CP-ABE based single-cloud storage model and the n servers in multi-cloud storage model may play the roles of the n participants in SSSs, respectively, and the policies in both models may play the role of access structures.

In this paper, we consider application scenarios where part of the attributes/servers have to be removed such that the left attributes/servers in every authorized subset remain eligible to access data, in order to make the access policies *less restrictive*. Such scenarios may appear in both storage models. For example, in the single-cloud storage model, a patient Alice may have encrypted her electronic health records (EHRs) [19]–[21] as a CP-ABE ciphertext with an access policy `'hospital A'` $\wedge$ `'branch 1'` $\wedge$ `'respiratory'` such that Bob, a respiratory physician in the branch 1 of hospital A can decrypt the ciphertext. After an initial diagnosis, Bob may conclude that the condition of Alice is so complicated that a consultation by the respiratory physicians from all branches (not just branch 1) of hospital A is needed. In this

scenario, Alice needs to update the ciphertext and remove the attribute 'branch 1' from the policy such that all involved physicians are able to decrypt. In the multi-cloud storage model, organizations such as transaction platforms may collect tons of customer preference data and share the data among n cloud servers. Users of the data may contact $t+1$ out of the n servers, reconstruct the data, perform machine learning algorithms, and use the resulting model to make higher profits. The users need to pay for the services, as per the total amount of data downloaded from the $t+1$ servers. As the data may lose relevance over time and damage the model's accuracy [22], both the value and the privacy level of the data could be reasonably reduced over time. It is reasonable for the organization to gradually reduce the threshold t such that less servers are needed to reconstruct the data over time. In this scenario, the privacy threshold t may be reduced by gradually closing some of the servers and *relocating* the data (shares) on these servers to the remaining ones.

In both application scenarios, it suffices for the data owners to consider the problem of *how to remove an unauthorized subset of the attributes/servers but still enable the remaining attributes/servers in every authorized subset to recover the data*. In the language of SSS, a more formal description of the above problem is as follows:

(p1) *A secret s has been shared according to an access structure Γ over a set $\mathcal{P}$ of participants and later an unauthorized subset $Q \subseteq \mathcal{P}$ have to be removed. How to distribute the shares of Q to the participants in $\mathcal{P}\backslash Q$ such that for every authorized subset $A \in \Gamma$ the participants in $A \setminus Q$ are still able to reconstruct s.*

Via some abstraction, (p1) is related to the following problem:

(p2) *Given an SSS realizing an access structure Γ and an unauthorized subset Q of participants, construct a new SSS for $\Gamma_{.Q} = \{A \setminus Q : A \in \Gamma\}$.*

A solution to (p2) may provide a solution to (p1), if given the shares of Q, every participant in $\mathcal{P} \setminus Q$ can combine with its own share to produce a new share, such that the new shares realize $\Gamma_{.Q}$ over $\mathcal{P}\backslash Q$ for the secret s. In the literature, $\Gamma_{.Q}$ has been called the *contraction* of Γ at Q and the problem (p2) has been studied by [23], [24]. In particular, the ideas of [23], [24] can be extended to solve (p1), either by Q simply moving their shares to a public storage or every other participant. However, both solutions consume *additional storage*. In this paper, we are interested in solutions that require *no additional storage*.

A. Theoretical Contributions

Informally, an SSS for Γ is *ideal* if all of the shares are from the same domain as the secret [25]. If there is an ideal SSS realizing Γ, then Γ is ideal. In Section III, we propose a solution for (p2) under ideal access structures. More precisely, we provide two algorithms: the first one is applicable to $|Q| = 1$ and the second one is an extension of the first and is applicable to $|Q| > 1$. It is well-known that the shares in every LSSS can be generated by a matrix. Our algorithms are efficient and apply simple linear transformations on the matrix that generates an SSS for Γ to output a new SSS for $\Gamma_{.Q}$. While for (p1), the transformation gives a method for Q

to distribute their shares: send the shares to every remaining participant and each remaining participant can apply the same transformations on its shares and the shares of Q to obtain its shares in the new SSS. This will keep the size of each remaining participant's share unchanged. In particular, if we apply the proposed algorithm to an ideal LSSS for Γ, then an ideal LSSS for $\Gamma_{.Q}$ will be obtained.

B. Applications

Our algorithm have applications in both multi-cloud storage and CP-ABE based single-cloud storage.

Multi-cloud storage. In multi-cloud storage, the data owners split their valuable data into multiple shares and store shares on multiple independent cloud servers such that the users who have paid for the services are eligible to access the data. Many existing schemes for multi-cloud storage [8], [26]–[29] involve a considerable number of servers and the number may be as big as 100. Closing some of the servers is reasonable as the data is gradually devaluing over time and the data owners need to economize expenses on server rental. Our algorithms allow the data owners to *properly relocate* shares on some of the servers to the other servers such that the data is recoverable by downloading shares from less servers. In Section IV, we show our solution and compare it with three existing solutions. The comparisons show that our solution is most efficient in terms of cloud storage as it requires *no additional storage* on the remaining servers. Our only price is a low cost linear computation on the remaining servers.

Single-cloud storage. In the CP-ABE based single-cloud storage, some of the attributes may become unnecessary [30] and the ciphertext has to be updated such that the decrypting information associated with these attributes is *properly associated* with the remaining attributes, in order to change the access policy. A trivial method is downloading and decrypting the ciphertexts, and then re-encrypting the messages with $\Gamma_{.Q}$. Its computation and communication costs may be high. In Section V, we propose CP-ABE-CAS, a novel model of CP-ABE with contractions of access structures. In the proposed model, we introduce contraction keys, which are generated by the data owner itself, and a contraction algorithm such that: (1) the data owner is enabled to dominate the policy update, and (2) given the contraction key, the servers can efficiently update the ciphertexts to adapt to a new access policy as per the data owner' preferences while the users' private keys remain unchanged. We construct a CP-ABE-CAS scheme based on Waters' CP-ABE scheme [31]. Experimental results show that our scheme may reduce the server-side storage cost and the user-side computation/communication cost through efficient update of ciphertexts on server-side.

C. Related Work

1) Attribute-Based Encryption in Single-Cloud Storage:
Attribute-based encryption. Goyal et al. [32] classified ABE [18] into two types: key-policy ABE (KP-ABE) and ciphertext-policy ABE (CP-ABE). In CP-ABE, a user's private key is associated with a set of attributes and a ciphertext is associated with a policy.

Policy updating. In CP-ABE, policy updating [33] refers to the problem of changing the access policy associated with a ciphertext. In [33], the updated access policy is more restrictive than the original, so their construction *cannot* support the contraction studied by this work. Ciphertext-policy attribute-based proxy re-encryption [34]–[37] allows a semi-trusted proxy to perform policy updating. Their scheme requires a private key whose associated attribute set satisfies the policy to generate the re-encryption key and gives the data owner *no* control over the update of access policy. In our work, no private key is required and the data owner has *full* control over the update of policy.

Revocation. In CP-ABE, revocation [38] means revoking the access right of a user such that the user is no loner able to decrypt a ciphertext that he used to be able to decrypt. Revocation may happen when the services purchased by the users have expired. It is different from contraction because contraction removes some attributes from every authorized subset so that more users become eligible to decrypt. For example, Ge et al. [39] proposed a revocable ABE scheme with data integrity protection that adds more attributes to every authorized set. The new access structure is more restrictive and results in the revocation of some authorized users.

Extendable access control. An extendable access control system [40] allows a data owner to encrypt its data under an access structure Γ and later allows any user whose attribute set satisfies Γ to extend Γ to a new access structure Γ' such that any attribute set in $\Gamma \cup \Gamma'$ is able to access the data. When $\Gamma' = \Gamma_{\cdot Q}$ for some authorized subset Q, their scheme gives contraction. Compared with us, the data owner in [40] has *no* control over the extended access structure and the resulting ciphertext becomes *longer* than the original one. In our work, the data owner has *full* control on the contracted access structure and the resulting ciphertext is *shorter* than the original one. Lai et al. [41] proposed a scheme in a different setting of *identity-based* encryption.

2) Multi-Cloud Storage: The multi-cloud storage model [1], [6], [7], [42] has been very popular for ensuring data confidentiality in cloud environment. For example, Xiong et al. [42] considered a problem of unbalanced bandwidth between users and servers in a multi-cloud storage system and proposed *adaptive bandwidth* SSSs based on both Shamir's SSS [13] and Staircase codes. In this work, we consider a different problem of relocating data from some servers to the other servers. Our scheme can be used to improve the user's communication efficiency by *closing* the servers with excessively low bandwidth.

3) Secret Sharing:

Dynamic secret sharing. SSSs that allow dynamic changes to access structures have been designed in [43]–[46] and called *dynamic* SSSs. For general access structures, the scheme proposed by Cachin [43] allows the participants to be added or removed. In particular, when the participants in some unauthorized subset are removed, their shares will be published so that the scheme can realize contraction of the access structure at the subset and *occupy additional storage*. Our solution requires *no additional storage*. The schemes in [44], [45] did *not* consider

contractions of access structures when removing some participants. In fact, the contraction of an access structure Γ at a set Q can also be the union of $\Gamma_{\cdot Q}$ and Γ, so the access structure can also contract through adding $A \setminus Q$ (where $A \in \Gamma$) as new authorized subsets. The schemes in [44], [45] allow one to add new authorized subsets and are *computationally secure*. Our work uses *information-theoretic* SSSs. The schemes of [46] can only realize *threshold* access structures rather than general access structures. Our transformation is applicable to *any* LSSS, even if the access structure is not threshold.

Proactive secret sharing. There is a long line of research on SSS that enables participants to update their shares such that the information obtained by any adversary will be invalid. Such an SSS was introduced by Herzberg et al. [47] and called a *proactive* SSS. The schemes proposed in [47], [48] are only applicable to *static committees*. Later, *dynamic* proactive SSSs have been proposed in [49]–[52], both for *threshold* access structures [49], [50] and for general access structures [51], [52]. But if we focus on contractions of access structures, in [51], [52], *the remaining participants are required to interact with each other*. In contrast, there is *no interaction among the remaining participants* in our work.

Contraction. Closest to our work are [23], [24], [53]. Slinko [53] studied several ways to merge two ideal linear SSSs into a new ideal linear SSS but did *not* consider contractions of access structures. The work in [23], [24] solved the problem (p2). For the contraction of an access structure Γ at a set Q of the removed participants, if we assume that the share size of each participant is ℓ, the solutions in [23], [24] require the contracted system to consume *additional storage of at least* $|Q| \cdot \ell$ to store the shares of the removed participants. In our work, the proposed construction allows the remaining participants to combine the shares of Q with their shares to produce new shares so that *no additional storage* is needed.

D. Organization

Section II provides some basic definitions and notations. Section III solves the problem (p2). Section IV and Section V solve the problem (p1) in multi-cloud storage and single-cloud storage, respectively. Finally, Section VI contains our concluding remarks.

II. PRELIMINARIES

For any integer $n > 0$, we denote $[n] = \{1, \ldots, n\}$. For any vector $\boldsymbol{s} = (s_1, \ldots, s_n)$ and any set $I = \{i_1, \ldots, i_k\} \subseteq [n]$, we denote $\boldsymbol{s}_I = (s_{i_1}, \ldots, s_{i_k})$. In particular, we will write $\boldsymbol{s}_I = (s_i)_{i \in I}$. For any d-dimensional vector $\boldsymbol{t}$, we denote $t_j = \boldsymbol{t}_{\{j\}}$ for any $j \in [d]$. For any finite set $\mathcal{P}$, we denote by $2^{\mathcal{P}}$ the *power set* of $\mathcal{P}$, i.e., the set of all subsets of $\mathcal{P}$. Let $\psi : A \to B$ be a function. For any $b \in B$ (resp. any $C \subseteq B$), we denote $\psi^{-1}(b) = \{a \in A : \psi(a) = b\}$ (resp. $\psi^{-1}(C) = \{a \in A : \psi(a) \in C\}$).

Definition 1 (**Access Structure** [54])**.** *Let* $\mathcal{P} = \{P_1, \ldots, P_n\}$ *be a set of* n *participants. A collection* $\Gamma \subseteq 2^{\mathcal{P}}$ *is said to be monotone if it satisfies the property: For any* $A, B \in 2^{\mathcal{P}}$, *if* $A \in \Gamma$ *and* $A \subseteq B$, *then* $B \in \Gamma$. *A collection* $\Gamma \subseteq 2^{\mathcal{P}}$ *is said*

to be an access structure *over* $\mathcal{P}$ *if it consists of non-empty subsets of* $\mathcal{P}$ *and is monotone.*

Let Γ be an access structure over $\mathcal{P} = \{P_1, \ldots, P_n\}$. Every set in Γ is said to be an *authorized* subset of $\mathcal{P}$. Every set in $2^{\mathcal{P}} \setminus \Gamma$ is said to be *unauthorized*. An authorized subset $A \in \Gamma$ is *minimal* in Γ if no proper subset of A belongs to Γ. The *basis* of Γ consists of all minimal authorized subsets in Γ and denoted as Γ^{-}. The access structure Γ is said to be *connected* if each participant $P_i \in \mathcal{P}$ belongs to at least one minimal authorized subset in Γ^{-}.

Any access structure can be realized by a secret sharing scheme, which is essentially a distribution scheme with privacy properties.

Definition 2 (**Distribution Scheme** [54]). *Let* $\mathcal{P} = \{P_1, \ldots, P_n\}$ *be a set of* n *participants and let* S *be the domain of secrets. A* distribution scheme *for sharing the secrets in* S *among the participants in* $\mathcal{P}$ *is a pair* $\Pi = (\pi, \mu)$, *where* μ *is a probability distribution over a finite set* R *of random strings, and* $\pi : S \times R \to S_1 \times \cdots \times S_n$ *is a mapping* (S_i *is the domain of* shares *of* P_i *for every* $i \in [n]$).

With the scheme Π, a dealer can distribute a secret $s \in S$ by firstly choosing a random string $r \in R$ according to μ, computing a vector of shares $\pi(s, r) = (s_1, \ldots, s_n)$, and privately communicating each share s_i to P_i. For a set $A \subseteq \mathcal{P}$, we denote by $\pi_A(s, r) = (s_i)_{i \in I_A}$ the restriction of $\pi(s, r)$ to $I_A = \{i \in [n] : P_i \in A\}$. The efficiency of Π can be measured by its *information rate* $\rho(\Pi) = \log |S| / \max_{i \in [n]} \log |S_i|$.

Without loss of generality, we can always assume that μ is the uniform distribution over a properly chosen set R of random strings. When μ is the uniform distribution over R, we shall denote $\Pi = \pi$, instead of $\Pi = (\pi, \mu)$.

Definition 3 (**Secret Sharing Scheme (SSS)** [54]). *Let* $\mathcal{P} = \{P_1, \ldots, P_n\}$ *be a set of* n *participants. Let* Γ *be an access structure over* $\mathcal{P}$. *Let* $\pi : S \times R \to S_1 \times \cdots \times S_n$ *be a distribution scheme for* $\mathcal{P}$. *The scheme* π *is said to be a* secret sharing scheme *realizing* Γ *if the following requirements are satisfied:*

Correctness. *Any authorized subset of participants can reconstruct a secret with their shares of the secret. Formally, for every authorized subset* $A = \{P_{i_1}, \ldots, P_{i_m}\} \in \Gamma$, *there is a reconstruction function* $\mathsf{Recon}_A : S_{i_1} \times \cdots \times S_{i_m} \to S$ *such that for any* $s \in S$, $\Pr_r[\mathsf{Recon}_A(\pi_A(s, r)) = s] = 1$.

Perfect Privacy. *Any unauthorized subset of participants cannot learn any information about a secret from their shares of the secret. Formally, for every unauthorized subset* $A = \{P_{i_1}, \ldots, P_{i_m}\} \in 2^{\mathcal{P}} \setminus \Gamma$, *for any* $a, b \in S$ *and any* $\boldsymbol{s} = (s_{i_1}, \ldots, s_{i_m}) \in S_{i_1} \times \cdots \times S_{i_m}$, $\Pr_r[\pi_A(a, r) = \boldsymbol{s}] = \Pr_r[\pi_A(b, r) = \boldsymbol{s}]$.

Beimel [54] showed that for any SSS $\pi : S \times R \to S_1 \times \cdots \times S_n$ realizing a connected access structure Γ, it must be that $|S_i| \geq |S|$ for all $i \in [n]$. Thus, the information rate of an SSS for a connected access structure is always ≤ 1. An SSS with information rate 1 is said to be *ideal*. An access structure Γ is *ideal* if there is an ideal SSS realizing Γ.

Definition 4 (**Linear Secret Sharing Scheme (LSSS)** [31]). *Let* $\mathbb{F}$ *be a finite field. Let* $\mathcal{P} = \{P_1, \ldots, P_n\}$ *be a set of* n *participants. Let* Γ *be an access structure over* $\mathcal{P}$. *Let* $\pi : S \times R \to S_1 \cdots \times S_n$ *be an SSS realizing* Γ. *The scheme* π *is said to be* linear *over* $\mathbb{F}$ *if there exist a matrix* $\boldsymbol{H} = (\boldsymbol{h}_1, \ldots, \boldsymbol{h}_\ell)^{\top} \in \mathbb{F}^{\ell \times d}$, *a target vector* $\boldsymbol{t} = (1, 0, \ldots, 0)^{\top} \in \mathbb{F}^d$, *and a surjective function* $\psi : [\ell] \to \mathcal{P}$ *such that the share generation and secret reconstruction procedures are done as follows:*

Share Generation. *To share a secret* $s \in \mathbb{F}$, $d - 1$ *random field elements* $r_2, \ldots, r_d \in \mathbb{F}$ *are chosen to form a vector* $\boldsymbol{v} = (s, r_2, \ldots, r_d)^{\top}$. *For every* $i \in [n]$, *the participant* P_i's *share* $\boldsymbol{s}_i$ *is computed as* $\boldsymbol{s}_i = (\boldsymbol{h}_j^{\top} \boldsymbol{v})_{j \in \psi^{-1}(P_i)}$.

Reconstruction. *For any authorized subset* $A \in \Gamma$, *there exist constants* $\{\alpha_j : \psi(j) \in A\}$ *such that* $\boldsymbol{t} = \sum_{j \in \psi^{-1}(A)} \alpha_j \boldsymbol{h}_j$, *and thus* $s = \boldsymbol{t}^{\top} \boldsymbol{v} = \sum_{j \in \psi^{-1}(A)} \alpha_j (\boldsymbol{h}_j^{\top} \boldsymbol{v})$.

In Definition 4, the tuple $\mathcal{M} = (\mathbb{F}, \boldsymbol{H}, \boldsymbol{t}, \psi)$ specifies an LSSS for Γ and has been called a *monotone span program* (MSP) [55] for Γ. Beimel [54] showed that LSSSs and MSPs are equivalent: *every LSSS for* Γ *can be derived from an MSP for* Γ *and vice versa.*

In this paper, we are interested in the transformation from an LSSS for Γ to new LSSSs for contractions of Γ.

Definition 5 (**Contraction** [23]). *Let* $\mathcal{P} = \{P_1, \ldots, P_n\}$ *be a set of* n *participants. Let* Γ *be an access structure over* $\mathcal{P}$. *For any* $Q \subseteq \mathcal{P}$, *the* contraction *of* Γ *at* Q, *denoted as* $\Gamma_{\cdot Q}$, *is an access structure on* $\mathcal{P} \setminus Q$ *such that for every* $A \subseteq \mathcal{P} \setminus Q$, $A \in \Gamma_{\cdot Q} \Leftrightarrow A \cup Q \in \Gamma$. *We also say that* Γ *is contracted at* Q *to* $\Gamma_{\cdot Q}$.

If $Q \in \Gamma$, then $(\Gamma_{\cdot Q})^{-} = \{\{P_i\} | P_i \in \mathcal{P} \setminus Q\}$. If $Q \in 2^{\mathcal{P}} \setminus \Gamma$, then $(\Gamma_{\cdot Q})^{-}$ consists of all the minimal nonempty subsets of the form $A \cap (\mathcal{P} \setminus Q)$, where A is taken over Γ^{-}. For any two disjoint subsets $Q_1, Q_2 \subseteq \mathcal{P}$, $(\Gamma_{\cdot Q_1})_{\cdot Q_2} = \Gamma_{\cdot (Q_1 \cup Q_2)}$.

III. OUR TRANSFORMATIONS

In this section, we show how to transform an ideal LSSS for an access structure Γ to ideal LSSSs for contractions of Γ.

Let $\pi : S \times R \to S_1 \times S_2 \times \cdots \times S_n$ be an ideal LSSS for a *connected* access structure Γ on $\mathcal{P} = \{P_1, \ldots, P_n\}$. We suppose that π is equivalent to an MSP $\mathcal{M} = (\mathbb{F}, \boldsymbol{H}, \boldsymbol{t}, \psi)$, where $\mathbb{F}$ is a finite field, $\boldsymbol{H} = (\boldsymbol{h}_1, \ldots, \boldsymbol{h}_\ell)^{\top}$ is an $\ell \times d$ matrix over $\mathbb{F}$, $\boldsymbol{t} = (1, 0, \ldots, 0)^{\top} \in \mathbb{F}^d$ is a target vector, and $\psi : [\ell] \to \mathcal{P}$ is a surjective function that assigns the ℓ rows of $\boldsymbol{H}$ to the n participants in $\mathcal{P}$. Since π is ideal, we must have that $|S_i| = |S|$ for every $i \in [n]$, $\ell = n$ and $\psi : [n] \to \mathcal{P}$ is a bijection. Without loss of generality, we can suppose that $\psi(i) = P_i$ for every $i \in [n]$.

The following lemma shows that for any unauthorized subset of participants, if the rows assigned to them form a submatrix of $\boldsymbol{H}$ of rank r, then the last $d - 1$ columns of the submatrix must contain an invertible submatrix of order r. This $r \times r$ submatrix will be used in our transformations.

Lemma 1. *Let* $\mathcal{M} = (\mathbb{F}, \boldsymbol{H}, \boldsymbol{t}, \psi)$ *be an MSP that realizes a connected access structure* Γ *over* $\mathcal{P}$. *Let* Q *be any unauthorized subset and let* $\boldsymbol{H}_Q = \left((\boldsymbol{h}_i)_{\psi(i) \in Q}\right)^{\top}$. *If* $\mathrm{rank}(\boldsymbol{H}_Q) = r$,

then there exists a set $W = \{w_1, \ldots, w_r\} \subseteq \psi^{-1}(Q)$ and a set $K = \{k_1, \ldots, k_r\} \subseteq [d] \setminus \{1\}$ such that the order-r square matrix $\boldsymbol{U} = ((\boldsymbol{h}_{w_1})_K, \ldots, (\boldsymbol{h}_{w_r})_K)^\top$ is invertible over $\mathbb{F}$. (see Appendix A for the proof)

We will start with an algorithm (Algorithm 1) that takes $\mathcal{M}$ and an unauthorized subset $Q \subseteq \mathcal{P}$ with $|Q| = 1$ as input (w.l.o.g., $Q = \{P_n\}$) and outputs a new ideal LSSS $\mathcal{M}' = (\mathbb{F}, \boldsymbol{H}', \boldsymbol{t}, \psi')$ for $\Gamma_{\cdot Q}$, and then show an extended algorithm (Algorithm 2) for any unauthorized subset Q with $|Q| = m > 1$ (w.l.o.g., $Q = \{P_{n-m+1}, \ldots, P_n\}$).

Algorithm 1: Contraction at Q with $|Q| = 1$

Input: $\mathcal{M} = (\mathbb{F}, \boldsymbol{H}, \boldsymbol{t}, \psi)$, $Q = \{P_n\}$
Output: $\mathcal{M}' = (\mathbb{F}, \boldsymbol{H}', \boldsymbol{t}, \psi')$

1 Choose $k \in [d] \setminus \{1\}$ such that $h_{nk} \neq 0$;
2 **for** $i \in [n-1]$ **do**
3 $\quad \boldsymbol{h}'_i = \boldsymbol{h}_i - \frac{h_{ik}}{h_{nk}} \boldsymbol{h}_n$;
4 $\boldsymbol{H}' = (\boldsymbol{h}'_1, \ldots, \boldsymbol{h}'_{n-1})^\top$;
5 Define $\psi' : [n-1] \to \mathcal{P} \setminus Q$ such that $\psi'(i) = P_i$ for every $i \in [n-1]$;
6 **return** $\mathcal{M}' = (\mathbb{F}, \boldsymbol{H}', \boldsymbol{t}, \psi')$;

The step 1 of Algorithm 1 is always feasible, due to Lemma 1. It is also clear that the output $\mathcal{M}'$ of Algorithm 1 gives an ideal SSS. Below we show that $\mathcal{M}'$ realizes $\Gamma_{\cdot Q}$.

Theorem 1. *If $\mathcal{M}$ is an ideal LSSS realizing the access structure Γ over $\mathcal{P} = \{P_1, \ldots, P_n\}$, then for $Q = \{P_n\} \in 2^{\mathcal{P}} \setminus \Gamma$, the ideal LSSS $\mathcal{M}'$ output by Algorithm 1 realizes $\Gamma_{\cdot Q}$. (see Appendix B for the proof)*

For $Q = \{P_{n-m+1}, \ldots, P_n\}$, we can iteratively performing Algorithm 1 for all $\{P_j\} \subseteq Q$. However, there is a simpler one-step transformation (see Algorithm 2).

Algorithm 2: Contraction at Q with $|Q| > 1$

Input: $\mathcal{M} = (\mathbb{F}, \boldsymbol{H}, \boldsymbol{t}, \psi)$, $Q = \{P_{n-m+1}, \ldots, P_n\}$
Output: $\mathcal{M}' = (\mathbb{F}, \boldsymbol{H}', \boldsymbol{t}, \psi')$

1 Compute $r = \text{rank}((\boldsymbol{h}_{n-m+1}, \ldots, \boldsymbol{h}_n)^\top)$;
2 Find a set $W = \{w_1, \ldots, w_r\} \subseteq [n] \setminus [n-m]$ and a set $K = \{k_1, \ldots, k_r\} \subseteq [d] \setminus \{1\}$ such that the square matrix $\boldsymbol{U} = ((\boldsymbol{h}_{w_1})_K, \ldots, (\boldsymbol{h}_{w_r})_K)^\top$ is invertible;
3 **for** $i \in [n-m]$ **do**
4 $\quad$ Compute a new vector $\boldsymbol{h}'_i$ such that
$\quad \boldsymbol{h}'^\top_i = \boldsymbol{h}^\top_i - (\boldsymbol{h}^\top_i)_K \cdot \boldsymbol{U}^{-1} \cdot (\boldsymbol{h}_{w_1}, \ldots, \boldsymbol{h}_{w_r})^\top$;
5 $\boldsymbol{H}' = (\boldsymbol{h}'_1, \ldots, \boldsymbol{h}'_{n-m})^\top$;
6 Define $\psi' : [n-m] \to \mathcal{P} \setminus Q$ such that $\psi'(i) = P_i$ for every $i \in [n-m]$;
7 **return** $\mathcal{M}' = (\mathbb{F}, \boldsymbol{H}', \boldsymbol{t}, \psi')$;

Likewise, the step 2 of Algorithm 2 is always feasible due to Lemma 1 and the output $\mathcal{M}'$ of Algorithm 2 is ideal. Theorem 2 shows that $\mathcal{M}'$ exactly realizes $\Gamma_{\cdot Q}$.

Theorem 2. *If $\mathcal{M}$ is an ideal LSSS realizing the access structure Γ over $\mathcal{P} = \{P_1, \ldots, P_n\}$, then for $Q =$*

$\{P_{n-m+1}, \ldots, P_n\} \in 2^{\mathcal{P}} \setminus \Gamma$, *the ideal LSSS $\mathcal{M}'$ output by Algorithm 2 realizes $\Gamma_{\cdot Q}$. (see Appendix C for the proof)*

IV. APPLICATION IN MULTI-CLOUD STORAGE

In this section, we show an application of our transformation in multi-cloud storage. As stated in problem (p1), the scenario we will consider is as follows: A user has used an ideal LSSS π for an ideal access structure Γ to share a secret s as n shares $s_1, \ldots, s_n$ and stored the share s_i with a server P_i for every $i \in [n]$. Later the user may want to unsubscribe the servers in some unauthorized subset $Q \subseteq \{P_1, \ldots, P_n\}$. We need to figure out how to distribute the shares of Q to the servers in $\mathcal{P} \setminus Q$ such that for every authorized subset $A \in \Gamma$ the servers in $A \setminus Q$ are still able to reconstruct s. In particular, the dealer should be not involved in the process. In this section, we will discuss four possible solutions, the first one of which is based on our transformation from Section III, and show that ours is superior than the others. Furthermore, to understand clearly the four solutions, a toy example is given in Appendix D.

A. Solutions to the Storage Relocation Problem

1) Our Method: Referring to Algorithm 2, let $\mathcal{M}$ be the ideal LSSS π for Γ and let $Q = \{P_{n-m+1}, \ldots, P_n\}$. On input $(\mathcal{M}, Q)$, Algorithm 2 provides a solution to (p2) and outputs an ideal LSSS $\mathcal{M}' = (\mathbb{F}, \boldsymbol{H}', \boldsymbol{t}, \psi')$ for $\Gamma_{\cdot Q}$, where $\boldsymbol{H}' = (\boldsymbol{h}'_1, \ldots, \boldsymbol{h}'_{n-m})^\top$ and every $\boldsymbol{h}'_i$ is computed as

$$\boldsymbol{h}'^\top_i = \boldsymbol{h}^\top_i - (\boldsymbol{h}^\top_i)_K \cdot \boldsymbol{U}^{-1} \cdot (\boldsymbol{h}_{w_1}, \ldots, \boldsymbol{h}_{w_r})^\top \qquad (1)$$

at step 4 of Algorithm 2. If $\boldsymbol{v} = (s, r_2, \ldots, r_d)$ is used for computing the original shares $s_1, \ldots, s_n$ of all servers, i.e., $s_i = \boldsymbol{h}^\top_i \cdot \boldsymbol{v}$ for all $i \in [n]$, then $\{s'_i = (\boldsymbol{h}'_i)^\top \cdot \boldsymbol{v} : i \in [n-m]\}$ will be $n - m$ shares that realize the scheme $\mathcal{M}'$ for sharing s. In particular, as per (1), we have that

$$s'_i = s_i - (\boldsymbol{h}^\top_i)_K \cdot \boldsymbol{U}^{-1}(s_{w_1}, \ldots, s_{w_r})^\top \qquad (2)$$

is a linear combination of P_i's share s_i and Q's shares $s_{w_1}, \ldots, s_{w_r}$ with constant coefficients. Our method for (p1) simply *requires each server P_i to perform the computation of* (2) *and store the new share s'_i.*

Intuitively, our method for (p1) requires every remaining server to *linearly combine* its share with the shares of the removed servers. Thereby for any LSSS for access structure Γ and any unauthorized subset Q, we represent the new scheme for $\Gamma_{\cdot Q}$ obtained by using this method as π_{lc}.

In particular, a more intuitive solution for (p1) has been shown in Appendix E and is equivalent to our method. So its performance is the same as our method.

2) Martin's Method: In Martin [23], an SSS $\pi : S \times R \to S_1 \times S_2 \times \cdots \times S_n$ for Γ is represented as a matrix $\boldsymbol{M}$ that has $|S \times R|$ rows and n columns. Each row of the matrix is labeled with a pair $(s, r) \in S \times R$ and for every $i \in [n]$, the ith column of the matrix is labeled with P_i. For any $(s, r) \in S \times R$, and any $i \in [n]$, the entry of $\boldsymbol{M}$ in row (s, r) and column P_i is defined as the ith element of $\pi(s, r)$, i.e., P_i's share of $s \in S$ when $r \in R$ is used as the random string for sharing. Given $\boldsymbol{M}$ and Q, Martin [23] has a transformation from $\boldsymbol{M}$ to a SSS

M' for $\Gamma_{\cdot Q}$. For $Q = \{P_{n-m+1}, \ldots, P_n\}$, the transformation can be described as follows:

- *Choose* $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_m) \in S_{n-m+1} \times \cdots \times S_n$;
- *Define* $\boldsymbol{M}'$ *as the submatrix of* $\boldsymbol{M}$ *with rows labeled by* $\{(s,r) : (s,r) \in S \times R, \pi_Q(s,r) = \boldsymbol{\alpha}\}$ *and columns labeled by* $\mathcal{P} \setminus Q$.

The transformation as above provides a solution to (p2). To extend it to solve (p1), the key point is enabling the servers in $\mathcal{P} \setminus Q$ to have access to the shares of Q. A possible method is to *transfer Q's shares $\boldsymbol{\alpha}$ to a pubic storage* so that the servers in $\mathcal{P} \setminus Q$ can determine $\boldsymbol{M}'$. The new scheme from Martin's method remains *ideal* and is represented as π_{ps}.

3) Nikov-Nikova Method: Let $\mathcal{M} = (\mathbb{F}, \boldsymbol{H}, \boldsymbol{t}, \psi)$ be an MSP and an ideal LSSS for Γ. Let Q be an unauthorized subset. Nikov and Nikova [24] has a method of constructing an MSP $\mathcal{M}' = (\mathbb{F}, \boldsymbol{H}', \boldsymbol{t}, \psi')$ for $\Gamma_{\cdot Q}$ out of $(\mathcal{M}, Q)$:

- *Let $\boldsymbol{H}_{\psi^{-1}(Q)}$ be the rows assigned to Q in $\mathcal{M}$. The matrix $\boldsymbol{H}'$ is obtained by appending $n - m - 1$ copies of $\boldsymbol{H}_{\psi^{-1}(Q)}$ at the end of $\boldsymbol{H}$.*
- *The map ψ' is defined such that each server $P_i \in \mathcal{P} \setminus Q$ is assigned both one of the $n - m$ copies of $\boldsymbol{H}_{\psi^{-1}(Q)}$ and the rows $\boldsymbol{H}_{\psi^{-1}(P_i)}$.*

More precisely, this solution to (p2) requires each server in Q to make its share accessible to $\mathcal{P} \setminus Q$. For (p1), this idea simply *requires the servers in Q to transfer their shares to every server in $\mathcal{P} \setminus Q$*. Every server in $\mathcal{P} \setminus Q$ needs additional storage to *individually store a copy of Q's shares*. The new scheme $\mathcal{M}'$ turns out to be *non-ideal* and is represented as π_{is}.

4) Extended Nikov-Nikova Method: In Nikov-Nikova method, every server in $\mathcal{P} \setminus Q$ has to store a copy of Q's shares. When Q is removed, for every $A \in \Gamma$, to enable the servers in $A \setminus Q$ to reconstruct s, $A \setminus Q$ only need to *collectively store a copy of the shares of Q*. Thus the shares of Q may be properly hand out to $\mathcal{P} \setminus Q$ such that every server in $\mathcal{P} \setminus Q$ only needs to hold a part of Q's shares, and for every $A \in \Gamma$, $A \setminus Q$ can reassemble the shares of Q and then combine with their old shares to recover s. The new SSS is *non-ideal*. We represent the new scheme as π_{cs}.

B. Comparison

In this section, we restrict our attention to the threshold access structures and compare among four methods mentioned in Section IV-A, in terms of storage and information rate of the new scheme after contraction.

1) Theoretical Analysis: Let t, n be integers such that $1 \le t \le n$ and let $\mathcal{P} = \{P_1, \ldots, P_n\}$ be a set of n servers. Shamir's (t, n)-threshold secret sharing scheme (TSSS) [13] realizes a t-out-of-n threshold access structure $\Gamma = \{A \subseteq \mathcal{P} : |A| \ge t\}$. In Shamir's scheme π_0, a finite field $\mathbb{F}_p$ of prime order $p > n$ is chosen as the domain of secrets and the n servers $P_1, \ldots, P_n$ are associated with n distinct nonzero field elements $x_1, \ldots, x_n \in \mathbb{F}_p$, respectively. To share a secret $s \in \mathbb{F}_p$, the dealer chooses $t - 1$ field elements $a_1, \ldots, a_{t-1} \in \mathbb{F}_p$ randomly, defines a polynomial $P(x) = s + \sum_{j=1}^{t-1} a_j x^j$, and assigns a share $s_i = P(x_i)$ to the server P_i for all $i \in [n]$. Any $\ge t$ servers can reconstruct s

by interpolating the polynomial $P(x)$ with their shares. It has been well showed in [56] that Shamir's (t, n)-TSSS is ideal and linear. That is, for Shamir's (t, n)-TSSS π_0, there exists an MSP $\mathcal{M} = (\mathbb{F}_p, \boldsymbol{H}, \boldsymbol{t}, \psi)$ where $\boldsymbol{H} = (\boldsymbol{h}_1, \ldots, \boldsymbol{h}_n)^\top \in \mathbb{F}^{n \times t}$ such that $\boldsymbol{h}_i = (x_i^0, \ldots, x_i^{t-1})$, $\boldsymbol{t} = (1, 0, \ldots, 0)^\top \in \mathbb{F}_p^t$, and $\psi(i) = P_i$ for each $i \in [n]$. In particular, for any unauthorized subset Q of size m, $\Gamma_{\cdot Q}$ will be a $(t - m)$-out-of-$(n - m)$ threshold access structure. We shall apply our method (Section IV-A1), Martin's method (Section IV-A2), Nikov-Nikova method (Section IV-A3) and extended Nikov-Nikova method (Section IV-A4), respectively, to generate four new schemes $\pi_{\mathrm{lc}}, \pi_{\mathrm{ps}}, \pi_{\mathrm{is}}, \pi_{\mathrm{cs}}$ for $\Gamma_{\cdot Q}$. Let z be the number of secrets that have been shared using π_0, ℓ be the storage occupied by every share, and $L(\pi)$ be the total storage occupied by all shares in an SSS π.

TABLE I
THE TOTAL STORAGE (L) AND THE INFORMATION RATE (ρ)

π	$L(\pi)$	$\rho(\pi)$
π_{ps}	$n\ell z$	1
π_{is}	$(n + (n - m - 1)m)\ell z$	$(m+1)^{-1}$
π_{cs}	$(n + (n - t)m)\ell z$	$\lceil m(n - t + 1)/(n - m) + 1 \rceil^{-1}$
π_{lc}	$(n - m)\ell z$	1

In Table I, because Q is unauthorized, we have that $m \le t - 1$. It is not difficult to observe that when $m > 0$, $L(\pi_{\mathrm{lc}}) < L(\pi_{\mathrm{ps}}) < L(\pi_{\mathrm{cs}}) \le L(\pi_{\mathrm{is}})$ and $\rho(\pi_{\mathrm{lc}}) = \rho(\pi_{\mathrm{ps}}) > \rho(\pi_{\mathrm{cs}}) \ge \rho(\pi_{\mathrm{is}})$. That is, our method gives the most storage-efficient and the most communication-efficient (i.e., highest information rate) scheme for $\Gamma_{\cdot Q}$ among the four methods.

Remark. As a special case of HSS, a d-multiplicative secret sharing [57], [58] allows a user to share d secrets among the universal set $\mathcal{P}$ of n servers such that every server is able to locally convert its shares to a partial result and the sum of all server's partial results is equal to the product of the d secrets. In particular, Barkol [57] has showed that Shamir's (t, n)-TSSS is $\lfloor n/t \rfloor$-multiplicative. Thus, the four schemes $\pi_{\mathrm{lc}}, \pi_{\mathrm{ps}}, \pi_{\mathrm{is}}, \pi_{\mathrm{cs}}$ are $\lfloor (n - m - 1)/(t - m) \rfloor$-multiplicative. Since $\lfloor (n - m - 1)/(t - m) \rfloor \ge \lfloor n/t \rfloor$, the new schemes allow to homomorphically compute the product of more secrets.

2) Performance Analysis: To confirm the advantages of our method in terms of storage and information rate, a simple experiment is performed.

Like [27], [28], the users in our experiment are also allowed to subscribe at most 10 servers (i.e., $n = 10$) and we denote by $\mathcal{P} = \{P_1, \ldots, P_{10}\}$ the set of all servers. Let $p = 2^{16} - 15$ be a 16-bit prime. We set the threshold to be $t = 8$ and construct Shamir's $(8, 10)$-TSSS $M = (\mathbb{F}_p, \boldsymbol{H}, \boldsymbol{t}, \boldsymbol{\psi})$ for the threshold access structure Γ in the following way. We firstly choose $\boldsymbol{H}, \boldsymbol{t}$, and ψ such that $\boldsymbol{H} = (\boldsymbol{h}_1, \ldots, \boldsymbol{h}_{10})^\top$, where $\boldsymbol{h}_i = (i^0, \ldots, i^7)^\top$ for each $i \in [10]$, $\boldsymbol{t} = (1, 0, \ldots, 0)^\top \in \mathbb{F}_p^8$, and $\psi(i) = P_i$ for each $i \in [10]$. We randomly generate a list of 10^6 field elements as the secrets to be stored (i.e., the number of secrets is $z = 10^6$). Let $Q \subseteq \mathcal{P}$ be a set of m servers to be removed. The storage ℓ occupied by every share is 16 bit.

We have implemented the four methods given in Section IV-A on a Dell OptiPlex 7050 Personal Computer that runs

(a) Storage

(b) Information rate

Fig. 1. The total **storage** occupied by all shares in $\pi_{\text{ps}}, \pi_{\text{is}}, \pi_{\text{cs}}$, our π_{lc} respectively and the **information rate** of the four schemes, when m servers are removed from $(8, 10)$-threshold access structure. The storage occupied by every share is 16 bit ($\ell = 16$), and the number of secrets is $z = 10^6$

with an Intel Core i5-6500 (3.20GHz) processor and a RAM of 16 GB. We compare the four new schemes $\pi_{\text{ps}}, \pi_{\text{is}}, \pi_{\text{cs}}, \pi_{\text{lc}}$ for $\Gamma_{\cdot Q}$ in terms of the total storage L occupied by all shares (part (a) of Fig. 1) and the information rate (part (b) of Fig. 1) when the size m of Q increases from 0 to 7 with a step 1.

In part (a) of Fig. 1, when $n = 10, t = 8, \ell = 16, z = 10^6$, we have that $L(\pi_{\text{is}}) = -1.9073m^2 + 17.1661m + 19.0735$, $L(\pi_{\text{cs}}) = 3.8147m + 19.0735$, $L(\pi_{\text{ps}}) = 19.0735$, $L(\pi_{\text{lc}}) = -1.0973m + 19.0735$. Part (a) of Fig. 1 shows that our method occupies less storage than others, because our method can eliminate the storage occupied by Q's shares. The more servers are removed, the more storage-effective our method is than others. In part (b) of Fig. 1, when $n = 10, t = 8$, we have that $\rho(\pi_{\text{lc}}) = \rho(\pi_{\text{ps}}) = 1, \rho(\pi_{\text{cs}}) = \lceil 3m/(10 - m) + 1 \rceil^{-1}, \rho(\pi_{\text{is}}) = (m + 1)^{-1}$. Part (b) of Fig. 1 shows that the information rate of π_{lc} and π_{ps} is higher than that of π_{is} and π_{cs}.

V. APPLICATION IN SINGLE-CLOUD STORAGE

In an ABE scheme with an access structure Γ, the contraction of Γ means reduction in the attribute requirements for

Fig. 2. **CP-ABE (-CAS) Model**

decryption, so that more users will be allowed to access the encrypted data. In this section, we focus on contractions of access structures in CP-ABE schemes.

A. CP-ABE Model

A ciphertext-policy attribute-based encryption (CP-ABE) scheme $(\text{Setup}, \text{KeyGen}, \text{Encrypt}, \text{Decrypt})$ consists of four polynomial-time algorithms:

- $\text{Setup}(\lambda, U) \to (\text{PK}, \text{MSK})$. The *setup* algorithm takes a security parameter λ and a universal set U of attributes as input and outputs a public key PK and a master secret key MSK.
- $\text{KeyGen}(\text{PK}, \text{MSK}, A) \to \text{SK}$. The *key generation* algorithm takes the public key PK, the master secret key MSK, and a set A of attributes as input and outputs a private key SK for A.
- $\text{Encrypt}(\text{PK}, M, \Gamma, \mathcal{M}) \to \text{CT}$. The *encryption* algorithm takes the public key PK, a message M, an access structure Γ over U, and an LSSS $\mathcal{M}$ for Γ as input, and outputs a ciphertext CT such that a user can extract M from CT if and only if its attributes form an authorized subset in Γ. It is assumed that CT implicitly includes Γ.
- $\text{Decrypt}(\text{SK}, \text{CT}) \to M$. The *decryption* algorithm takes as input a private key SK for a set A of attributes and a ciphertext CT, which includes an access structure Γ. If $A \in \Gamma$, it outputs a message M.

System architecture. A CP-ABE scheme (depicted in Fig. 2) involves four entities: *authority, data owner, server,* and *user.* The authority is trusted and responsible to run Setup to generate (PK, MSK) and run KeyGen to generate a private key SK for every registered user. By running Encrypt, the *data owner* may use PK to encrypt its data M with an access policy Γ. The server is *honest-but-curious* and stores the resulting ciphertext CT. To learn M, the user simply downloads CT from the server and runs Decrypt.

Security. The security of CP-ABE schemes [31] can be defined with a security game G_1 between a challenger and an adversary and the game consists of the following phases:

- *Setup:* The challenger runs the setup algorithm to generate (PK, MSK) and gives PK to the adversary.
- *Phase 1:* The adversary queries the challenger for private keys corresponding to the attribute sets $S_1, \ldots, S_{q_1}$.

- *Challenge:* The adversary declares two equal length messages M_0, M_1 and an access structure Γ^* such that none of the queried attribute sets $S_1, \ldots, S_{q_1}$ satisfies Γ^*. The challenger chooses $b \in \{0, 1\}$ randomly, encrypts M_b under Γ^*, and gives the ciphertext CT^* the adversary.
- *Phase 2:* The adversary queries the challenger for private keys corresponding to the attribute sets $S_{q_1+1}, \ldots, S_q$, with the restriction that none of these satisfies Γ^*.
- *Guess:* The adversary outputs a guess b' for b.

The *advantage* of the adversary in G_1 is defined as $\Pr[b' = b] - 1/2$. A CP-ABE scheme is *secure* if all PPT adversaries have at most a negligible advantage in G_1.

B. Waters' CP-ABE Scheme

Waters [31] constructed a CP-ABE scheme that is secure under the decisional parallel bilinear diffie-hellman exponent assumption for bilinear groups (see Appendix F for definition of bilinear groups). Their scheme can be detailed as follows:

- Setup(λ, U). Choose a bilinear group $\mathbb{G} = \langle g \rangle$ of prime order p and a bilinear map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$. Choose random exponents $\beta, a \in \mathbb{Z}_p$. For every attribute $x \in U$, choose a random value $T_x \in \mathbb{G}$. Output

$$\mathrm{MSK} = g^\beta, \mathrm{PK} = \{g, g^a, e(g, g)^\beta, (T_x)_{x \in U}\}.$$

- KeyGen$(\mathrm{PK}, \mathrm{MSK}, A)$. Choose a random $t \in \mathbb{Z}_p$. Compute $K = g^\beta g^{at}, L = g^t, K_x = T_x^t$ for all $x \in A$. Output

$$\mathrm{SK} = \{A, K, L, (K_x)_{x \in A}\}.$$

- Encrypt$(\mathrm{PK}, M, \Gamma, \mathcal{M})$. Parse $\mathcal{M}$ as $(\mathbb{Z}_p, \boldsymbol{H}, \boldsymbol{t}, \psi)$, an LSSS for access structure Γ, where $\boldsymbol{H} = (\boldsymbol{h}_1, \ldots, \boldsymbol{h}_\ell)^\top$ is an $\ell \times d$ matrix over $\mathbb{Z}_p$, $\boldsymbol{t} = (1, 0, \ldots, 0)^\top \in \mathbb{Z}_p^d$ is a target vector, $\psi : [\ell] \to U$ is a map from each row $\boldsymbol{h}_i^\top$ to an attribute $\psi(i)$. Choose a random vector $\boldsymbol{v} = (s, v_2, \ldots, v_d) \in \mathbb{Z}_p^d$. For each $i \in [\ell]$, compute $s_i = \boldsymbol{h}_i^\top \boldsymbol{v}$ and chooses a random value $r_i \in \mathbb{Z}_p$. Let $C = Me(g, g)^{\beta s}, C' = g^s, C_i = g^{as_i} T_{\psi(i)}^{-r_i}, D_i = g^{r_i}$ for all $i \in [\ell]$. Finally, output

$$\mathrm{CT} = \{\mathcal{M}, C, C', (C_i, D_i)_{i \in [\ell]}\}.$$

- Decrypt$(\mathrm{SK}, \mathrm{CT})$. Suppose $A \in \Gamma$. Compute the constants $\{\alpha_i \in \mathbb{Z}_p : \psi(j) \in A\}$ such that $\sum_{\psi(j) \in A} \alpha_j \boldsymbol{h}_j^\top = \boldsymbol{t}$. Compute

$$\frac{e(C', K)}{\prod_{\psi(j) \in A} \left(e(C_j, L) e(D_j, K_{\psi(j)})\right)^{\alpha_j}} = e(g, g)^{\beta s}.$$

Finally, output $M = C / e(g, g)^{\beta s}$.

For every $i \in [\ell]$, the component C_i of the ciphertext CT is associated with an attribute $\psi(i)$ and provides necessary *decrypting information* to an authorized attribute set A that contains $\psi(i)$. As a result, the problem of eliminating the control of $\psi(i)$ (in general, an unauthorized subset of attributes) over decryption is reduced to the problem (p1).

C. CP-ABE with Contractions of Access Structure

To enable contractions of access structures, we extend the CP-ABE model of Section V-A to a new model of *CP-ABE with contractions of access structures (CP-ABE-CAS)*. The new model (Setup, KeyGen, Encrypt*, Decrypt, Contract) is obtained from that of CP-ABE by making two changes: (1) enhancing the Encrypt in CP-ABE to a new encryption algorithm Encrypt* that also outputs a contraction key CK; (2) adding a new contraction algorithm Contract that allows one to use CK (or its restrictions) to contract the access structure associated with a ciphertext. More precisely, the new algorithms Encrypt* and Contract can be detailed as below:

- Encrypt$^*(\mathrm{PK}, M, \Gamma, \mathcal{M}) \to (\mathrm{CT}, \mathrm{CK})$. The *modified encryption* algorithm additionally outputs a contraction key CK, which implicitly includes U. For any $Q \subseteq U$, CK can be restricted to CK_Q.
- Contract$(\mathrm{PK}, \mathrm{CT}, Q, \mathrm{CK}_Q) \to \mathrm{CT}_{.Q}$. The *contraction* algorithm takes the public key PK, a ciphertext CT with an access structure Γ, a set $Q \in 2^U \setminus \Gamma$, and a contraction key CK_Q restricted to Q as input and outputs a new ciphertext $\mathrm{CT}_{.Q}$ that can be decrypted by any authorized attribute set in $\Gamma_{.Q}$. Similarly, $\mathrm{CT}_{.Q}$ includes $\Gamma_{.Q}$.

System architecture. Referring to Fig. 2, to enable contractions of access structures with CP-ABE-CAS, a data owner may run Encrypt$^*(\mathrm{PK}, M, \Gamma, \mathcal{M})$ to produce $(\mathrm{CT}, \mathrm{CK})$, store CT on the cloud server and keep CK secret in local storage such that it is the only one that can later request the server to contract access structures. To remove an attribute y $(Q = \{y\} \in 2^U \setminus \Gamma)$, the data owner may send a restricted contraction key CK_Q to the server and let the server execute Contract$(\mathrm{PK}, \mathrm{CT}, Q, \mathrm{CK}_Q)$. Afterwards, all users with an authorized attribute set in $\Gamma_{.Q}$ will be able to decrypt the contracted ciphertext $\mathrm{CT}_{.Q}$.

Security. We define the security of CP-ABE-CAS with a security game G_2 that consists of the following phases:

- *Setup:* The challenger runs the setup algorithm to generate $(\mathrm{PK}, \mathrm{MSK})$ and gives PK to the adversary.
- *Phase 1:* The adversary queries the challenger for private keys corresponding to the attribute sets $S_1, \ldots, S_{q_1}$.
- *Challenge:* The adversary declares two equal length messages M_0, M_1, an access structure Γ^*, an LSSS $\mathcal{M}$ for Γ^*, and an unauthorized subset Q of Γ^*, where $\Gamma^*_{.Q}$ cannot be satisfied by any of $S_1, \ldots, S_{q_1}$. The challenger chooses $b \in \{0, 1\}$ randomly, encrypts M_b under Γ^*, producing CT^*, runs Contract$(\mathrm{PK}, \mathrm{CT}^*, Q, \mathrm{CK}_Q)$ to output $\mathrm{CT}^*_{.Q}$, and gives $(\mathrm{CT}^*, \mathrm{CT}^*_{.Q}, \mathrm{CK}_Q)$ to the adversary.
- *Phase 2:* The adversary queries the challenger for private keys corresponding to the attribute sets $S_{q_1+1}, \ldots, S_q$, with the restriction that none of these satisfies $\Gamma^*_{.Q}$.
- *Guess:* The adversary outputs a guess b' for b.

The *advantage* of an adversary in G_2 is $\Pr[b' = b] - 1/2$. A CP-ABE-CAS is *secure* if all PPT adversaries have at most a negligible advantage in G_2.

D. Our CP-ABE-CAS scheme

In this section, we upgrade the CP-ABE scheme of Waters [31] (see Section V-B) to a CP-ABE-CAS scheme with spe-

cific constructions of Encrypt* and Contract. Our contraction algorithm will be constructed based on the transformations from Section III. In particular, we will consider contractions of ideal access structures at an unauthorized attribute set Q of cardinality 1, because the algorithm can be easily extended to the case $|Q| > 1$. The details of Encrypt* and Contract are described as follows:

- Encrypt*$(\text{PK}, M, \Gamma, \mathcal{M})$. Execute the encryption algorithm Encrypt$(\text{PK}, M, \Gamma, \mathcal{M})$. Output the contraction key $\text{CK} = \{r_i : i \in [\ell]\}$. For any $Q \subseteq U$, let $\text{CK}_Q = \{r_i : i \in \psi^{-1}(Q)\}$ be the restricted contraction key.
- Contract$(\text{PK}, \text{CT}, Q, \text{CK}_Q)$. Suppose that $Q = \{y\} \in 2^U \setminus \Gamma$. W.l.o.g, assume that $\psi^{-1}(y) = \ell$. Invoke Algorithm 1 with $(\mathcal{M}, Q)$ as input to generate an MSP $\mathcal{M}'$ for $\Gamma_{.Q}$. Let k be the integer chosen at step 1 of Algorithm 1. For all $i \in [\ell - 1]$, compute

$$C_i' = C_i (C_{\psi^{-1}(y)} T_y^{r_{\psi^{-1}(y)}})^{-\frac{h_{ik}}{h_{\ell k}}}.$$

Finally, output $\text{CT}_{.Q} = \{\mathcal{M}', C, C', (C_i', D_i)_{i \in [\ell-1]}\}$.

Our contraction algorithm properly integrates the decrypting information associated with Q into that associated with every remaining attribute. As the new ciphertext $\text{CT}_{.Q}$ is *shorter* than the original ciphertext CT, hereafter we denote by Contract$_{\text{sct}}$ this contraction algorithm.

Correctness and security. The correctness of our scheme follows from that of Waters' scheme (Section V-B) and Theorem 1. The security of our scheme is an easy extension to that of Waters' scheme and appears in Appendix G.

Client-side storage. In our CP-ABE-CAS, the client needs to store a contraction key CK whose length is ℓ times that of a single message, where ℓ is the total number of attributes. When $|\text{CK}|$ occupies more storage than the outsourced data, storing data with a cloud server will become meaningless. This concern can be easily relieved by the data owner choosing a PRF $F_\gamma : \{0,1\}^* \to \mathbb{Z}_p$ and generating every element r_i in CK as $r_i = F_\gamma(i)$. The data owner only needs to keep the secret key γ of the PRF as a *long-term* contraction key.

Remark. If we apply Martin's method or Nikov-Nikova method from Section IV-A to construct CP-ABE-CAS, then intuitively Contract may realized by *appending the restricted contraction key to the original ciphertext*. That is, Contract simply outputs $\text{CT}_{.Q} = \{\mathcal{M}, C, C', (C_i, D_i)_{i \in [\ell]}, \text{CK}_Q\}$. Then an authorized attribute set in $\Gamma_{.Q}$ can leverage $\text{CK}_{.Q}$ to compute $e(C_j, L)e(D_j, K_{\psi(j)}) = e(g,g)^{as_j t} = e(C_j T_{\psi(j)}^{r_j}, L)$ for all $j \in \psi^{-1}(Q)$ and then recover the message M. This straightforward contraction algorithm via *ciphertext extension* is referred to as Contract$_{\text{ect}}$.

E. Performance Analysis

In this section, in order to compare the trivial solution (referred to as Contract$_{\text{re}}$) mentioned in Section I-B, Contract$_{\text{ect}}$ and Contract$_{\text{sct}}$ in Section V-D, we have implemented all algorithms on a Dell PowerEdge T640 Server that runs with an Intel Xeon Gold 5218 (2.30GHz) processor and a RAM of 16GB. We have used the Paring-Based Cryptography (PBC) library and the fast library for number theory (FLINT) to

Fig. 3. **Computation of contraction.** The execution time (resp. T_{re}, T_{ect}, and T_{sct}) of Contract$_{\text{re}}$, Contract$_{\text{ect}}$ and our Contract$_{\text{sct}}$ when 1 attribute is removed from $(8, n)$-threshold access structure (where $n = 10, 20, \ldots, 100$)

implement the algorithm. We use the type A elliptic curve $y^2 = x^3 + x$ and the order of the bilinear group is a 160-bit prime $p = 2^{159} + 16225927682921336339157801028129$.

We compare three solutions in terms of the following measures: during the contraction process, the communication and computation cost for the contraction operation; after contraction, the storage cost for the contracted ciphertext, the communication cost for downloading ciphertext, the computation cost for the user decryption. All the following experiments are for a *single* message. We refer to the parameter settings of [39]. For the contraction process, the main factor of the difference between the three solutions is the number n of attributes in the universe set U. We use the threshold access structure to encrypt the message, set the threshold to be $t = 8$ and set $n = |U|$ to be from 10 to 100 with a step 10. For the performance of the contracted ciphertext, the main factor of the difference between the three solutions is the number m of the attributes to be removed. We use a (t,n)-threshold structure where $t = 8, n = 10$ and choose the unauthorized subset Q of attributes to be removed such that $m = |Q|$ is from 0 to 7 with a step 1. Furthermore, to compute the execution time of the decryption for the original ciphertext and the contracted ciphertext, we simply allow the client to possess a set A of attributes such that $|A| = t$ and introduce a user whose attribute set is B such that $|B| = t - m$. To make the execution time more accurate, we execute each experiment 1000 times to compute an average time.

1) Evaluation of the Contraction Process:

Communication cost. In the trivial solution Contract$_{\text{re}}$, to realize the contraction of the access structure, provided that the client also has access to its data on the server and its attribute set is A, it should download and decrypt the ciphertext from the server and then upload new ciphertext after and re-encrypting. While in Contract$_{\text{sct}}$ and Contract$_{\text{ect}}$, it only requires the client to send the contraction key CK_Q. Theoretically, regardless of client-side or server-side communication, Contract$_{\text{sct}}$ and Contract$_{\text{ect}}$ are superior to Contract$_{\text{re}}$. This is confirmed by the following results. Fig. 4 shows the server-side and client-side communication of Contract$_{\text{re}}$ (SC$_{\text{re}}$ and

(a) Server-side communication

(b) Client-side communication

Fig. 4. **Communication of contraction.** The server-side (resp. SC_{re}, SC_{ect}, and SC_{sct}) and client-side communication (resp. CC_{re}, CC_{ect}, and CC_{sct}) of Contract$_{re}$, Contract$_{ect}$ and our Contract$_{sct}$ when 1 attribute is removed from $(8, n)$-threshold access structure (where $n = 10, 20, \ldots, 100$)

Fig. 5. **Storage of contracted ciphertext (implies communication of downloading contracted ciphertext).** The length (resp. l_{re}, l_{ect}, and l_{sct}) of the contracted ciphertext obtained by Contract$_{re}$, Contract$_{ect}$, and our Contract$_{sct}$ after m attributes are removed from $(8, 10)$-threshold access structure (where $m = 1, 2, \ldots, 7$)

CC_{re}), Contract$_{ect}$ (SC_{ect} and CC_{ect}) and Contract$_{sct}$ (SC_{sct} and CC_{sct}) when the threshold of access structure is $t = 8$ and the number of the attributes to be removed is $m = 1$. In particular, $CC_{re} = 0.1836n - 0.1445, CC_{ect} = CC_{sct} = 0.0195, SC_{re} = 0.2031n + 0.0391, SC_{ect} = SC_{sct} = 0$. From the results, it follows that whether on the client side or on the server side, the communication cost of Contract$_{sct}$ and Contract$_{ect}$ is much lower than Contract$_{re}$.

Computation cost. Here we compare the execution time (resp. T_{re}, T_{ect}, and T_{sct}) of three algorithms Contract$_{re}$, Contract$_{ect}$, and Contract$_{sct}$. Note that we ignore the time consumed by the communication between the server and the client, thus Contract$_{ect}$ takes negligible time in the contraction process. As shown in Fig. 3, when the threshold and the number of attributes to be removed are fixed, the execution time of Contract$_{re}$ and Contract$_{sct}$ is almost linear with the number n of attributes. In particular, when $t = 5, m = 1$, we can obtain by linear fitting that $T_{re} = 2.7802n + 9.8260, T_{sct} = 0.8964n + 0.5536, T_{ect} = 0$. Theoretically, as the number of attributes grows, more pairs $(C_i, D_i)_{i \in [|U| - 1]}$ in the encryption algorithm and more values $\{C_i' : i \in [|U| - 1]\}$ in the con-

traction algorithm need to be calculated. The results confirm it. Precisely, the execution time of Contract$_{re}$ is roughly 3 times that of Contract$_{sct}$. This shows that Contract$_{sct}$ is more efficient than Contract$_{re}$.

2) Evaluation of the Contracted Ciphertext:
Storage cost. We compare the length (resp. l_{re}, l_{ect}, and l_{sct}) of the contracted ciphertext obtained by three algorithms Contract$_{re}$, Contract$_{ect}$, and Contract$_{sct}$ in Fig. 5. The length l_{ect} of the contracted ciphertext generated by Contract$_{ect}$ is positively correlated with the size m of Q, while the length l_{sct} (and l_{re}) of the ciphertext generated by Contract$_{sct}$ (and l_{re}) is negatively correlated with m. In particular, when $n = 10, t = 8$, we have that $l_{ect} = 0.0195m + 2.0703, l_{re} = l_{sct} = 0.0195m^2 - 0.3984m + 2.0703$. Theoretically, as the size of Q grows, the length of the contraction key $CK._Q$ included in the contracted ciphertext generated by Contract$_{ect}$ increases, while in the contracted ciphertext generated by Contract$_{sct}$ (and Contract$_{re}$), the number of the pairs $(C_i', D_i)_{i \in [|U| - |Q|]}$, the size of the matrix $\boldsymbol{H}$ and the map ψ in the LSSS $\mathcal{M}$ all decrease. Thus, Contract$_{sct}$ (and Contract$_{re}$) is more cost-efficient than Contract$_{ect}$ in terms of storage.

Communication cost. During the user decryption process, the server sends the ciphertext to the user, and the server-side communication is exactly the length of the ciphertext, so the result is same as storage cost. Thus, Contract$_{sct}$ and Contract$_{re}$ are more efficient in terms of communication for the decryption of the contracted ciphertext than Contract$_{ect}$.

Computation cost. As shown in Fig. 6, when the universe set U of attributes is set such that $n = |U| = 10$ and the threshold is set to be $t = 8$, the execution time T_{ect}^{Dec} of the decryption for the contracted ciphertext obtained by Contract$_{ect}$ is positively correlated with the number m of attributes to be removed, while the execution time T_{sct}^{Dec} and T_{re}^{Dec} of the decryption for the contracted ciphertext obtained by Contract$_{sct}$ and Contract$_{re}$ are negatively correlated with the number m of attributes to be removed. In particular, when $n = 10, t = 8$, we can obtain by linear fitting that $T_{ect}^{Dec} =$

Fig. 6. **Computation of decrypting contracted ciphertext.** The execution time (resp. $T_{\text{re}}^{\text{Dec}}$, $T_{\text{ect}}^{\text{Dec}}$, and $T_{\text{sct}}^{\text{Dec}}$) of decryption for the contracted ciphertext obtained by $\text{Contract}_{\text{re}}$, $\text{Contract}_{\text{ect}}$ and $\text{Contract}_{\text{sct}}$ after m attributes are removed from $(8, 10)$-threshold access structure (where $m = 1, 2, \ldots, 7$)

$1.5540m + 9.9558, T_{\text{re}}^{\text{Dec}} = -1.2073m + 10.3447, T_{\text{sct}}^{\text{Dec}} = -1.1978m + 10.2759$. Thus, the computation cost of the user-side decryption for the contracted ciphertext obtained by $\text{Contract}_{\text{sct}}$ and $\text{Contract}_{\text{re}}$ is lower than the computation cost of the user-side decryption for the contracted ciphertext obtained by $\text{Contract}_{\text{ect}}$.

3) Advantages of Our Contraction Algorithm: Consider that in many practical scenarios the number of users may be quite large, our contraction algorithm $\text{Contract}_{\text{sct}}$ allows the server to update the ciphertext only once when removing attributes, then the contracted ciphertext may be downloaded and decrypted by a large number of users. For example, when the number of attributes in U is $n = 10$, the threshold is $t = 8$, the number of attributes to be removed is $m = 7$, the execution time of $\text{Contract}_{\text{sct}}$ is roughly 10 ms. For every user, the execution time $T_{\text{sct}}^{\text{Dec}}$ of the decryption for the contracted ciphertext obtained by $\text{Contract}_{\text{sct}}$ is roughly 2 ms, while the execution time $T_{\text{ect}}^{\text{Dec}}$ of the decryption for the contracted ciphertext obtained by $\text{Contract}_{\text{ect}}$ is roughly 20 ms. Note that the time of updating the ciphertext is only about 10 ms, so when the number of users is quite large, it is clear that our algorithm is superior to the algorithm with ciphertext extension in terms of chronic computation cost. Furthermore, the length of the contracted ciphertext obtained by $\text{Contract}_{\text{sct}}$ is less than that obtained by $\text{Contract}_{\text{ect}}$. This not only saves the server-side storage, but also saves the server-side communication cost for users to download the ciphertext. In conclusion, our algorithm takes a certain amount of time to update the ciphertext, but optimizes the server-side storage and the overall communication and the user-side computation when a large number of users request new ciphertext.

VI. CONCLUDING REMARKS

In this paper, we proposed algorithms that can efficiently transform a given LSSS for an access structure to LSSSs for contractions of the access structure. We also show their applications in solving the data relocating problem in multi-cloud storage and the attribute removal problem in the CP-ABE

based single-cloud storage. Our solutions are storage efficient and assume honest-but-curious cloud servers. It remains open to consider malicious servers and also ensure the integrity of the cloud data against malicious servers. In Appendix H, we briefly discuss several existing techniques that may be used to solve the data integrity problem.

REFERENCES

[1] V. Miranda-López, A. Tchernykh, M. G. Babenko, V.A. Kuchukov, M.A. Deryabin, E. Golimblevskaia, E. Shiryaev, A. Avetisyan, R. Rivera-Rodríguez, G. I. Radchenko, and E. Talbi, "Weighted two-levels secret sharing scheme for multi-clouds data storage with increased reliability," in *HPCS '19*: 915–922.

[2] J. Ning, Z. Cao, X. Dong, K. Liang, L. Wei and K.R. Choo, "Cryptcloud$^+$: Secure and expressive data access control for cloud storage," *IEEE Trans. Serv. Comput.*, 14(1): 111–124 (2021).

[3] M.A. Deryabin, N.I. Chervyakov, A. Tchernykh, M.G. Babenko, N.N. Kucherov, V. Miranda-López, and A. Avetisyan, "Secure verifiable secret short sharing scheme for multi-cloud storage," in *HPCS '18*: 700–706.

[4] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attribute-based encryption," in *S&P '07*: 321–334.

[5] Z. Ying, W. Jiang, X. Liu, S. Xu, and R. Deng, "Reliable policy updating under efficient policy hidden fine-grained access control framework for cloud data sharing," *IEEE Trans. Serv. Comput.*, 15(6): 3485–3498 (2021).

[6] B. Fabian, T. Ermakova, and P. Junghanns, "Collaborative and secure sharing of healthcare data in multi-clouds," *Inf. Syst.*, 48: 132–150 (2015).

[7] M. Li, C. Qin, P.P.C. Lee, and J. Li, "Convergent dispersal: Toward storage-efficient security in a cloud-of-clouds," in *HotCloud '14*.

[8] H. Zhang, J. Yu, C. Tian, P. Zhao, G. Xu, and J. Lin, "Cloud storage for electronic health records based on secret sharing with verifiable reconstruction outsourcing," *IEEE Access*: 40713–40722 (2018).

[9] E. Boyle, N. Gilboa, and Y. Ishai, "Breaking the circuit size barrier for secure computation under DDH," in *CRYPTO '16*, 9814: 509–539 (2016).

[10] L.F. Zhang and R. Safavi-Naini, "Protecting data privacy in publicly verifiable delegation of matrix and polynomial functions," *Des. Codes Cryptogr.*, 88(4): 677–709 (2020).

[11] L.F. Zhang and H. Wang, "Multi-server verifiable computation of low-degree polynomials," in *S&P '22*: 596–613.

[12] G. Ohtake, R. Safavi-Naini, and L. F. Zhang, "Outsourcing scheme of ABE encryption secure against malicious adversary," *Comput. Secur.*, 86: 437–452 (2019).

[13] A. Shamir, "How to share a secret," *Commun. ACM*, 1979.

[14] G.R. Blakley, "Safeguarding cryptographic keys," in *Managing Requirements Knowledge, International Workshop on*: 313–313 (1979).

[15] M. Ito, A. Saito, and T. Nishizeki, "Secret sharing scheme realizing general access structure," *Electronics and Communications in Japan (Part III: Fundamental Electronic Science)*, 72: 56–64 (1989).

[16] M. Ben-Or, S. Goldwasser, and A. Wigderson, "Completeness theorems for non-cryptographic fault-tolerant distributed computation," in *STOC '88*: 1–10.

[17] B. Chor, O. Goldreich, E. Kushilevitz, and M. Sudan, "Private information retrieval," in *FOCS '95*: 41–50.

[18] A. Sahai and B. Waters, "Fuzzy identity-based encryption," in *EURO-CRYPT '05*: 457–473.

[19] J. Wang, X. Yin, J. Ning, S. Xu, G. Xu, and X. Huang, "Secure updatable storage access control system for EHRs in the cloud," *IEEE Trans. Serv. Comput.*, 16(4): 2939–2953 (2023).

[20] T. Wang, Y. Zhou, H. Ma, and R. Zhang, "Flexible and controllable access policy update for encrypted data sharing in the cloud," *Comput. J.*, 66(6): 1507–1524 (2023).

[21] M. Yang, H. Wang, and Z. Wan, "PUL-ABE: An efficient and quantum-resistant CP-ABE with policy update in cloud storage," *IEEE Trans. Serv. Comput.*, Early Access (2023).

[22] E. Valavi, J. Hestness, N. Ardalani, and M. Iansiti, "Time and the value of data," *CoRR* abs/2203.09118 (2022).

[23] K. Martin, "New secret sharing schemes from old," *J. Comb. Math. Comb. Comput.*, 14: 65–77 (1993).

[24] V. Nikov and S. Nikova, "New monotone span programs from old," *IACR Cryptol. ePrint Arch.*, 2004: 282.

[25] E. F. Brickell, "Some ideal secret sharing schemes," in *EUROCRYPT '89*: 468–475.

[26] N. Wang, J. Fu, S. Zhang, Z. Zhang, J. Qiao, J. Liu, and B.K. Bhargava, "Secure and distributed IoT data storage in clouds based on secret sharing and collaborative blockchain," *IEEE/ACM Trans. Netw.*, 31(4): 1550–1565 (2023).

[27] T. Li, J. Chu, and L. Hu, "CIA: a collaborative integrity auditing scheme for cloud data with multi-replica on multi-cloud storage providers," *IEEE Trans. Parallel Distributed Syst.*, 34(1):154–162, (2023).

[28] P. Singh, N. Agarwal, and B. Raman, "Secure data deduplication using secret sharing schemes over cloud," *Future Gener. Comput. Syst.*, 88: 156–167 (2018).

[29] M. Hayashi and T. Koshiba, "Universal adaptive construction of verifiable secret sharing and its application to verifiable secure distributed data storage," *IEEE/ACM Trans. Netw.*, Early Access (2023).

[30] Y. Jiang, W. Susilo, Y. Mu, and F. Guo, "Ciphertext-policy attribute-based encryption supporting access policy update and its extension with preserved attributes," *Int. J. Inf. Sec.*, 2018.

[31] B. Waters, "Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization," in *PKC '11*: 53–70.

[32] V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-based encryption for fine-grained access control of encrypted data," in *CCS '06*: 89–98.

[33] A. Sahai, H. Seyalioglu, and B. Waters, "Dynamic credentials and ciphertext delegation for attribute-based encryption," in *CRYPTO '12*: 199–217.

[34] X. Liang, Z. Cao, H. Lin, and J. Shao, "Attribute based proxy re-encryption with delegating capabilities," in *AsiaCCS '09*: 276–286.

[35] S. Yu, C. Wang, K. Ren, and W. Lou, "Attribute based data sharing with attribute revocation," in *ASIACCS '10*: 261–270.

[36] K. Liang, L. Fang, W. Susilo, and D. S. Wong, "A ciphertext-policy attribute-based proxy re-encryption with chosen-ciphertext security," in *INCoS '13*: 552–559.

[37] K. Liang, M.H. Au, W. Susilo, D.S. Wong, G. Yang, and Y. Yu, "An adaptively cca-secure ciphertext-policy attribute-based proxy re-encryption for cloud data sharing," in *ISPEC '14*: 448–461.

[38] R.R. Al-Dahhan, Q. Shi, G.M. Lee, and K. Kifayat, "Survey on revocation in ciphertext-policy attribute-based encryption," *Sensors*, 19(7): 1695 (2019).

[39] C. Ge, W. Susilo, J. Baek, Z. Liu, J. Xia, and L. Fang, "Revocable attribute-based encryption with data integrity in clouds," *IEEE Trans. Dependable Secure Comput.*: 1–1 (2021).

[40] W. Susilo, P. Jiang, F. Guo, G. Yang, Y. Yu, and Y. Mu, "EACSIP: extendable access control system with integrity protection for enhancing collaboration in the cloud," *IEEE Trans. Inf. Forensics Secur.*, 12(12): 3110–3122 (2017).

[41] J. Lai, F. Guo, W. Susilo, X. Huang, P. Jiang, and F. Zhang, "Data access control in cloud computing: Flexible and receiver extendable," *IEEE Trans. Serv. Comput.*: 1–1 (2021).

[42] H. Xiong, C. Hu, Y. Li, G. Wang, and H. Zhou, "Secure secret sharing with adaptive bandwidth in distributed cloud storage systems," *IEEE Access*, 8: 108148–108157 (2020).

[43] C. Cachin, "On-line secret sharing," in *Cryptography and Coding, 5th IMA Conference*: 190–198 (1995).

[44] S. Ye, G. Yao, and Q. Guan, "A multiple secrets sharing scheme with general access structure," in *IUCE '09*: 461–464.

[45] J. Yuan and L. Li, "A fully dynamic secret sharing scheme," *Inf. Sci.*, 496: 42–52 (2019).

[46] M.H. Tadayon, H. Khanmohammadi, and M.S. Haghighi, "Dynamic and verifiable multi-secret sharing scheme based on hermite interpolation and bilinear maps," *IET Inf. Secur.*: 234–239 (2015).

[47] A. Herzberg, S. Jarecki, H. Krawczyk, and M. Yung, "Proactive secret sharing or: How to cope with perpetual leakage," in *CRYPTO '95*: 339–352.

[48] S. Mashhadi, "Secure publicly verifiable and proactive secret sharing schemes with general access structure," *Inf. Sci.*, 378: 99–108 (2017).

[49] D. A. Schultz, B. Liskov, and M.D. Liskov, "MPSS: mobile proactive secret sharing," *ACM Trans. Inf. Syst. Secur.*, 13(4): 34:1–34:32 (2010).

[50] S.K.D. Maram, F. Zhang, L. Wang, A. Low, Y. Zhang, A. Juels, and D. Song, "CHURP: dynamic-committee proactive secret sharing," *IACR Cryptol. ePrint Arch.*, 2019: 17.

[51] V. Nikov, S. Nikova, B. Preneel, and J. Vandewalle, "Applying general access structure to proactive secret sharing schemes," *IACR Cryptol. ePrint Arch.*, 2002: 141.

[52] V. Nikov, S. Nikova, B. Preneel, and J. Vandewalle, "On distributed key distribution centers and unconditionally secure proactive verifiable secret sharing schemes based on general access structure," in *INDOCRYPT '02*: 422–435.

[53] A. Slinko, "Ways to merge two secret sharing schemes," *IET Inf. Secur.*, 14(1): 146–150 (2020).

[54] A. Beimel, "Secret-sharing schemes: A survey," in *IWCC '11*: 11–46.

[55] M. Karchmer and A. Wigderson, "On span programs," in *SCT '93*.

[56] D. R. Stinson, "An explication of secret sharing schemes," *Des. Codes Cryptogr.*, 2(4): 357–390 (1992).

[57] O. Barkol, Y. Ishai, and E. Weinreb, "On d-multiplicative secret sharing," *J. Cryptol.*, 23(4): 580–593 (2010).

[58] G. Tsaloli, B. Liang, and A. Mitrokotsa, "Verifiable homomorphic secret sharing," in *ProvSec '18*: 40–55.

[59] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, and D. Song, "Provable data possession at untrusted stores," in *CCS '07*: 598–609.

[60] K. He, J. Chen, Q. Yuan, S. Ji, D. He, and R. Du, "Dynamic group-oriented provable data possession in the cloud," *IEEE Trans. Dependable Secur. Comput.*, 18(3): 1394-1408 (2021).

[61] H. Shacham and B. Waters, "Compact proofs of retrievability," in *ASIACRYPT '08*: 90–107.

[62] G. Anthoine, J. Dumas, M. Jonghe, A. Maignan, C. Pernet, M. Hanling, and D.S. Roche, "Dynamic proofs of retrievability with low server storage," in *USENIX Security '21*: 537–554.

[63] D. Catalano and D. Fiore, "Practical homomorphic MACs for arithmetic circuits," in *EUROCRYPT '13*: 336-–352.

[64] S. Feng, S. Xu, and L.F. Zhang, "Multi-key homomorphic MACs with efficient verification for quadratic arithmetic circuits," in *ASIACCS '22*: 17–27.

[65] L.F. Zhang and R. Safavi-Naini, "Generalized homomorphic MACs with efficient verification," in *AsiaPKC '14*: 3–12.

APPENDIX A
PROOF FOR LEMMA 1

As $\mathrm{rank}(\boldsymbol{H}_Q) = r$, there exist r rows of $\boldsymbol{H}_Q$ that are linear independent over $\mathbb{F}$. Suppose that these rows are $(\boldsymbol{h}_{w_1})^\top, \ldots, (\boldsymbol{h}_{w_r})^\top$, where $w_1, \ldots, w_r \in \psi^{-1}(Q)$. Below we show that there exists a set $K \subseteq [d] \setminus \{1\}$ such that the matrix $\boldsymbol{U}$ is invertible over $\mathbb{F}$. Assume for contradiction that $\boldsymbol{U}$ is not invertible for all $K \subseteq [d] \setminus \{1\}$. Then $((\boldsymbol{h}_{w_1})_{[d]\setminus\{1\}}, \ldots, (\boldsymbol{h}_{w_r})_{[d]\setminus\{1\}})^\top$ must be an $r \times (d-1)$ matrix of rank $< r$. Consequently, there exist r constants $\alpha_1, \ldots, \alpha_r \in \mathbb{F}$, which are not all 0, such that $\sum_{i\in[r]} \alpha_i (\boldsymbol{h}_{w_i})^\top = (\beta, 0, \ldots, 0)$. If $\beta \neq 0$, then $\{P_{\psi(w_1)}, \ldots, P_{\psi(w_r)}\}$ must be an authorized set of participants. Due to monotonicity, Q is authorized as well, which contradicts to the fact that Q is unauthorized. If $\beta = 0$, then the rows $(\boldsymbol{h}_{w_1})^\top, \ldots, (\boldsymbol{h}_{w_r})^\top$ must be linearly dependent over $\mathbb{F}$, which contradicts to our choice of the rows.

APPENDIX B
PROOF FOR THEOREM 1

Let Γ' be the access structure realized by $\mathcal{M}'$. It suffices to show that $\Gamma' = \Gamma_{\cdot Q}$.

Firstly, we show that $\Gamma' \subseteq \Gamma_{\cdot Q}$. For any $A \in \Gamma'$, there exist a set of constants $\{\alpha_i' : \psi(i) \in A\}$ such that $\boldsymbol{t} = \sum_{i\in\psi^{-1}(A)} \alpha_i' \boldsymbol{h}_i'$. Note that $\boldsymbol{h}_i' = \boldsymbol{h}_i - \frac{h_{ik}}{h_{nk}}\boldsymbol{h}_n$ for every $i \in [n-1]$. So we have that

$$\boldsymbol{t} = \sum_{i\in\psi^{-1}(A)} \alpha_i' \left(\boldsymbol{h}_i - \frac{h_{ik}}{h_{nk}}\boldsymbol{h}_n \right),$$

which shows that $\boldsymbol{t}$ is a linear combination of the vectors $\{\boldsymbol{h}_i : \psi(i) \in A \cup Q\}$. The set $A \cup Q$ must be authorized in the access structure Γ. Hence, $A \in \Gamma_{\cdot Q}$.

Secondly, we show that $\Gamma_{\cdot Q} \subseteq \Gamma'$. For any $A \in \Gamma_{\cdot Q}$, we have that $A \cup Q \in \Gamma$. Then there exist a set of constants $\{\alpha_i : \psi(i) \in A \cup Q\}$ such that

$$\boldsymbol{t} = \sum_{i\in\psi^{-1}(A\cup Q)} \alpha_i \boldsymbol{h}_i = \sum_{i\in\psi^{-1}(A)} \alpha_i \boldsymbol{h}_i + \alpha_n \boldsymbol{h}_n. \qquad (3)$$

As $Q \in 2^{\mathcal{P}} \setminus \Gamma$, there exists $k \in [d] \setminus \{1\}$ such that $h_{nk} \neq 0$. Due to Equation (3), we have that $0 = t_k = \sum_{i\in\psi^{-1}(A)} \alpha_i h_{ik} + \alpha_n h_{nk}$. By solving it in α_n, we have that

$$\alpha_n = - \sum_{i\in\psi^{-1}(A)} \alpha_i \frac{h_{ik}}{h_{nk}}. \qquad (4)$$

Equations (3) and (4) together imply that

$$\boldsymbol{t} = \sum_{i\in\psi^{-1}(A)} \alpha_i \left(\boldsymbol{h}_i - \frac{h_{ik}}{h_{nk}}\boldsymbol{h}_n \right) = \sum_{i\in\psi'^{-1}(A)} \alpha_i \boldsymbol{h}_i',$$

i.e., $\boldsymbol{t}^\top$ is a linear combination of the row vectors of $\boldsymbol{H}'$ labeled by $(\psi')^{-1}(A)$. Hence, $A \in \Gamma'$.

APPENDIX C
PROOF FOR THEOREM 2

Let Γ' be the access structure realized by $\mathcal{M}'$. It suffices to show that $\Gamma' = \Gamma_{\cdot Q}$.

Firstly, we show that $\Gamma' \subseteq \Gamma_{\cdot Q}$. For any $A \in \Gamma'$, there exist a set of constants $\{\alpha_i' : \psi(i) \in A\}$ such that

$$\boldsymbol{t}^\top = \sum_{i\in\psi^{-1}(A)} \alpha_i' \boldsymbol{h}_i'^\top. \qquad (5)$$

Note that $\boldsymbol{h}_i'^\top = \boldsymbol{h}_i^\top - (\boldsymbol{h}_i^\top)_K \cdot \boldsymbol{U}^{-1} \cdot (\boldsymbol{h}_{w_1}, \ldots, \boldsymbol{h}_{w_r})^\top$ for each $i \in [n-m]$. Equation (5) can be translated into

$$\boldsymbol{t}^\top = \sum_{i\in\psi^{-1}(A)} \alpha_i' \left(\boldsymbol{h}_i^\top - (\boldsymbol{h}_i^\top)_K \cdot \boldsymbol{U}^{-1} \cdot (\boldsymbol{h}_{w_1}, \ldots, \boldsymbol{h}_{w_r})^\top \right).$$

Note that for every $i \in \psi^{-1}(A)$, $(\boldsymbol{h}_i^\top)_K \cdot \boldsymbol{U}^{-1}$ is an r-dimensional row vector. It is not difficult to observe that $\sum_{i\in\psi^{-1}(A)} \alpha_i'(\boldsymbol{h}_i^\top)_K \cdot \boldsymbol{U}^{-1} \cdot (\boldsymbol{h}_{w_1}, \ldots, \boldsymbol{h}_{w_r})^\top$ is a linear combination of the vectors $\{\boldsymbol{h}_{w_1}^\top, \ldots, \boldsymbol{h}_{w_r}^\top\}$. Hence, $\boldsymbol{t}$ is a linear combination of the vectors $\{\boldsymbol{h}_i : \psi(i) \in A \cup \{P_{w_1}, \ldots, P_{w_r}\}\}$. The set $A \cup \{P_{w_1}, \ldots, P_{w_r}\}$ must be authorized in Γ. Due to monotonicity, we have that $A \cup Q \in \Gamma$ and thus $A \in \Gamma_{\cdot Q}$.

Secondly, we show that $\Gamma_{\cdot Q} \subseteq \Gamma'$. For any $A \in \Gamma_{\cdot Q}$, we have that $A \cup Q \in \Gamma$. Then there exist a set of constants $\{\alpha_i : \psi(i) \in A \cup Q\}$ such that

$$\boldsymbol{t}^\top = \sum_{i\in\psi^{-1}(A)} \alpha_i \boldsymbol{h}_i^\top + \sum_{j\in\psi^{-1}(Q)} \alpha_j \boldsymbol{h}_j^\top. \qquad (6)$$

Since the rank of the matrix $(\boldsymbol{h}_{n-m+1}, \ldots, \boldsymbol{h}_n)^\top$ is r and W labels a set of r linearly independent rows of the matrix, there must exist a set of constants $\{\alpha_w' : w \in W\}$ such that

$$\sum_{j\in\psi^{-1}(Q)} \alpha_j \boldsymbol{h}_j^\top = \sum_{w\in W} \alpha_w \boldsymbol{h}_w^\top. \qquad (7)$$

Due to Equations (6) and (7), we have that

$$
\begin{aligned}
& \sum_{i\in\psi^{-1}(A)} \alpha_i (\boldsymbol{h}_i^\top)_K + \sum_{j\in\psi^{-1}(Q)} \alpha_j (\boldsymbol{h}_j^\top)_K \\
= & \sum_{i\in\psi^{-1}(A)} \alpha_i (\boldsymbol{h}_i^\top)_K + \sum_{w\in W} \alpha_w \boldsymbol{h}_w^\top \\
= & \sum_{i\in\psi^{-1}(A)} \alpha_i (\boldsymbol{h}_i^\top)_K + (\alpha_w')_{w\in W} \cdot \boldsymbol{U} \\
= & \ (\boldsymbol{t}^\top)_K \\
= & \ \boldsymbol{0}.
\end{aligned}
\qquad (8)
$$

By solving the linear equation system (8), we have that

$$(\alpha_w')_{w\in W} = - \sum_{i\in\psi^{-1}(A)} \alpha_i (\boldsymbol{h}_i^\top)_K \cdot \boldsymbol{U}^{-1}. \qquad (9)$$

Equations (6), (7) and (9) together imply that

$$
\begin{aligned}
\boldsymbol{t}^\top &= \sum_{i \in \psi^{-1}(A)} \alpha_i \boldsymbol{h}_i^\top + \sum_{w \in W} \alpha_w' \boldsymbol{h}_w^\top \\
&= \sum_{i \in \psi^{-1}(A)} \alpha_i \boldsymbol{h}_i^\top + (\alpha_w')_{w \in W} \cdot (\boldsymbol{h}_{w_1}, \dots, \boldsymbol{h}_{w_r})^\top \\
&= \sum_{i \in \psi^{-1}(A)} \alpha_i \left(\boldsymbol{h}_i^\top - (\boldsymbol{h}_i^\top)_K \cdot \boldsymbol{U}^{-1} \cdot (\boldsymbol{h}_{w_1}, \dots, \boldsymbol{h}_{w_r})^\top \right) \\
&= \sum_{i \in (\psi')^{-1}(A)} \alpha_i \boldsymbol{h}_i'^\top.
\end{aligned}
$$

Thus, $\boldsymbol{t}^\top$ is a linear combination of the rows of $\boldsymbol{H}'$ labeled by $(\psi')^{-1}(A)$. Therefore, $A \in \Gamma'$.

APPENDIX D
A TOY EXAMPLE OF STORAGE RELOCATION

Let $\mathcal{P} = \{P_1, P_2, P_3, P_4\}$ be a set of four servers. Consider an access structure Γ over $\mathcal{P}$ such that $\Gamma^- = \{\{P_1, P_2, P_4\}, \{P_1, P_3, P_4\}\}$. Let $\mathcal{M} = (\mathbb{F}_2, \boldsymbol{H}, \boldsymbol{t}, \psi)$ be an MSP and an ideal LSSS for Γ, where

$$
\boldsymbol{H} = (\boldsymbol{h}_1, \boldsymbol{h}_2, \boldsymbol{h}_3, \boldsymbol{h}_4)^\top = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 1 & 0 \end{bmatrix},
$$

$\boldsymbol{t} = (1, 0, 0)^\top$, and $\psi(i) = P_i$ for each $i \in [4]$. To share a secret $s \in \mathbb{F}_2$ with $\mathcal{M}$, the dealer chooses $r_2, r_3 \leftarrow \mathbb{F}_2$, defines $\boldsymbol{v} = (s, r_2, r_3)^\top$, and gives $s_i = \boldsymbol{h}_i^\top \boldsymbol{v}$ to P_i for all $i \in [4]$. Let $Q = \{P_4\}$ be an unauthorized subset. We show how to use the methods in Section IV to solve (p1).

Our method. By running Algorithm 1 on input $(\mathcal{M}, Q)$ with $k = 2$, we get an LSSS $\mathcal{M}' = (\mathbb{F}, \boldsymbol{H}', \boldsymbol{t}, \psi')$ for $\Gamma_{\cdot Q}$, where

$$
\boldsymbol{H}' = (\boldsymbol{h}_1', \boldsymbol{h}_2', \boldsymbol{h}_3')^\top = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix}
$$

and $\psi'(i) = P_i$ for all $i \in [3]$. Our method requires each server $P_i \in \mathcal{P} \setminus Q$ to compute and store a new share $s_i' = s_i - \frac{h_{i2}}{h_{42}} s_4$. Specifically, $s_1' = s_1$, $s_2' = s_2 - s_4$, and $s_3' = s_3 - s_4$.

Martin's method. To use Martin's method, the SSS $\mathcal{M}$ can be represented as the following matrix

$$
\boldsymbol{M} = \begin{array}{c} \\ (0,(0,0)) \\ (0,(0,1)) \\ (0,(1,0)) \\ (0,(1,1)) \\ (1,(0,0)) \\ (1,(0,1)) \\ (1,(1,0)) \\ (1,(1,1)) \end{array} \begin{array}{cccc} P_1 & P_2 & P_3 & P_4 \\ \left(\begin{array}{cccc} 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{array} \right) \end{array},
$$

whose rows and columns are labeled by the elements of $\mathbb{F}_2 \times \mathbb{F}_2^2$ and $\mathcal{P}$, respectively. By choosing $\boldsymbol{\alpha} = (0)$, the SSS can be represented with

$$
\boldsymbol{M}' = \begin{array}{c} \\ (0,(0,0)) \\ (0,(0,1)) \\ (1,(0,0)) \\ (1,(0,1)) \end{array} \begin{array}{ccc} P_1 & P_2 & P_3 \\ \left(\begin{array}{ccc} 0 & 0 & 0 \\ 1 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 1 \end{array} \right) \end{array}.
$$

It suffices to keep the shares of P_1, P_2 and P_3 unchanged and send the share of $Q = \{P_4\}$ to a public storage.

Nikov-Nikova method. This method will construct an MSP $\mathcal{M}' = (\mathbb{F}_2, \boldsymbol{H}', \boldsymbol{t}, \psi')$ for $\Gamma_{\cdot Q}$, where

$$
\boldsymbol{H}' = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \end{bmatrix}
$$

and $(\psi')^{-1}(P_i) = \{i, i+3\}$ for every $i \in [3]$. It requires P_4 to transfer its share to every server in $\{P_1, P_2, P_3\}$ and requires each of these servers to additionally store s_4. The new scheme is *non-ideal*.

Extended Nikov-Nikova method. This method requires every authorized subset to keep a copy of the share s_4 of Q. As $(\Gamma_{\cdot Q})^- = \{\{P_1, P_2\}, \{P_1, P_3\}\}$, the method can be optimized by P_1 storing s_4.

APPENDIX E
A SOLUTION TO STORAGE RELOCATION

Here we give a more intuitive solution for the storage relocation problem, which is equivalent to our method in Section IV-A1.

Let $\mathcal{M} = (\mathbb{F}, \boldsymbol{H}, \boldsymbol{t}, \psi)$ be an MSP and an ideal LSSS for Γ, where $\boldsymbol{H} = (\boldsymbol{h}_1, \dots, \boldsymbol{h}_n)^\top \in \mathbb{F}^{n \times d}$, $\boldsymbol{t} = (1, 0, \dots, 0)^\top \in \mathbb{F}^d$, and $\psi(i) = P_i$ for every $i \in [n]$. For any unauthorized subset $Q \subseteq \mathcal{P}$, the shares $\{s_j\}_{P_j \in Q}$ leaves the secret s completely undetermined, i.e., from Q's view of point, any element of $\mathbb{F}$ could be s. In particular, there must exist a vector $\boldsymbol{v}' = (0, r_2', \dots, r_d')^\top \in \mathbb{F}^d$ such that $s_j = \boldsymbol{h}_j^\top \boldsymbol{v}'$ for all $P_j \in Q$. Such a vector $\boldsymbol{v}'$ may be not uniquely determined. Based on these observations, an alternative solution of (p1) can be described as follows:

- *Every server $P_j \in Q$ broadcasts its shares s_j.*
- *The servers in $\mathcal{P} \setminus Q$ agree on a vector $\boldsymbol{v}'$ such that $s_j = \boldsymbol{h}_j^\top \boldsymbol{v}'$ for all $P_j \in Q$.*
- *Every server $P_i \in \mathcal{P} \setminus Q$ replaces its share s_i with $s_i' = s_i - \boldsymbol{h}_i^\top \boldsymbol{v}'$.*

Note that there is a vector $\boldsymbol{v} = (s, r_2, \dots, r_d)$ such that $s_i = \boldsymbol{h}_i^\top \boldsymbol{v}$ for all $i \in [n]$. The correctness of this solution follows from the fact that $\{s_i'\}_{i \in [n]}$ is a set of valid shares of s, which are generated by choosing a random vector $\boldsymbol{v}'' = \boldsymbol{v} - \boldsymbol{v}' = (s, r_2 - r_2', \dots, r_d - r_d')$ and setting $s_i' = \boldsymbol{h}_i^\top \boldsymbol{v}''$, and the fact that $s_j' = 0$ for all $P_j \in Q$ and the shares of Q are actually not needed in any reconstruction.

Now we show that this method is equivalent to our method. Due to Lemma 1, if we let $\boldsymbol{H}_Q = \left((\boldsymbol{h}_i)_{\psi(i)\in Q}\right)^\top$ and assume that $\mathrm{rank}(\boldsymbol{H}_Q) = r$, then there exists a set $W = \{w_1,\ldots,w_r\} \subseteq \psi^{-1}(Q)$ and a set $K = \{k_1,\ldots,k_r\} \subseteq [d]\setminus\{1\}$ such that the order-r square matrix $\boldsymbol{U} = ((\boldsymbol{h}_{w_1})_K,\ldots,(\boldsymbol{h}_{w_r})_K)^\top$ is invertible over $\mathbb{F}$. To find a vector $\boldsymbol{v}' = (0, r'_2,\ldots,r'_d)^\top \in \mathbb{F}^d$ such that $s_j = \boldsymbol{h}_j^\top \boldsymbol{v}'$ for all $P_j \in Q$, it suffices to solve the equation $(\boldsymbol{h}_{w_1},\ldots,\boldsymbol{h}_{w_r})^\top \boldsymbol{v}' = (s_{w_1},\ldots,s_{w_r})^\top$ in $\boldsymbol{v}'$. It is not difficult to observe that there exists a vector $\boldsymbol{v}'$, where $r'_i = 0$ for each $i \in ([d]\setminus\{1\})\setminus K$ and $(r'_{k_1},\ldots,r'_{k_r})^\top = \boldsymbol{U}^{-1}(s_{w_1},\ldots,s_{w_r})^\top$, such that $s_j = \boldsymbol{h}_j^\top \boldsymbol{v}'$ for all $P_j \in Q$. Then for every server $P_i \in \mathcal{P}\setminus Q$,

$$
\begin{aligned}
s'_i = s_i - \boldsymbol{h}_i^\top \boldsymbol{v}' &= s_i - \sum_{k\in K} h_{ik} r'_k \\
&= s_i - (\boldsymbol{h}_i^\top)_K \cdot (r'_{k_1},\ldots,r'_{k_r})^\top \\
&= s_i - (\boldsymbol{h}_i^\top)_K \cdot \boldsymbol{U}^{-1}(s_{w_1},\ldots,s_{w_r})^\top.
\end{aligned}
$$

This shows that the two methods are equivalent. Although our method is less intuitive, it can be well applied to construct the CP-ABE-CAS, the above method cannot.

APPENDIX F
BILINEAR PAIRING

Definition 6 (Bilinear Pairing). *Let $\mathbb{G}$ and $\mathbb{G}_T$ be two multiplicative cyclic groups of prime order p. Let g be a generator of $\mathbb{G}$ and $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ be a bilinear map such that: (1) $e(u^a, v^b) = e(u,v)^{ab}$ for all $u, v \in \mathbb{G}$ and $a, b \in \mathbb{Z}_p$; (2) $e(g,g) \neq 1$. We say that $\mathbb{G}$ is a bilinear group if the group operations in $\mathbb{G}$ and $\mathbb{G}_T$ as well as the bilinear map e are efficiently computable.*

APPENDIX G
SECURITY OF OUR CP-ABE-CAS SCHEME

Theorem 3. *A CP-ABE-CAS is secure if Waters' CP-ABE scheme is secure.*

Proof. Assume that a CP-ABE-CAS is not fully secure. Then there exists an adversary $\mathcal{A}$ who has a non-negligible advantage in G_2. Next we show that an adversary $\mathcal{B}$ can win the game G_1 by means of the advantage of $\mathcal{A}$.

- *Setup:* The challenger runs the setup algorithm and gives the public parameter PK to $\mathcal{A}$ and $\mathcal{B}$.
- *Challenge:* $\mathcal{A}$ chooses two equal length messages M_0, M_1, an ideal access structure Γ^*, an LSSS $\mathcal{M}$ for Γ^*, and an unauthorized subset $Q = \{y\}$ of Γ^*, (w.l.o.g, assume that $\psi^{-1}(y) = \ell$) and then sends them to $\mathcal{B}$. $\mathcal{B}$ runs Algorithm 1 on input $(\mathcal{M}, Q)$, obtains the output $\mathcal{M}'$ for $\Gamma^*_{\cdot Q}$ and records k. Then $\mathcal{B}$ sends $M_0, M_1, \Gamma^*_{\cdot Q}, \mathcal{M}'$ to the challenger. The challenger randomly chooses $b \in \{0,1\}$, and encrypts M_b under $\Gamma^*_{\cdot Q}$, producing $\mathrm{CT}^*_{\cdot Q} = \{\mathcal{M}', C, C', (C_i, D_i)_{i\in[\ell-1]}\}$. It gives $\mathrm{CT}^*_{\cdot Q}$ to $\mathcal{B}$.
- *Adversary $\mathcal{B}$:* The adversary $\mathcal{B}$ chooses two random values r_ℓ, s_ℓ and creates

$$
\begin{aligned}
\mathrm{CK}_Q = r_\ell, \qquad C'_i &= C_i \cdot g^{as_\ell \cdot \frac{h_{ik}}{h_{\ell k}}}, \\
C'_\ell = g^{as_\ell}T_y^{-r_\ell}, \quad D_\ell &= g^{r_\ell}
\end{aligned}
$$

for each $i \in [\ell] \setminus \{\psi^{-1}(y)\}$. It then produces a new ciphertext $\mathrm{CT}^* = \{\mathcal{M}, C, C', (C'_i, D_i)_{i\in[\ell]}\}$. It gives $\mathrm{CT}^*, \mathrm{CT}^*_{\cdot Q}, \mathrm{CK}_Q$ to the adversary $\mathcal{B}$.
- *Adversary $\mathcal{A}$:* The adversary $\mathcal{A}$ returns a guess b'.
- *Guess:* The adversary $\mathcal{B}$ output b'.

Since $\mathcal{A}$ has a non-negligible advantage in G_2, $\mathcal{B}$ will also have a non-negligible advantage to win G_1. $\qquad\square$

Waters' CP-ABE scheme has been showed to be secure under the decisional q-parallel Bilinear Diffie-Hellman Exponent Assumption with $q \geq \ell, d$, where $\ell \times d$ is the size of the matrix $\boldsymbol{H}$ in the LSSS $\mathcal{M}$, so the CP-ABE-CAS is also secure under the same assumption according to Theorem 3.

APPENDIX H
ON THE DATA INTEGRITY

The data integrity is another critical security concern in cloud storage and requires that after the access structure contraction/policy update, the original data can still be recovered from the updated shares/ciphertexts.

Single-cloud storage. In single-cloud storage, the data integrity problem has been considered in [20], [39]. Ge et al. [39] proposed a revocable CP-ABE scheme that solves the data integrity problem by adding a commitment to Waters' scheme [31]. In our model, we assume the server is honest-but-curious and only the data owner is allowed to contract the policy. If we consider *malicious* servers/users, two threats may appear:

- A malicious server may tamper with the ciphertexts.
- A malicious user may impersonate the data owner and sends an arbitrary contraction key $\widehat{\mathrm{CK}}$ to the server.

Fortunately, both the above two threats can be relieved by applying our policy updating techniques from Section V to the scheme of [39].

Multi-cloud storage. In multi-cloud storage, provable data possession (PDP) [59], [60] and proofs of retrievability (PoR) [61], [62] have been invented for ensuring the data integrity. PDPs ensure data integrity through the validation of file block signatures. PoRs incorporate error-correcting codes to provide not only the data integrity but also the recoverability of corrupted data. In addition, techniques like homomorphic authenticators (HAs) [63]–[65] provide an alternative approach to solve the data integrity problem in multi-cloud storage through authenticating the shares on the servers.