Fuzzy Authorization for Cloud Storage

Shasha Zhu and Guang Gong, *Fellow, IEEE*

Abstract

By leveraging and modifying Ciphertext-Policy Attribute Based Encryption (CP-ABE) and OAuth, we propose a new authorization scheme, called *fuzzy authorization*, to facilitate an application registered with one cloud party to access data residing in another cloud party. The new proposed scheme enables the fuzziness of authorization to enhance the scalability and flexibility of file sharing by taking advantage of the one-to-one correspondence between Linear Secret-Sharing Scheme (LSSS) and generalized Reed Solomon (GRS) code. Furthermore, by conducting attribute distance checking and distance adjustment, operations like sending attribute sets and satisfying an access tree are eliminated. In addition, the automatic revocation is realized with update of $TimeSlot$ attribute when data owner modifies the data. The security of the fuzzy authorization is proved under the d-BDHE assumption. In order to measure and estimate the performance of our scheme, we have implemented the protocol flow of fuzzy authorization with OMNET++ 4.2.2 and realized the cryptographic part with Pairing-Based Cryptography (PBC) library. Experimental results show that fuzzy authorization can achieve fuzziness of authorization among heterogeneous clouds with security and efficiency.

Index Terms

Access control, attribute based encryption, ciphertext-policy, cloud storage, fuzzy authorization, privacy, security, generalized Reed-Solomon code.

I. Introduction

Advantages of cloud storage such as ease of accessibility, in-time syncing and less physical space consuming, etc., have motivated more and more people to adopt cloud storage services. In the meantime, cloud application services are boosting as well. As a result, the demand of inter-operations and authorizations between cloud storage service providers and cloud application service providers becomes more and more urgent. For example, a data owner stores several PDF files inside Justcloud, which is the top one cloud storage service provider [1]. Later on, data owner wants to merge some of the PDF files into one with the help of PDFMerge, an online cloud application service provider registered with Google Chrome Web Store [2]. The application PDFMerge needs to be authorized to access the pdf files residing in Justcloud, i.e., cloud storage provider; otherwise owner has to download the files from Justcloud and upload them to PDFMerge.

Because owner and the cloud applications are from different cloud domains, building trust between them is challenging. Another unwieldy issue is that more than one access token or secret key is needed if owner

Shasha Zhu is currently with UXP Systems Inc., Suite 1501, 5000 Yonge Street, Toronto, Ontario, Canada M2N 7E9 (e-mail: s29zhu@uwaterloo.ca), and this work was conducted when she was a graduate student in the Department of Electrical and Computer Engineering, University of Waterloo. Guang Gong is with the Department of Electrical and Computer Engineering, University of Waterloo, 200 University Avenue West, Waterloo, Ontario, Canada N2L 3G1 (e-mail: ggong@uwaterloo.ca).

wants to authorize access right of several files. Therefore, a scheme which can address the authorization among heterougeous clouds and reduce the number of access tokens and secret keys is required.

It is believed that OAuth [3] is the most widely-adopted authorization scheme. Unfortunately, it is infeasible to address the situation mentioned above. Because OAuth protocol requires both resource data and accessing application to be in the same domain. For example, pixlr.com, a web-application targeting on editing pictures online, registered with Google Chrome Web Store which can easily access data residing in Google Drive, but can hardly edit pictures from JustCloud. By introducing a trusted organization authority to maintain the integrity of cloud applications, AAuth, proposed by Tassanaviboon *et al.*, addressed a similar authorization situation in which owner and consumer are in different domains [4]. Unfortunately, the lack of scalability of authorization in AAuth does not facilitate the multiple authorizations.

In order to address the aforementioned issues, we propose fuzzy authorization (FA) for cloud storage which is an secure file-sharing scheme with high scalability and flexibility by leveraging and modifying Ciphertext-Policy Attribute Based Encryption (CP-ABE) [5] and OAuth. The term *fuzzy* means that this authorization scheme possesses attribute-discrepancy tolerance. In other words, a secret key associated with one attribute set can be applied to another attribute set through proper adjustment as long as the two attribute sets share certain amount of overlap.

The key features of our FA include: i) FA enables data owner to share their data with applications from a different cloud party. ii) By leveraging the natural transformation from Linear Secret-Sharing Scheme (LSSS) to generalized Reed Solomon (GRS) code [6] and inserting checking nodes into the access tree, FA enhances the scalability and flexibility of file-sharing. Moreover, through discrepancy detection and correction, FA avoids sending attributes to applications and eliminates performing satisfying an access tree procedure. iii) FA scheme revokes applications' right of accessing to a file automatically when the file is modified and re-encrypted by updating the secret share of $TimeSlot$ attribute. To summarize, the contributions of our work are as follows:

1) We propose a new secure authorization scheme for cloud storage providing file discrepancy tolerance, called fuzzy authorization.
2) The security analysis shows that our FA scheme provides a thorough security of outsourced data, including confidentiality, integrity and secure access control.

3) We have implemented the cryptographic part and simulated the protocol based on PBC library and OMNET++ 4.2.2, respectively. The simulation results demonstrate that FA reduces the storage consumption compared to other similar possible authorization schemes. It also asserts that our scheme could efficiently achieve distance tolerance and realize fuzzy authorization in practice.

The rest of the paper is organized as follows. The related works are discussed in Section II. In Section III, some basic concepts and definitions are introduced. In Section IV, we present the FA scheme. Detailed security analyses are given in Section V. In Section VI, we show the implementation of our scheme including environment, optimizations and experimental results. Section VII concludes the paper and addresses some future work.

II. Related Works

The wide adoption of cloud storage is raising several concerns about the data stored in cloud. Among which, confidentiality, integrity and access control of the data are the most significant and urgent issues [7, 8].

For the confidentiality of the outsourced data, Agudo *et al.* suggest several encryption schemes that can be adopted in cloud storage environment [9]. Xu *et al.* adopt the traditional AES encryption for their scheme and introduce an access policy on top of this encryption [10].

As to integrity, several researchers suggest to adopt a third party auditor (TPA) [11, 12, 13]. Wang *et al.* suggest a TPA leveraging the homomorphic linear authenticator to reduce the communication and computation overhead compared to the straightforward data auditing approaches [14]. Erway *et al.* present a definitional framework and efficient constructions for dynamic provable data possession (DPDP), which supports provable updates to stored data with a low slowdown in practice [15].

A series of new access control schemes and solutions have been investigated and devised for cloud environment based on the general access control solutions. Due to its scalability and security, Attribute-Based Encryption (ABE) [16] gains the most popularity in the schemes for access control. A distinguished work Fuzzy Identity-Based Encryption (IBE) [17] was introduced by Sahai and Waters in 2005. In a Fuzzy IBE scheme, a private key for an identity set ω can be used to decrypt a cipher-text encrypted with a slightly different identity set ω'. Fuzzy IBE realizes error tolerance by setting the threshold value of root node smaller than the size of identity set. Based on Fuzzy IBE, Goyal *et al.* present Keypolicy-Attribute

Based Encryption (KP-ABE) [16] and Bethencourt *et al.* introduce a complementary scheme to KP-ABE, called CP-ABE [5]. There are more concrete and general CP-ABE constructions in a later paper [18]. On the other hand, Boneh constructed BB1 and BB2 approaches [19] to build Identity-Based Encryption. Both CP-ABE and KP-ABE can be easily adapted to cloud environment, which has gained extensive researches along this line, say [4, 20, 21], just to list a few.

Tassanaviboon *et al.* propose an OAuth and ABE based authorization in semi-trusted cloud computing called AAuth [4]. Their authorization method enables an owner-to-consumer encryption and supports encrypted file sharing without revealing owner's secret key to consumers by introducing a third party authority. Based one ABE, Yu *et al.* introduce a way to enable the authority to revoke user attributes with minimal effort [22] and a method to achieve secure, scalable, and fine-grained data access control in cloud computing [23].

A cryptographic-based access control [20] for owner-write-user-read applications is introduced by Wang *et al.* in 2009. Their access control system encrypts every data block of cloud storage and adopts a key derivation method to reduce the number of keys. Yu also addresses fine-grained data access control, efficient key/user management, user accountability and etc., for cloud storage in his dissertation [21]. Moreover, a novel decentralized access control with anonymous authentication is introduced by Ruj *et al.* [24].

Different from the existing researches, we propose FA in this paper which not only maintains the confidentiality and integrity of the data, but also provides a scalable, efficient and flexible access control by modifying the general CP-ABE to adapt to the cloud storage environment. Through the integration of fuzzy functionality into the system, we enhance the scalability and flexibility of the secure authorization.

III. Preliminaries

In this section, we first review the asymmetric bilinear pairing. Then we present the decisional Binilear Diffie-Hellman Exponent assumption.

A. Bilinear Maps

Benefits such as broader choice of elliptic curve implementations and more compact representations of group elements make asymmetric bilinear pairing more favorable if the symmetry is not explicitly

required by the cryptographic scheme [19, 25]. Hence, we adopt an asymmetric bilinear pairing in our cryptographic scheme.

Denote $\mathbb{G}_1, \mathbb{G}_2$ and $\mathbb{G}_T$ the three multiplicative cyclic groups of prime order q. Define the generators of $\mathbb{G}_1$ and $\mathbb{G}_2$ as g_1 and g_2 respectively. Then the efficiently computable function bilinear pairing is $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$. Bilinear map e has the following properties:

1) Bilinearity: for all $u \in \mathbb{G}_1, v \in \mathbb{G}_2$ and $a,\ b \in \mathbb{Z}_q$, $e(u^a,\ v^b) = e(u,\ v)^{ab}$.

2) Non-degeneracy: $e(g_1,\ g_2) \neq 1$.

Tuple $(p,\ g_1,\ g_2,\ \mathbb{G}_1,\ \mathbb{G}_2,\ \mathbb{G}_T)$ is called asymmetric bilinear setting when $g_1 \neq g_2$ and $\mathbb{G}_1 \neq \mathbb{G}_2$. If $\mathbb{G}_1 = \mathbb{G}_2 = \mathbb{G}$ with g as the generator of $\mathbb{G}$, then tuple $(p,\ g,\ \mathbb{G},\ \mathbb{G}_T)$ is a symmetric bilinear setting.

B. Decisional Bilinear Diffie-Hellman Exponent Assumption

Waters proposes the decisional parallel Bilinear Diffie-Hellman Exponent assumption [18] and established the security of CP-ABE based on this assumption. Under the generalization of asymmetric pairings, we introduce decisional Bilinear Diffie-Hellman Exponent (d-BDHE) problem as follows.

Let $\zeta \in \mathbb{Z}_q$ be the target secret that an adversary intends to recover and K be the threshold value attached to the target node. We denote $\widetilde{W}$ an index set of secret shares. Let sets $\widetilde{W}'$ and $\widetilde{W}''$ where $|\widetilde{W}''| < K$ be two disjoint subsets of $\widetilde{W}$. Give tuple

$$\bar{y} = (g_1, g_2, g_2^{y_t}, g_1^{\mu_t y_t}, g_1^{\mu_{t'} r_{t'}},\ g_2^{r_{t'}}, g_1^{ra+\mu_{t''} r_{t''}},\ g_2^{r_{t''}}),\ \forall t \in \widetilde{W}, \forall t' \in \widetilde{W}', \forall t'' \in \widetilde{W}'' \tag{1}$$

with random numbers $r,\ a,\ s,\ y_t,\ \mu_t,\ \mu_{t'}$,$\mu_{t''}$,$r_t,\ r_{t'},\ r_{t''}$ chosen from $\mathbb{Z}_q$. If $\widetilde{W}, \widetilde{W}'$ or $\widetilde{W}''$ is empty, then the corresponding entries $\bar{y}$ with the indexes in $\widetilde{W}''$ will not be presented. For example, if $\widetilde{W}''$ is empty, the tuple will become

$$\bar{y} = (g_1, g_2, g_2^{y_t}, g_1^{\mu_t y_t}, g_1^{\mu_{t'} r_{t'}},\ g_2^{r_{t'}}),\ \forall t \in \widetilde{W}, \forall t' \in \widetilde{W}'. \tag{2}$$

To distinguish a random element $T \in \mathbb{G}_T$ from $e(g_1, g_2)^{ras}$ is referred to as the d-BDHE problem.

We say algorithm $\mathcal{B}$ outputting $z \in \{0, 1\}$ has advantage ϵ in solving d-BDHE in $(\mathbb{G}_1,\ \mathbb{G}_2)$ if

$$|Pr[\mathcal{B}(\bar{x}, e(g_1,\ g_2)^{ras}) = 0] - Pr[\mathcal{B}(\bar{y}, T) = 0]| \geq \epsilon \tag{3}$$

Definition 1: The decisional Bilinear Diffie-Hellman Exponent Assumption holds if no polynomial algorithm has a non-negligible advantage in solving the d-BDHE problem.

IV. Fuzzy Authorization Scheme for Cloud Storage

In this section, we present the FA scheme. First, we introduce the system model and adversary models. Next, we demonstrate the access tree structure and the transformation from LSSS to GRS code. At last, we present the main procedures and algorithms of FA.

A. System Model

1) Overview of the Protocol: There are four main entities in the system as shown in Fig. 1.

- Data owner: an entity who stores his/her data inside cloud storage and wishes to utilize cloud application services to process the data. A data owner must register with cloud storage provider and must be logged-in in order to upload, access data or authorize.
- Application service provider (ASP): an entity to be authorized to access cloud storage data. It is an application software resides in vendor's system or cloud and can be accessed by users through a web browser or a special purpose client software. For example, PDFMerge is an online tool which can be used to merge several pdf files into one pdf file. With proper authorization, PDFMerge fetches the source pdf files from cloud storage. As a result, uploading files from data owner's local device is avoided.
- Cloud storage provider (CSP): an entity which supplies storage as a service to its clients and also provides access application programming interfaces (APIs) to ASP when ASP holds a valid access token. Dropbox and JustCloud mentioned previously are examples of such entity.
- Application store (AS): an entity with which ASP must be registered to ensure itself's integrity and authenticity. Google Chrome Web Store is a typical application store.

Data owner encrypts his data with a random symmetric key KE and encrypts KE with our modified CP-ABE scheme. Owner encapsulates ciphertext of KE and ciphertext of data as an archive and stores the archive in the CSP. Format of the archive is similarly defined as AAuth archive [4]. When owner needs to share data with ASP, he/she and CSP join together to issue ASP the indirect secret shares of file attributes while AS and owner collaborate to issue the indirect secret shares of application attributes. In this paper, an indirect share contains a genuine secret share as its exponent or a part of its exponent. For example, when ζ_1 is a genuine secret share, g is a group element and r is random element, then $g^{\zeta_1 r}$ is

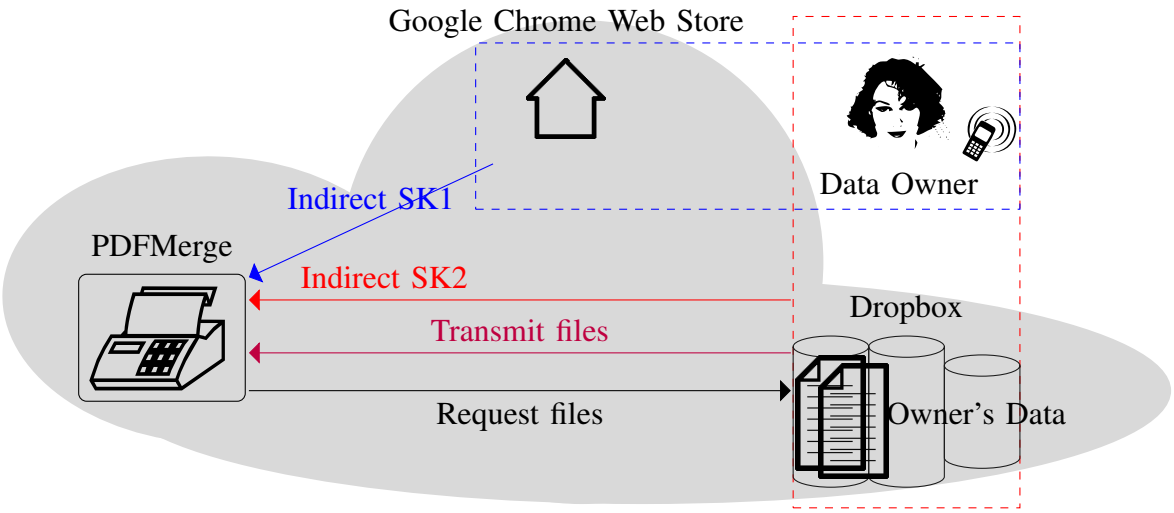


Fig. 1: System model.

an indirect secret share.

Since we emphasize the flexibility of multiple-file sharing, fuzziness is realized based on the file attributes. Once ASP gets all the indirect secret shares, it sends a request to CSP for formatted archive and then performs decryption of the archive header for KE. The main objective of this paper is to propose a secure and feasible way to address file-sharing issue with high scalability and flexibility in cloud storage. The way owner accesses the archive is not discussed here.

We assume that CSP, AS and ASP hold valid public-key certificates from Certificate Authorities and communications among the four parties are protected by SSL/TLS channels. We also assume that owner has both reading and writing permissions to cloud storage while ASP can be authorized with merely reading right.

2) Adversary Models: For system availability, it is natural to assume that every entity trusts the proposed protocol and execute the protocol honestly, although the entities do not trust each other. Despite we cannot ensure every entity not to exploit the threats to attack the system, we consider the following possible threats as adversary models.

(a) CSP is trusted to provide storage services properly but may intend to access owner's data illegally. CSP may take advantage of the indirect shares that it possesses and query the other indirect shares so as to reconstruct the top secret.

(b) ASP may try to decrypt the unauthorized files by utilizing the previous indirect shares issued to him.

ASP is allowed to query for the indirect shares that he/she does not possess.

(c) AS which is involved in issuing the indirect application secret shares may try to access owner's data in the name of ASP. Since it knows about partial indirect shares of application attributes, he/she may query about the indirect shares of file attributes and try to obtain the complete indirect shares of application attributes.

(d) An adversary owner may personate other owners to contribute the g_1^{ra} part for each attribute share.

(e) Targeting on the secret keys and access tokens, general network attacks might be launched by Internet hackers.

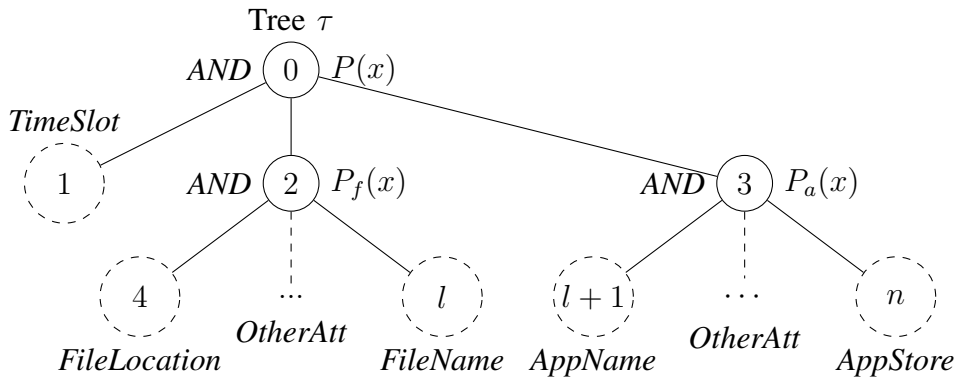
B. Access Tree Structure with Checking Nodes

Properly arranging access policy and inserting additional checking nodes at proper locations can achieve scalable and flexible authorization.

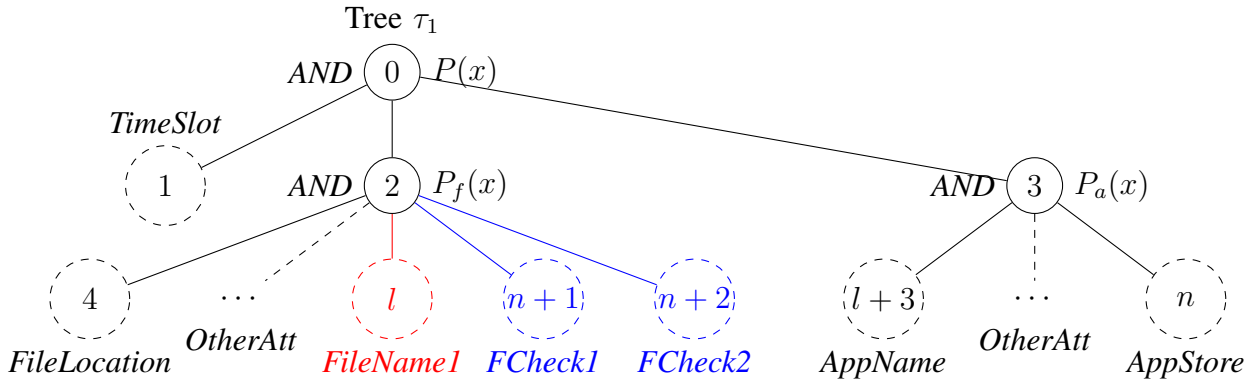
1) Construction of Access Tree with Checking Nodes: For all the archive files, the access tree structures are the same. But the polynomials for the root nodes of access trees are different. The symmetric key KE used to encrypt the plain file is encrypted under the access tree. Access trees are constructed with standard techniques [5] through ANDing operation. The subtree of file attributes, the subtree of application attributes and the $TimeSlot$ attribute are ANDed at the root node, as shown in Fig. 2(a). For simplicity, we call a subtree containing file attributes as F-subtree and a subtree containing application attributes as A-subtree. All file attributes, such as $FileName$, $FileLocation$, $FileType$, $FileOwner$, $FilePermission$, etc., are ANDed at the root node of the F-subtree. While the A-subtree contains attributes such as $AppStore$, $AppName$, $AppExpireDate$, $AppFunctionality$, $AppAuthor$, $AppAddress$ and so on. Each node in the tree is labeled with one index number. We use the index numbers to represent the nodes. A polynomial attached to F-subtree root node is denoted as $P_f(x)$ and $P_a(x)$ is the polynomial attached to the root node of A-subtree.

Compare two attribute sets in Fig. 2(b) and Fig. 2(c), only the attributes $FileName1$ and $FileName2$ are different. We say file1 and file2 have one-unit distance. Similarly, we say file1 and file2 have n-unit distance if there are n different attributes between their attribute sets.

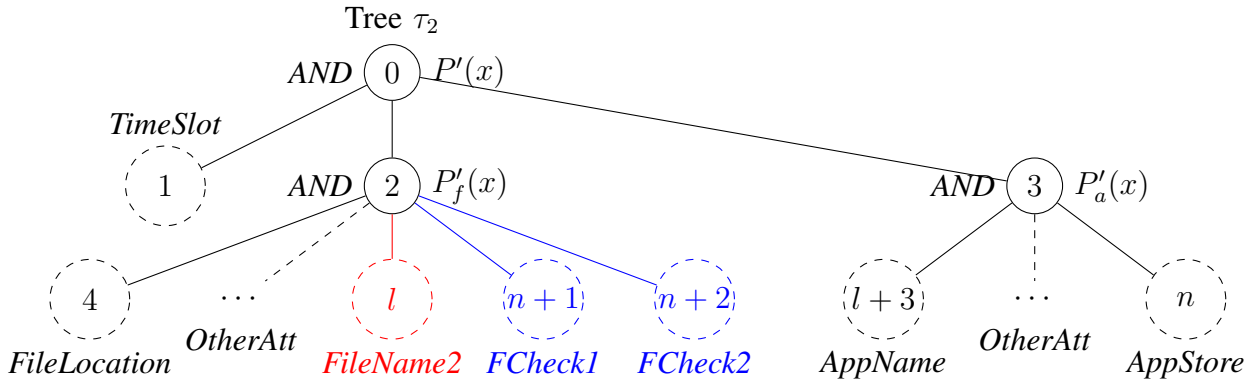
Before each authorization, owner can enable the checking nodes or disable them. If no redundant nodes are inserted, the issued secret key could only decrypt one single file without any security loss. However,



(a) Access tree τ without checking nodes.



(b) File1 access tree τ_1 with checking nodes in subtree of confined attributes.



(c) File2 access tree τ_2 with checking nodes in subtree of confined attributes.

Fig. 2: Access tree structure.

in many occasions, ASP needs to access more than one file. For example, PDFMerge needs to access several pdf files to perform merging. By inserting a designed number of redundant checking nodes into F-subtree, a token issued to ASP could be used to decrypt different archives. Fig. 2(b) and Fig. 2(c) are examples of adding two redundant nodes in the F-subtree which gives us one-unit distance tolerance. In Fig. 2(b) and Fig. 2(c), the values of additional nodes are computed by $P_f(n+1)$, $P_f(n+2)$, $P'_f(n+1)$

and $P'_f(n+2)$. The new cipher components of the additional nodes are computed and appended to the archives of file1 and file2 separately. So the token issued to decrypt file1 can be used to decrypt file2 and vice versa.

Similarly, owner could insert the additional nodes in the A-subtree to empower one token to be used by several applications. For simplicity, we only consider inserting redundant nodes in the F-subtree.

C. Transformation from Shamir's Linear Secret Sharing Scheme to GRS

We omit the details of original algorithms of GRS encoding and decoding and Shamir's LSSS. For convenience, we provide basic algorithms and notations of GRS and LSSS in Appendix. There is a one-to-one correspondence between Shamir's (K, N) secret sharing scheme and the GRS encoding and decoding algorithms. In other words, there is a transformation from secret shares distributing to GRS encoding and a transformation from secret recover to GRS decoding.

1) Transformation From Secret Distributing to GRS Encoding: By setting column multipliers vector $\boldsymbol{v}$ to $(1,\ 1,\ ...,\ 1)$ and code locator vector $\boldsymbol{\gamma} = (\gamma_0,\ \gamma_1,\ ...,\ \gamma_{N-1})$ to a vector with entries are indexes of the nodes, the process of GRS encoding is basicly the secret distributing procedure.

2) Transformation From Secret Recovery to GRS Decoding: As shown in equations (41) and (30), interpolation is the kernel of both secret recovery and GRS decoding. The difference is that GRS codeword has N coordinates, of which $N-K$ are redundant for error correction. So in order to leverage the error correction ability from GRS, we add some checking nodes into the tree as redundant nodes.

D. Main Procedures of Fuzzy Authorization

In lieu of using symmetric pairing which can only be constructed by some suitable supersingular elliptic curves, we adopt asymmetric pairing which allows a greater variety of known curves to be used. A Type 2 bilinear pairing [26] is adopted here. Recall that, $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ are cyclic groups of prime order q. Assume that Diffie-Hellman problem is hard in $\mathbb{G}_1$. Let $\phi:\ \mathbb{G}_2 \to \mathbb{G}_1$ be an efficient computable group isomorphism. Set $g_1 = \phi(g_2)$. A security parameter, k, determines the size of those three groups. An efficiently computable function is defined as $e:\ \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$. In addition, we are able to choose a hash function $H:\ (0,\ 1)^* \to \mathbb{G}_1$ which maps any binary string to a random element from $\mathbb{G}_1$ [27].

1) Setup(k): The setup algorithm, is first initiated by CSP who chooses a bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ of prime order q according to the input security parameter k. Generators of $\mathbb{G}_1$ and $\mathbb{G}_2$ are g_1 and g_2 respectively. CSP then chooses a random exponent β and publishes the public key as

$$CPK = \langle \mathbb{G}_1,\ \mathbb{G}_2,\ g_1,\ g_2,\ h = g_1^{\beta},\ f = g_2^{1/\beta} \rangle. \tag{4}$$

CSP's keeps $CSK = \langle \beta \rangle$ as its secret key.

Owner chooses a random exponent α and computes its public key and private key respectively as

$$OPK = \langle e(g_1,\ g_2)^{\alpha} \rangle,\ OSK = \langle g_2^{\alpha} \rangle. \tag{5}$$

2) EncryptandEncode(CPK, OPK, m, τ): Performed by owner, this algorithm encrypts a message KE under the access tree τ. Let $\mathcal{Y}$ be the overall leaf nodes set of τ and $P_y(x)$, $y \in \mathcal{Y}$ be the polynomial that assigned to each leaf node. The ciphertext CT is given by

$$CT = \langle \tau,\ \widetilde{C} = KE \cdot e(g_1,\ g_2)^{\alpha s},\ C = h^s, \forall y \in \mathcal{Y} : C_y = g_2^{P_y(0)},\ C'_y = H(att(y))^{P_y(0)} \rangle. \tag{6}$$

If 2η checking nodes are added, the corresponding cipher components of checking nodes are added to ciphertext as well.

3) KeyGen(CSK, OSK, ω): The algorithm requires CSP, owner, ASP and AS to collaborate together to issue access token and secret key without revealing their secret keys to each other. Taking secret keys of CSP and owner, together with a set of attributes ω as input, the procedure outputs a common part D and a set of indirect secret shares of secret key.

First, Owner and CSP work together to compute $D = g_2^{(\alpha + ra)/\beta}$ and g_1^{ra} in which $r,\ a \in \mathbb{Z}_q$ are chosen by CSP and owner separately. The sequence of interactions ensures that owner only knows g_2^{ra} and g_1^{ra} whereas CSP is merely aware of $g_2^{(\alpha+ra)/\beta}$ [4]. The common part D is sent by CSP to ASP.

Let ω' be the file-attribute set and ω'' be the application-attribute set, then the overall attribute set $\omega = \{TimeSlot\} \cup \omega' \cup \omega''$. After receiving the appointed file attribute set $\omega' \cup \{TimeSlot\}$ from owner, $\forall i \in \omega' \cup \{TimeSlot\}$, CSP randomly chooses $r_i \in \mathbb{Z}_q$ and computes $H(i)^{r_i}$ and $g_2^{r_i}$. Then owner computes $g_1^{ra} H(i)^{r_i}$ and sends them along with $g_2^{r_i}$ to ASP. ASP then authenticates itself to AS and presents the attributes of ω''. If authentication succeeds, $\forall j \in \omega''$, AS chooses $r_j \in \mathbb{Z}_q$ and compute $H(j)^{r_j}$ and $g_2^{r_j}$.

Again owner computes $g_1^{ra}H(j)^{r_j}$ and sends them along with $g_2^{r_j}$ to ASP. This algorithm ends up with ASP getting the SK which is represented as

$$SK = \langle D = g_2^{(\alpha+ra)/\beta},\ \forall t \in \omega : D_t = g_1^{ra}H(t)^{r_t},\ D'_t = g_2^{r_t}\rangle. \tag{7}$$

4) Delegate(SK, $\widetilde{\omega}$): The algorithm takes in a secret key SK with which an attribute set ω is embedded and another attribute set $\widetilde{\omega} \subset \omega$. Normally, this algorithm is used by an ASP. The algorithm first chooses a random value $\widetilde{r} \in \mathbb{Z}_q$ and $\forall l \in \widetilde{\omega}, \widetilde{r}_l \in \mathcal{Z}_q$ are randomly picked. After that, a new private key $\widetilde{SK}$ for an attribute set $\widetilde{\omega}$ is generated as

$$\widetilde{SK} = \{\widetilde{D} = Df^{\widetilde{r}}, \forall k \in \widetilde{\omega} : \widetilde{D}_k = D_k g_1^{\widetilde{r}a}H(k)^{\widetilde{r}_k}, \widetilde{D}'_k = D'_k g_2^{\widetilde{r}_k}\}. \tag{8}$$

5) Decrypt(CT, SK, τ): The decryption algorithm is a recursive procedure over the access structure τ, which is conducted by ASP.

Let $DecryptNode(CT, SK, x)$ denote the function that takes ciphertext CT, secret key SK and the node x in the tree as input. Let x be a leaf node, and i is the attribute attached to x,

$$DecryptNode(CT, SK, x) = \frac{e(D_i,\ C_x)}{e(C'_x,\ D'_i)} = \frac{e(g_1^{ra}H(i)^{r_i},\ g_2^{P_x(0)})}{e(H(i)^{P_x(0)},\ g_2^{r_i})} = e(g_1,\ g_2)^{raP_x(0)}. \tag{9}$$

When x is the root node of F-subtree or A-subtree, assume there are N child nodes of x. Let $\mathcal{Z} = \{z_0, z_1, \cdots, z_{N-1}\}$ denote the index set of x's children. For a child node z_i of x, algorithm calls $DecryptNode(CT, SK, z_i)$ and stores the result as

$$\boldsymbol{f_z} = (f_{z_0}, f_{z_1}, \cdots, f_{z_{N-1}}), f_{z_i} = e(g_1, g_2)^{raP_{z_i}(0)}. \tag{10}$$

If x is the the root node of the F-subtree, function $DistanceCheckandAdjust(CT, SK, \boldsymbol{f_z})$ will be called. Otherwise, the decryption result of node x is

$$e(g_1,\ g_2)^{raP_x(0)} = e(g_1,\ g_2)^{ra\sum_{t=0}^{N-1}\prod_{\forall i\in\mathcal{Z}, i\neq z_t}\frac{(0-i)f_{z_t}}{z_t-i}}. \tag{11}$$

The final decryption algorithm is based on $DecryptNode$ function and the distance adjustment procedure. First, the DecryptNode function is called on the root node, r, of the access tree τ. If the F-subtree decryption goes smoothly (i.e., successful), the decryption result of root node is presented as

$$A = DecryptNode(CT, SK, r) = e(g_1,\ g_2)^{ras}. \tag{12}$$

Then the encrypted KE can be computed as

$$KE = Decrypt(CT,\ SK) = \frac{\widetilde{C}}{e(C, D)/A} = \frac{KE \cdot e(g_1,\ g_2)^{\alpha s}}{e(g_1^{\beta s},\ g_2^{(\alpha+ra)/\beta})/e(g_1,\ g_2)^{ras}} \tag{13}$$

Otherwise, the decryption fails.

6) DistanceCheckandAdjust(CT, SK, $\boldsymbol{f_z}$): This is a two-step sub-procedure only applied to F-subtree.

(a) Distance Checking. Distance checking is enforced first. Based on the result of distance checking, distance adjustment may be performed.

According to the way we construct the access structure, the code locator vector is $\boldsymbol{\gamma} = (4,\ 5,\ ...,\ l,\ n+1,\ n+2,\ ...,\ n+2\eta)$ and column multiplier vector is $\boldsymbol{v} = (1,\ 1, ...,\ 1)$ with the length of $N = l - 3 + 2\eta$. Parity check matrix $\mathcal{H}$ can be easily obtained through equation (27) with $\boldsymbol{\gamma}$ and $\boldsymbol{v}$. The symbols of codeword, $P_{z_i}(0)$, are exponents of $e(g_1,\ g_2)$. Resembling computing syndromes in (28) and (29), the checking procedure is performed over the exponent of $e(g_1,\ g_2)$. The derived syndrome vector is obtained as

$$\begin{aligned} \boldsymbol{s'} &= (s'_0,\ s'_1,\ ...,\ s'_{N-1}) \\ &= (e(g_1,\ g_2)^{ra\sum_{j=0}^{N-1} P_{z_j}(0) v_j \gamma_j^0},\ e(g_1,\ g_2)^{ra\sum_{j=0}^{N-1} P_{z_j}(0) v_j \gamma_j^1},\ ...,\ e(g_1,\ g_2)^{ra\sum_{j=0}^{N-1} P_{z_j}(0) v_j \gamma_j^{N-1}}). \end{aligned} \tag{14}$$

An all-one vector $\boldsymbol{s'}$ indicates no distance, therefore no further adjustment is needed. In this case, the decryption procedure continues to interpolate $e(g_1,\ g_2)^{raP_f(0)}$ with $N - 2\eta$ child nodes similar as equation (11).

Otherwise, a not all-one vector of $\boldsymbol{s'}$ indicates that distance does exist and adjustment of distance must be performed.

(b) Distance Adjustment. Interpolation-based decoding and syndrome-based decoding are two well-known decoding types of GRS codes. For easy reference, Berlekamp-Welch algorithm [28], a typical interpolation-based decoding algorithm, and Peterson-Gorenstein-Zierler (PGZ) algorithm [29], a classical syndrome-based decoding procedure are presented in the appendix. Both algorithms are well known for their efficiency. Intuitively, one of these algorithms should be adopted to our scheme. Unfortunately, none of them, nor the other advanced decoding algorithms are applicable in our system based on our analysis and experiments. A detailed presentation of how these algorithms fail to decode in our system is shown in the appendix. So we employ the original decoding method [30, 31] which merely involves interpolation.

Given a leaf node set $\mathcal{Z}' = \{4, 5, ..., l, n{+}1, n{+}2, ..., n{+}2\eta\}$, construct the subset $\mathcal{Z}'_s = \{j_0, j_1, ..., j_{K-1}\}$, $\mathcal{Z}'_s \subset \mathcal{Z}'$ which gives us $\binom{N}{K}$ different $\mathcal{Z}'_s$. For each $\mathcal{Z}'_s$, do interpolation as

$$e(g_1,\ g_2)^{raP'_f(x)} = e(g_1,\ g_2)^{ra\sum\limits_{t=0}^{K-1}\prod\limits_{\forall i\in\mathcal{Z}'_s, i\neq j_t}\frac{(x-i)f_{j_t}}{j_t-i}}. \tag{15}$$

Thus $\binom{N}{K}$ results shall be obtained. From Reed and Solomon's Lemma [30], the occurring most often result shall the selected and used to obtain another vector

$$\begin{aligned}\boldsymbol{f'_z} =&(e(g_1,\ g_2)^{raP'_f(4)},\ e(g_1,\ g_2)^{raP'_f(5)},\ ...,\ e(g_1,\ g_2)^{raP'_f(l)},\\ &e(g_1,\ g_2)^{raP'_f(n+1)},\ e(g_1,\ g_2)^{raP'_f(n+2)},\ ...,\ e(g_1,\ g_2)^{raP'_f(n+2\eta)}).\end{aligned} \tag{16}$$

Let $d(\boldsymbol{f_z},\ \boldsymbol{f'_z})$ denote the distance between $\boldsymbol{f_z}$ and $\boldsymbol{f'_z}$ (i.e., the number of different terms of the two vectors). If $d(\boldsymbol{f_z},\ \boldsymbol{f'_z}) \leq \lfloor\frac{N-K}{2}\rfloor = \eta$, the adjustment is successful. We set the decryption on the root node of F-subtree as $e(g_1,\ g_2)^{raP_f(0)} = e(g_1,\ g_2)^{raP'_f(0)}$. Otherwise, distance adjustment fails and the whole decryption aborts.

7) Time slot synchronization: We divide time into intervals, not necessarily of the same length. In each time interval, polynomials attached to the access structures of archives are updated. The main idea of lazy re-encryption [32] is used in our system regarding to re-encrypt the sensitive data. When an application's access right is revoked, lazy revocation allows to postpone the update of polynomials and re-encrypt the sensitive data until the writing action has happened. Since only data owner has writing permission, time slot synchronization happens when owner updates the file.

In the beginning of each time slot, CSP and owner collaborate together to re-encrypt the header file. Let $\mathcal{M}$ denote the root node's children index set, and $|\mathcal{M}| = m$. Assume the time-slot attribute is attached to node i^*, $i^* = 1$ in Fig.2. Time-slot synchronization procedure initiates with owner chooses a random value $\widetilde{s}_t$. The most up-to-date time slot share is given as

$$P(i^*)_t = P_{TS}(0)_t = P_{TS}(0)_{t-1} + \widetilde{s}_t. \tag{17}$$

The new ciphertext components for a new time slot can be obtained as $C_{TS_t} = g_2^{P(i^*)_t}$ and $C'_{TS_t} = H(TimeSlot)^{P(i^*)_t}$. From equation (42), the top secret can be updated as

$$s_t = P(0)_t = \sum_{u=1}^{i^*-1}\prod_{\substack{\forall i\in\mathcal{M},\\ i\neq u}}\frac{(0-i)P(u)}{u-i} + \prod_{\substack{\forall i\in\mathcal{M},\\ i\neq i^*}}\frac{(0-i)P(i^*)_t}{i^*-i} + \sum_{u=i^*+1}^{m}\prod_{\substack{\forall i\in\mathcal{M},\\ i\neq u}}\frac{(0-i)P(u)}{u-i}. \tag{18}$$

So the discrepancy between s_t and s_{t-1} can be obtained based on equations (17) and (18)

$$\begin{aligned}\Delta_s =& s_t - s_{t-1} \\ =& \prod_{\substack{\forall i\in\mathcal{M},\\ i\neq i^*}} \frac{(0-i)P(i^*)_t}{i^*-i} - \prod_{\substack{\forall i\in\mathcal{M},\\ i\neq i^*}} \frac{(0-i)P(i^*)_{t-1}}{i^*-i} \\ =& (\prod_{\substack{\forall i\in\mathcal{M},\\ i\neq i^*}} \frac{0-i}{i^*-i})\widetilde{s_t}.\end{aligned} \tag{19}$$

Number i^* is constant, so as the most left part of Δ_s, $\prod_{\forall i\in\mathcal{M}, i\neq i^*} \frac{0-i}{i^*-i}$. Without any hard effort, the new ciphertext's main component $\widetilde{C}$ can be updated as $\widetilde{C} = KE \cdot e(g_1,\ g_2)^{\alpha(s+\Delta_s)}$ and $C = h^{(s+\Delta_s)}$.

V. Security Analysis and Some Comparisons

In this section, we first show the analysis of our system from perspectives of internal and external adversaries. Then we show that Fuzzy IBE [17] can be converted to a FA scheme and some comparisons with our scheme are demonstrated.

For internal adversaries, all entities in the system are considered to be semi-trusted. In the sense that they can exploit threats to subvert authorization control and data security, but still honestly follow the protocol. As for external adversaries, they may not run the protocol but try to launch general communication or network attacks to compromise the security of data. Here we give security analysis for internal adversary models, introduced in Section IV, provided that adversaries can get the cipher-text $CT = \langle \mathcal{T},\ \widetilde{C} = m \cdot e(g_1,\ g_2)^{\alpha s},\ C = h^s, \forall y \in \mathcal{Z} : C_y = g_2^{q_y(0)},\ C'_y = H(att(y))^{q_y(0)} \rangle$. According to our access structure, in order to recover the top secret s, the adverse party has to recover $e(g_1, g_2)^{raP_{TS}(0)}, e(g_1, g_2)^{raP_f(0)}$ and $e(g_1, g_2)^{raP_a(0)}$ in the first place.

In the following analysis, we use some reduction related methods to reduce the security of the FA to the d-BDHE assumption, defined in Section III. Since the d-BDHE assumption holds, the FA is secure.

We first specify the three sets $\widetilde{W}$, $\widetilde{W'}$ and $\widetilde{W''}$ which are introduced in Section III. Let $\widetilde{W}$ be the overall attribute set that adversary need to know in order to crack the target indirect secret, $\widetilde{W'}$, a set which adversary knows about $H(t')^{r_{t'}}$ and $g_2^{r_{t'}}$ for $\forall t' \in \widetilde{W'}$, and the $\widetilde{W''}$, a set which adversary knows about $g_2^{ra} H(t'')^{r_{t''}}$ for $\forall t'' \in \widetilde{W''}$.

A. Cloud Server Tries To Illegally Access or Modify Owner's Plain Data

Without any collusions with other parties, cloud server is able to get the $TimeSlot$ and file attributes, i.e., $\{TimeSlot\} \cup \omega'$. From the key generation in Section IV, we know that $\forall t \in \{TimeSlot\} \cup \omega'$, cloud server obtains $H(t)^{r_t}$ and $g_2^{r_t}$, but not $g_2^{ra} H(t)^{r_t}$. CSP has to compute $e(g_1, g_2)^{raP_{TS}(0)}$, $e(g_1, g_2)^{raP_f(0)}$ and $e(g_1, g_2)^{raP_a(0)}$ so as to compute the top secret $e(g_1, g_2)^{ras}$.

Scenario 1. Let the target indirect secret share be $e(g_1, g_2)^{raP_{TS}(0)}$ and the threshold value $K = 1$. In this case, $\widetilde{W} = \{TimeSlot\}$, $\widetilde{W'} = \widetilde{W}$ and $\widetilde{W''} = \emptyset$. Thus $|\widetilde{W''}| < K$. CSP knows about $\langle g_2^{y_t}, g_1^{\mu_t y_t} \rangle, t \in \widetilde{W}$ from ciphertext CT and public keys g_1, g_2. By definition, $\forall t' \in \widetilde{W'}$, $H(t')^{r_{t'}}$, i.e. $g_2^{\mu_{t'} r_{t'}}$, and $g_2^{r'_t}$ are known by CSP as well. To summarize, given tuple

$$\bar{y} = (g_1, g_2, g_2^{y_t}, g_1^{\mu_t y_t}, g_1^{\mu_{t'} r_{t'}},\ g_2^{r_{t'}}),\ \forall t \in \widetilde{W}, \forall t' \in \widetilde{W'}, \tag{20}$$

CSP has to differentiate $e(g_1, g_2)^{raP_{TS}(0)}$ from a random element $T \in \mathbb{G}_T$, i.e. the d-BDHE problem needs to be solved. In other words, the d-BDHE assumption holds in this scenario.

Scenario 2. Let the target indirect secret share be $e(g_1, g_2)^{raP_f(0)}$ and the threshold value of F-subtree's root node be a positive integer K, then we have $\widetilde{W} = \omega', \widetilde{W'} = \widetilde{W} = \omega'$ and $\widetilde{W''} = \emptyset$, $|\widetilde{W''}| < K$. Similar as Scenario 1, tuple, given by (20), is obtained by CSP so as to get the $e(g_1, g_2)^{raP_f(0)}$. The d-BDHE assumption still holds in this situation.

Scenario 3. Setting the target indirect secret share as $e(g_1, g_2)^{raP_a(0)}$ and the threshold value as a positive integer K. Sets $\widetilde{W} = \omega''$, and $\widetilde{W'} = \emptyset, \widetilde{W''} = \emptyset, |\widetilde{W''}| < K$. So given tuple

$$\bar{y} = (g_1, g_2, g_2^{y_t}, g_1^{\mu_t y_t}),\ \forall t \in \widetilde{W} \tag{21}$$

CSP has to guess $e(g_1, g_2)^{raP_a(0)}$. Thus the d-BDHE assumption holds.

Without $e(g_1, g_2)^{raP_{TS}(0)}$, $e(g_1, g_2)^{raP_f(0)}$ or $e(g_1, g_2)^{raP_a(0)}$, it is impossible for CSP to get $e(g_1, g_2)^{ras}$ and perform the final decryption.

B. ASP Tries to Decrypt Owner's Data without Permission

Two cases must be considered if ASP tries to access Owner's data illegally. The first case is that, an ASP is registered with an AS, but has never be requested by owner to fetch and handle owner's data. The

second is, an ASP registered with an AS and has been issued a token to access a certain file, but tries to access the file illegally after the revocation.

Scenario 4. In the first case, neither $TimeSlot$ nor the attributes of F-subtree is known by ASP. However, as to the application attribute set ω'', for any $t \in \omega''$, ASP could randomly choose r_t, and fabricates $\widetilde{D}_t = H(t)^{r_t}$ and $g_2^{r_t}$. Obviously, setting the target indirect secret share to $e(g_1, g_2)^{raP_{TS}(0)}$ or $e(g_1, g_2)^{raP_f(0)}$ leads us to Scenario 3 with set $\widetilde{W}$ being different. In either case, the d-BDHE assumption holds when ASP tries to get $e(g_1, g_2)^{raP_{TS}(0)}$ or $e(g_1, g_2)^{raP_f(0)}$.

If ASP sets the target indirect secret share to $e(g_1, g_2)^{raP_a(0)}$ and the threshold of A-subtree to a positive integer K. Then we have $\widetilde{W} = \omega', \widetilde{W}' = \widetilde{W}$ and $\widetilde{W}'' = \emptyset, |\widetilde{W}''| < K$. That is to say, with tuple in (20), ASP needs to differentiate the $e(g_1, g_2)^{raP_a(0)}$ from a random element $T \in \mathbb{G}_T$. Once again, the d-BDHE assumption holds.

Scenario 5. In the second case, we assume ASP keeps the $e(g_1, g_2)^{raP_f(0)}$ and $e(g_1, g_2)^{raP_a(0)}$ from previous authorization. Though ASP could forge the random exponent r_t for $TimeSlot$ attribute and get $H(TimeSlot)^{r_t}$, $g_2^{r_t}$, ASP has no idea of $g_1^{ra}H(TimeSlot)^{r_t}$. Let the target secret be $e(g_1, g_2)^{raP_{TS}(0)}$, then $\widetilde{W} = \{TimeSlot\}, \widetilde{W}' = \widetilde{W}$ and $\widetilde{W}'' = \emptyset, |\widetilde{W}''| < K$. That is, given tuple of (20), ASP has to guess $e(g_1, g_2)^{raP_{TS}(0)}$. The d-BDHE assumption holds as well. Since the root of the access tree is an AND node, without $e(g_1, g_2)^{raP_{TS}(0)}$, ASP could not get $e(g_1, g_2)^{ras}$.

C. AS Tries to Access Owner's Data Illegally

It is clear to find that attributes exposed to AS are application attributes and thus $\widetilde{W} = \omega''$. Similarly as we reduce the adversary ASP model to d-BDHE assumption, we can also reduce this adversary AS model to our d-BDHE assumption. Hence AS cannot get $e(g_1, g_2)^{ras}$.

D. Owner Propose Tokens to Access Other Owners' File

A malicious owner may either pretend to be an innocent owner to issue tokens or he/she may fabricate the tokens in place of another owner. The former case is unlikely as the malicious owner has to authenticate himself/herself to CSP. As to the latter case, the malicious owner may fabricate the partial components of indirect secret shares attached to file attributes and application attributes and multiply them with his own $g_1^{ra'}$. Alternatively, for any $t \in \omega$ (ω is the attribute set that is appointed by the innocent owner), an

owner may fabricate $H(t)^{r_t}$ and $g_2^{r_t}$ and combine them with $g_1^{ra'}$. Even in the best case, the malicious owner gets $e(g_1, g_2)^{ra's}$ and $e(g_1, g_2)^{(ra+\alpha)s}$. With $e(g_1, g_2)^{ra's}$ and $e(g_1, g_2)^{(ra+\alpha)s}$ to compute $e(g_1, g_2)^{\alpha s}$, the problem is reduced to a discrete logarithm problem and hence the fabrication is unsuccessful.

E. Comparisons with Fuzzy IBE Adapted in Our Scenario

In order to achieve flexibility of distance tolerance, Fuzzy IBE suggested two simple methods [17]. We adjust these two methods to fit in our occasion.

The first solution is referred to as Fuzzy IBE1. For each file, owner creates multiple access trees with distinct threshold values of F-subtrees. A smaller threshold value gives us larger distance-tolerant ability. Then owner encrypts the symmetric key KE under these different trees to obtain different ciphertexts. When conducting authorization, owner appoints one of the ciphertext to be sent to ASP. Consequently, a large piece of extra space and extra computation is required.

In the second solution, which is called Fuzzy IBE2, owner reserves some default attributes in the F-subtrees of all the files and keeps the threshold values unchanged. By increasing the number of default attributes, the ability of distance tolerance is enhanced. Since ASP is not aware of the file attributes, owner has to send ASP the file attributes together with the secret key. Before performing decryption with the secret key, ASP has to carry out satisfying an access tree procedure with the received file attributes to determine which attributes and corresponding components can be used for decryption.

Similar to the second solution, our FA scheme adds additional attributes into the F-subtree. Except that by adding two times of additional attributes into the subtree, FA has the ability to check and adjust the distance. Thus FA avoids owner from sending file attributes to ASP and eliminating the necessity of carrying out the satisfying an access tree procedure. We summarize the comparisons of FA and the other two solutions in Table I.

Note that in comparison with our FA scheme, both adapted fuzzy IBE schemes need to send attributes to ASP and must perform satisfying access tree procedure, which result in an extra computation as well as communication cost.

VI. IMPLEMENTATION

In this section, we first introduce the implementation environment and communication parameters among four entities. We then show some optimizations of the implementation. Finally, we provide the

TABLE I: Comparison among fuzzy authorization, Fuzzy IBE1, and Fuzzy IBE2.

Requirements	Fuzzy Auth	Fuzzy IBE1	Fuzzy IBE2
Sending attributes	No	Yes	Yes
Performing satisfying access tree	No	Yes	Yes
Distance adjustment	Yes	No	No

measurements of performance and performance comparison.

A. Security Parameter Selection and Simulation Environment

Our implementation uses symmetric bilinear pairing and was implemented with PBC [33]. A 160-bit elliptic curve group $\mathbb{G}$ based on the supersingular curve $y^2 = x^3 + x$ over 512-bit finite field is adopted. Operations on the elements of group $\mathbb{G}$, such as addition, negation and exponentiation are computed through calling corresponding functions from PBC library. Random bits read from Linux kernel file /dev/urandom are used to generate random number from Z_q where q is the order of group $\mathbb{G}$. Using a computer with 4 Intel(R) Core(TM) i3-2130 CPUs running at 3.40 GHz, it costs approximate 1.14 ms to compute bilinear pairing, 1.51 ms and 0.14 ms on average to fulfill exponentiations in $\mathbb{G}$ and $\mathbb{G}_T$ respectively. As for adding operations of elliptic curve points, less than 0.001ms is consumed which is negligible.

OMNET++ 4.2.2 is used to build the framework of the FA protocol. CSP, data owner, ASP and AS are simulated as simple modules in the project. For simplicity, we fix the number of CSP, ASP and AS as one for each, but the number of data owner is flexible which can be assigned manually at the beginning of simulation. OMNET++ 4.2.2 provides two self-defined methods, *handleMessage()* and *activity()*, to receive and deal with data packets for each module. And each module has to choose one of them. In our simulation, we adopt *handleMessage()* function due to its convenience of co-working with library PBC.

FA mainly facilitates users who are prone to use smart phones and tablets to access the cloud storage. In order to make the simulation close to reality, before setting the parameters such as delay and bandwidth, we monitor communications between a smart phone and online websites in real life with WebSitePulse [34], a tool used to monitor internet communications. Depending on the websites a smart phone or tablet accesses and the WiFi to which a smart phone or tablet is connected, connection time and responding time varies. The effective upload bandwidth of the WiFi is 500Kbps and download speed is 65KBps.

TABLE II: Response delay parameters.

Dropbox	Chrome Web Store	Owner Device (Android)	PDFMerge
15ms	10ms	49ms	20ms

Under this circumstance, after one thousand tests for each cloud storage provider, there exist 2ms delay of https://drive.google.com, 29ms delay of https://skydrive.live.com , and 69ms delay of https://dropbox.com. As a compromise, we set 15ms as the communication delay between CSP and owner. The response delay of all the parties are summarized in Table II. Bandwidth of cloud storage provider is unlimited just as most cloud storage providers set in real life [35] and so as bandwidth of application store. Upload bandwidth of owner is 500Kpbs, and download bandwidth is 65KBps which is the normal real life smart phone communication parameters.

B. Optimizations

Comparing to CP-ABE, FA has $\binom{N}{K} - 1$ more interpolations which is time-consuming. Fortunately, by arranging the interpolation sequence properly, there are overlaps between the adjacent interpolations. Based on the overlaps, the computation of interpolation can be optimized. In the following, we present the interpolations arrangement and how to optimize the later interpolation based on the preceding interpolation.

Before each interpolation, a set $\mathcal{Z}'_s$ of K indexes is chosen in all possibilities. Let k range from 1 to $\binom{N}{K}$ and $\mathcal{Z}'_{s_k}$ be the kth set. Combination in lexicographical order algorithm [36] is used to form set $\mathcal{Z}'_s$ and further optimization could be done on top of it. In lexicographical order combination, the next $\mathcal{Z}'_{s_{k+1}}$ is constructed based on current combination $\mathcal{Z}'_{s_k}$ and the difference between them is only one component. So instead of conducting $\binom{N}{K}$ complete interpolations, the optimized procedure performs the first interpolation completely and rest $\binom{N}{K} - 1$ interpolations partialy.

For each index $j_u \in \mathcal{Z}'_{s_k}$, compute the corresponding exponential Lagrange polynomial as

$$w_{k,j_u}(x) = e(g_1,\ g_2)^{ra \prod\limits_{\forall i \in \mathcal{Z}'_{s_k}, i \neq j_u} \frac{(x-i)P_f(j_u)}{j_u - i}}. \tag{22}$$

Then we obtain set $\mathcal{W}_k = \{w_{k,0}(x), w_{k,1}(x), ..., w_{k,K-1}(x)\}$ with $w_{k,j_u}(x)$ defined as equation (22).

Let us denote the complementary set of $\mathcal{Z}'_{s_k}$ as $\mathcal{Z}^C_{s_k} = \{1,\ 2,\ ...,\ N\} \setminus \mathcal{Z}'_{s_k}$. Let $i_{old} \in \mathcal{Z}'_{s_k}$ be the old index to be replaced by the new index $i_{new} \in \mathcal{Z}^C_{s_k}$. Then the kth set $\mathcal{W}_k$ is updated to $\mathcal{W}_{k+1}$ as follows.

1) $j_u \in \mathcal{Z}'_{s_k}$ and $j_u = i_{old}$,

$$w_{k+1,j_u} = w_{k,j_u}; \tag{23}$$

2) $\forall j_u \in \mathcal{Z}'_{s_k}$ and $j_u \neq i_{old}$,

$$w_{k+1,j_u} = w_{k,j_u}^{\frac{inew-ju}{i_{old}-ju}}. \tag{24}$$

For every $\mathcal{W}_k$ set, the interpolation will always be

$$e(g_1,\ g_2)^{raP_f(x)} = e(g_1,\ g_2)^{ra\sum\limits_{u=1}^{K} w_{k,j_u}}, u = 1, 2, ..., N. \tag{25}$$

Before the optimization, for each set $\mathcal{Z}'_{s_k}$, interpolation costs $1 + 2(K-1)$ exponential operations on the element from group $\mathbb{G}_T$ and $\binom{N}{K}[1+2(K-1)]$ exponentials overall. The optimization reduces $1+2(K-1)$ exponential operations to 1 exponential operation on the element from group $\mathbb{G}_T$ and the overall number of exponential operations is reduced from $\binom{N}{K}[1 + 2(K-1)]$ to $2(K-1) + \binom{N}{K}$. The time consumption of distance checking and distance adjustment before and after optimization is given in Table III. The performance improvement can be easily seen by comparing these two columns.

Another optimization can be adopted when decryption is performed over the root of subtree where the checking nodes are added, i.e., the F-subtree in our case. Instead of computing the exponential polynomial $P_f(x)$, unknown x can be replaced by node index number and the interpolation result is a potential indirect share. Replacing x with indexes of root's children nodes in turn, a set of new indirect secret shares are obtained. Instead of choosing the most frequently occurring polynomial, advantage of parity check matrix $\mathcal{H}$ could be used. For each new set of share components obtained, (28) and (29) can be applied to check whether they are the correct share components. If (28) and (29) are satisfied for a certain set of potential share components, stop interpolation and replace the unknown x to 0 to obtain$e(g_1, g_2)^{raP_f(0)}$.

C. Performance Measurements

The experiment statistics and comparisons about time consumption, storage consumption and revocation efficiency are shown in this subsection.

1) Time Consumption: Comparing to the other authorization schemes, FA utilizes distance checking and adjustment. Our simulation results, shown in Table III, suggest that distance checking and adjustment is not very time consuming. As in Fuzzy IBE2, the transmission of file attribute set costs at least one

TABLE III: Time consumption of error checking and correction.

Time Consumption before Optimization	Time Consumption after Optimization	Attribute Number in F-subtree	Distance Unit
$60.28ms$	$47.45ms$	6	1
$161.72ms$	$106.49ms$	8	2
$109.07ms$	$68.63ms$	8	1
$83.15ms$	$79.91ms$	6	2

TABLE IV: Overall time consumption of fuzzy authorization.

Time Consumption	Number of Files	Attribute Number in F-subtree	Distance Unit
$1.287s$	3	6	1
$1.431s$	3	8	2
$1.291s$	3	8	1
$1.338s$	3	6	2

round trip time (RTT). In the most commonly used 3G and 4G networks, the average RTT is around or over 100ms [37]. The distance checking and adjustment of FA is more efficient, compared to the communication overhead cost by transmission of file attribute set. From the data in the first and second columns in Table III, it is obvious that the optimized algorithm helps to sharply reduce the computation time.

For one authorization operation of 3 different files with one unit distance tolerance, we have collected some detailed overall experimental results in Table IV. The experimental results show that the overall time of FA is slightly longer than one second. We have also collected the AAuth [4] with the same predefined communication latency parameters and hardware environment as FA. Under the same environmental settings, the authorization time of AAuth ranges between 1s to 1.2s. The experimental results suggest that the overall time of FA protocol is slightly longer than AAuth which is expected because of the computational overhead introduced by distance checking and adjustment.

2) Storage Consumption: In the access tree, each leaf node is associated with an attribute y. At the same time, two corresponding cipher components $C_y = g_2^{P_y(0)}, C'_y = H(y)^{P_y(0)}$ must be added into the ciphertext. Assume the total number of leaf nodes of the access tree is n, and the number of F-subtree leaf nodes is $\frac{n}{2}$. Let k be the number of archives that could be decrypted with the same KE and η be the distance that can be tolerated. In our simulation, $n = 16$ and k ranges from 1 to 10. Of all the attributes, $FileName$ and $FileLocation$ are most likely to be different and cause the distance between two attribute

sets. Hence two typical values of η, 1 and 2 are simulated. In Fuzzy IBE1, fuzziness of authorization can

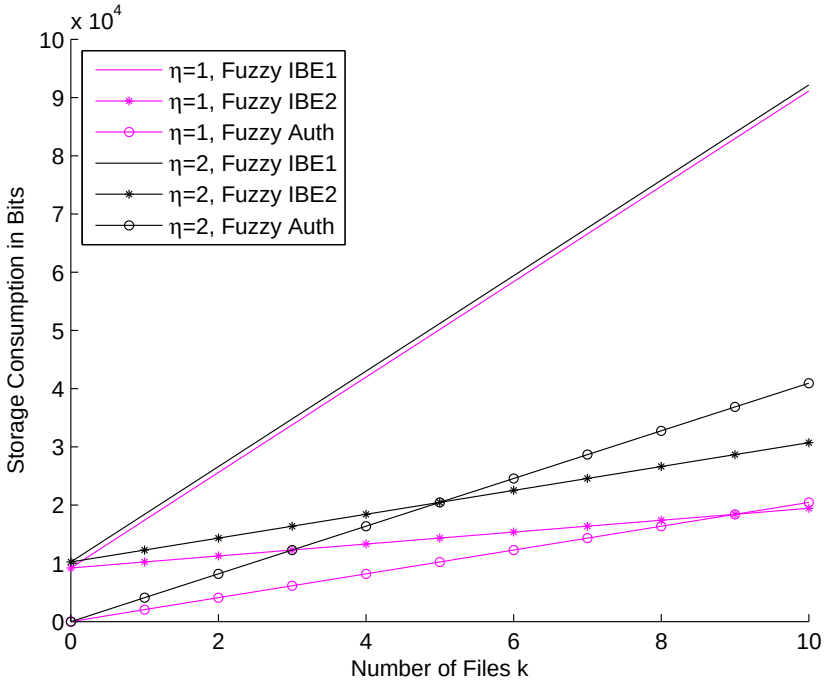


Fig. 3: Storage consumption of Fuzzy IBE and fuzzy authorization.

be achieved by changing the threshold value of F-subtree. Polynomial $P_f(x)$ and values of leaf nodes have to be recomputed. The extra cipher components for F-subtree leaf nodes must be updated and saved accordingly. At least $2 * \frac{n}{2}$ extra elements from group $\mathbb{G}$ are required.

As to the Fuzzy IBE2, extra default nodes are added into F-subtree resulting in extra cipher components to be mounted in the ciphertext, i.e., 2η group elements from $\mathbb{G}$. In addition, extra space for $\frac{n}{2}$ file attributes is needed.

In FA, according to the property of MDS code, for distance adjustment ability of η, at least 2η checking nodes are required. The number of extra elements from group $\mathbb{G}$ is 4η. As the distance adjustment ability η and the number of authorized files k grow, the storage consumption of FA grows faster than Fuzzy IBE2. Even though Fuzzy IBE2 has to store at least $\frac{n}{2}$ file attributes, when $k = 6, \eta = 2$, FA has a higher storage consumption as Fig. 3 shows. For the same reason, FA exceeds Fuzzy IBE2 in storage consumption when $k = 9, \eta = 1$.

From Fig. 3, it suggests that extra storage consumption of FA is alway less than Fuzzy IBE1. According to Fig. 3, when $\eta = 1, k < 10$ and $\eta = 2, k < 6$, FA has an advantage in storage consumption than Fuzzy IBE2.

3) Revocation Efficiency: Currently, most authorization scheme utilizes manual revocation. As the background of owner varies greatly and for a less-cared owner, he/she may easily forget revocation. We assume that once owner remembers, he/she will revoke. Therefore, based on Ebbinghaus Forgetting Curve, the probability of revocation failure is demonstrated in Fig. 4. Assume owner updates the original data at time t_{change}, then in FA scheme, the non-revocation probability before t_{change} is 100% and after t_{change} is 0%. As manifested in Fig. 4, the uncertainty of human brain may result in higher failure while revocation in FA is more determinate. And in a long run, revocation in FA is advantageous.

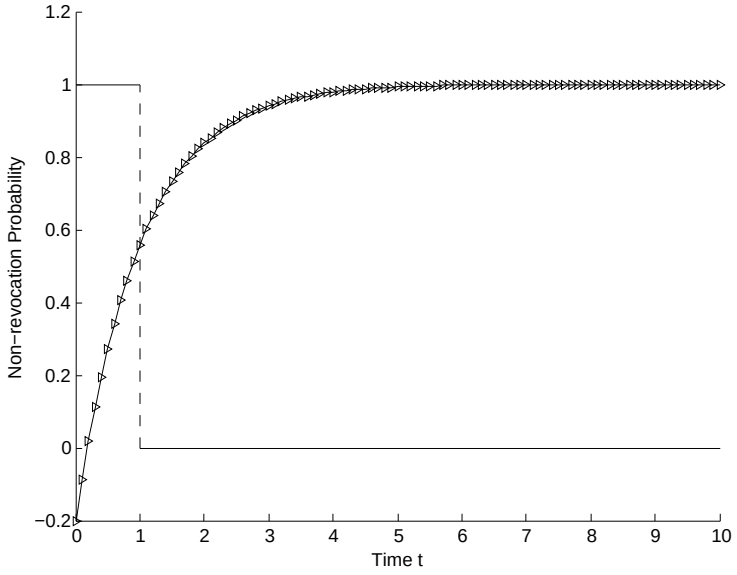


Fig. 4: Non-revocation probability of manually and fuzzy authorization.

VII. Conclusions and Future Work

In this paper, we propose FA which carries out a flexible file-sharing scheme between an owner who stores his/her data in one cloud party and applications which are registered within another cloud party. The simulation of FA protocol proves that our scheme can successfully adjust the attribute distance, quickly correct the unmatched indirect secret shares, resoundingly recover the top secret and then efficiently perform the decryption for KE. FA's self-distance-checking ability eliminates sending file attributes to ASP and distance-correcting ability omits necessity of performing *satisfying the access tree* procedure. Furthermore, the simulation indicates that with the update of $TimeSlot$ attribute, FA scheme automatically invalidates the authorized reading right from ASP. Comparing to Fuzzy IBE1 and Fuzzy IBE2, experimental results also demonstrates that FA reduces the storage consumption when distance is one unit

and number of authorization file is less than nine which is the most often occurring situation. The average time consumption of protocol collected in our simulation implies that FA is at the same efficiency level as AAuth.

While this work mainly addresses the reading authorization issue on cloud storage, the future work will aim to solve the security issue arising from writing right accreditation in cloud computing. For the latter, a more rigorous authentication is needed among data owner, ASP and AS, which makes the problem more challenging.

ACKNOWLEDGEMENT

The work was supported by NSERC SPG and ORF RE.

APPENDIX

In the appendix, we first demonstrate the original way of GRS encoding and decoding. We then introduce Berlekamp-Welch algorithm [28] and PGZ algorithm [29] followed by the analyses of why these two algorithms fail to be applied to FA scheme. At last, we briefly introduce the secret shares distribution and top secret construction of LSSS [38].

Denote $\mathbb{F}$ the finite field over which all calculations are computed. Define $\boldsymbol{\gamma} = (\gamma_0, \gamma_1, ..., \gamma_{N-1})$ as a vector of code locators and $\boldsymbol{v} = (v_0, v_1, ..., v_{N-1})$ a vector of column multipliers. Note that γ_i are distinct elements from finite field $\mathbb{F}$ and v_i are non-zero values (but not necessarily distinct) from $\mathbb{F}$. Let $\boldsymbol{r} = (r_0, r_1, ..., r_{N-1})$ be the received vector and $\boldsymbol{c} = (c_0, c_1, ..., c_{N-1})$ be the original codeword.

A. Original Methods of GRS Code Encoding and Decoding

1) GRS Code Encoding: Let message polynomial $p(x)$ be $\sum_{i=0}^{K-1} p_i x^i$, $p_i \in \mathbb{F}$. Then the corresponding codeword vector is presented as

$$\mathbf{c} = (c_0, c_1, ..., c_{N-1}) = (v_0 p(\gamma_0), v_1 p(\gamma_1), ..., v_{N-1} p(\gamma_{N-1})). \tag{26}$$

2) GRS Code Error Checking: GRS code is a linear $[N, K]$ code with error correction ability $\eta = \lfloor \frac{N-K}{2} \rfloor$. Parity check matrix is defined as

$$\mathcal{H} \triangleq \begin{bmatrix} 1 & 1 & \cdots & 1 \\ \gamma_0 & \gamma_1 & \cdots & \gamma_{N-1} \\ \gamma_0^2 & \gamma_1^2 & \cdots & \gamma_{N-1}^2 \\ \vdots & \vdots & \ddots & \vdots \\ \gamma_0^{N-K-1} & \gamma_1^{N-K-1} & \cdots & \gamma_{N-1}^{N-K-1} \end{bmatrix} \begin{bmatrix} v_0 & 0 & 0 & \cdots & 0 \\ 0 & v_1 & 0 & \cdots & 0 \\ 0 & 0 & v_2 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & v_{N-1} \end{bmatrix}. \tag{27}$$

Suppose vector $\boldsymbol{r} = (r_0, r_1, ..., r_{N-1})$ is received. Denote the error vector as $\boldsymbol{\eta} = (\eta_0, \eta_1, ..., \eta_{N-1})$. Decoder first computes the syndrome vector as

$$\boldsymbol{s} = (s_0, s_1, ..., s_{N-K-1}) = \mathcal{H}\boldsymbol{r}^\top = \mathcal{H}(\boldsymbol{c}^\top + \boldsymbol{\eta}^\top) = \mathcal{H}\boldsymbol{\eta}^\top \tag{28}$$

where the syndrome elements are given by

$$s_l = \sum_{j=0}^{N-1} \eta_j v_j \gamma_j^l,\ l = 0,\ 1,\ ...,\ N-K-1. \tag{29}$$

An all-zero vector $\boldsymbol{s}$ indicates no error, otherwise error(s) exists and error correction is needed.

3) GRS Code Decoding: We adopt the original GRS decoding procedure, given by Reed and Solomon [30].

Given a received vector $\boldsymbol{r}$, decoder selects K out of N indices to form a set $U' = \{i_0,\ i_1,\ ...,\ i_{K-1}\}$, $U' \subset U = \{0,\ 1,\ ...,\ N-1\}$, in all possible ways. For each set U', decoder interpolates the potential message polynomial $p'(x)$ of degree $K-1$ as

$$p'(x) = p'_0 + p'_1 x + p'_2 x^2 +, ..., + p'_{K-1} x^{K-1} = \sum_{t=0}^{K-1} \prod_{\substack{j \in U', \\ j \neq i_t}} \frac{x - \gamma_j}{\gamma_{i_t} - \gamma_j} r_{i_t}. \tag{30}$$

Since all possible choices are selected, the potential genuine message polynomial occurs more often than any other polynomials [30]. We then re-encode the message by evaluating the potential genuine polynomial at γ_i to get codeword $\boldsymbol{c}'$. Denote $d(\boldsymbol{c}', \boldsymbol{r})$ as the distance between $\boldsymbol{c}'$ and $\boldsymbol{r}$, if $d(\boldsymbol{c}', \boldsymbol{r}) \leq \eta$, $\boldsymbol{c} = \boldsymbol{c}'$ is the original transmitted codeword. If not, decoding fails.

B. Berlekamp-Welch Algorithm

We define error locator polynomial $E(x)$ over $\mathbb{F}$ as

$$E(\gamma_i) = 0 \text{ where } r_i \neq c_i \text{ and } deg(E(x)) = \eta. \tag{31}$$

More specificly, equation (31) gives us

$$E(x) = \prod_{\gamma_i \in J} (x - \gamma_i), \text{ where } J = \{\gamma_i | r_i \neq c_i\} \tag{32}$$

and

$$|J| \leq \frac{N-K}{2}. \tag{33}$$

From equation (31), we can obtain that equations

$$r_i E(\gamma_i) = v_i P(\gamma_i) E(\gamma_i), i = 1, 2, ...N \tag{34}$$

always hold. Over $\mathbb{F}$, we define polynomial

$$Q(x) = P(x)E(x). \tag{35}$$

From equation (33) and (34), we have the following properties of $Q(x)$

$$deg(Q(x)) \leq \frac{N-K}{2} + K - 1 \tag{36}$$

$$\forall Q(\gamma_i) = \frac{E(\gamma_i)r_i}{v_i}, i = 1, 2, ...N. \tag{37}$$

Berlekamp-Welch decoder takes codeword length N, error correction ability η and the received vector $\boldsymbol{r}$ as input, and outputs either $P(x)$ or failure in some cases. The decoder contains two main steps.

1) By interpolation, decoder computes a non zero polynomial $E(x)$ of degree η satisfying (33) and (36) and another polynomial $Q(x)$ satisfying equation (37). Failure is output if there is no such polynomials $E(x)$ or $Q(x)$ satisfying previous conditions.
2) Let $P'(x) = \frac{Q(x)}{E(x)}$, and $\boldsymbol{c'} = (c'_0, c'_1, ..., c'_{N-1})$ with $c_i = v_i P'(\gamma_i), i = 1, 2, ..., N$. Let $d(\boldsymbol{c'}, \boldsymbol{r})$ denote the distance between codeword derived from $P'(x)$ and the received codeword. If $d(\boldsymbol{c'}, \boldsymbol{r}) \leq \eta$, we set $P(x) = P'(x)$.

Then we adapt the two steps of Berlekamp-Welch algorithm to our scheme.

1) Given $F_{Zi} = e(g_1, g_2)^{raP(\gamma_i)}, i = 1, 2, ..., N$, the decoder tries to interpolate $e(g_1,\ g_2)^{raQ(x)}$ and $e(g_1,\ g_2)^{raE(x)}$ under some confinements. With the elements of codeword given as exponents, interpolation of $e(g_1,\ g_2)^{raQ(x)}$ and $e(g_1,\ g_2)^{raE(x)}$ would be complicated but still feasible.
2) Computing $e(g_1,\ g_2)^{raP(0)}$, i.e. $e(g_1,\ g_2)^{ra\frac{Q(0)}{E(0)}}$.

Unfortunately, even the decoder interpolate the $e(g_1,\ g_2)^{raE(x)}$ successfully, there is no efficient way to compute $e(g_1, g_2)^{\frac{1}{E(0)}}$ over $\mathbb{Z}_q$ given $e(g_1, g_2)^{raE(0)}$. As a result, decoding with Berlekamp-Welch algorithm is hardly fulfilled in this occasion.

C. Peterson-Gorenstein-Zierler algorithm

In PGZ algorithm, syndrome polynomial $S(x) = 1 + \sum_{i=1}^{N-K-1} s_i x^i$ is defined based on vecter $\boldsymbol{s}$. Error locator polynomial is represented as $E(x) = 1 + \sum_{i=1}^{\eta} E_i x^i$. A certain connection between coefficients of $E(x)$ and $S(x)$ can be deducted and expressed as

$$\begin{bmatrix} s_1 & s_2 & \cdots & s_\eta \\ s_2 & s_3 & \cdots & s_{\eta+1} \\ \vdots & \vdots & \ddots & \vdots \\ s_\eta & s_{\eta+1} & \cdots & s_{2\eta-1} \end{bmatrix} \begin{bmatrix} E_\eta \\ E_{\eta-1} \\ \vdots \\ E_1 \end{bmatrix} = \begin{bmatrix} -s_{\eta+1} \\ -s_{\eta+2} \\ \vdots \\ -s_{2\eta} \end{bmatrix} \tag{38}$$

Coefficients of error locator polynomial can be obtained by solving equation (38). With further factorization of $E(x)$, set $\boldsymbol{J}$, i.e., the locations of where go wrong are revealed. From equation (32), error polynomial is obtained and hence the error vector $\boldsymbol{e}$.

Now let us analyze the PGZ algorithm in our scheme. In order to find the coefficients of error locator polynomial, equation (38) must be solved for regular GRS decoding. While in our case, the equation (38) can be decomposed and derived into the following equation set

$$\begin{cases} e(g_1,\ g_2)^{ra\sum_{i=0}^{\eta-1} s_i E_{\eta-1-i}} & = e(g_1,\ g_2)^{-ras_\eta} \\ e(g_1,\ g_2)^{ra\sum_{i=1}^{\eta} s_i E_{\eta-i}} & = e(g_1,\ g_2)^{-ras_{\eta+1}} \\ & \vdots \\ e(g_1,\ g_2)^{ra\sum_{i=\eta-1}^{2\eta-2} s_i E_{2\eta-2-i}} & = e(g_1,\ g_2)^{-ras_{2\eta-1}} \end{cases} \tag{39}$$

There is no efficient computable method to compute $e(g_1,\ g_2)^{raE_i}$ without knowing s_i. To summarize, PGZ algorithm does not apply to our occasion.

D. Shamir's $(K,\ N)$ Threshold Scheme

1) Distribute the top secret: Let q be a positive integer and $Z_q = \{0, 1, \cdots, q-1\}$ be a residue ring modulo q. Given a top secret $\zeta \in \mathbb{Z}_q$, construct a polynomial $p(x) = \zeta + p_1 x + p_2 x^2 + ... + p_{K-1} x^{K-1}$ where p_i is a random element from $\mathbb{Z}_q$. Secret shares are computed as

$$\zeta_i = p(x_i), i = 0, 1, 2, \cdots, N-1 \tag{40}$$

where $x_i \in \mathbb{Z}_q$ are distinct non-zero numbers and N is a positive integer.

2) Reconstruct the top secret: Using the given secret shares, one can reconstruct the top secret ζ as follows. For a given index set $U' = \{i_0,\ i_1,\ ...,\ i_{K-1}\} \subset \{0, 1, 2, ..., N-1\}$, the corresponding K distinct points on the 2-dimensional plane are $(x_{i_0},\ \zeta_{i_0})$, $(x_{i_1},\ \zeta_{i_1})$, ..., $(x_{i_{K-1}},\ \zeta_{i_{K-1}})$. By interpolation, a unique polynomial

$$p(x) = p_0 + p_1 x + p_2 x^2 + ... + p_{K-1} x^{K-1} = \sum_{k=0}^{K-1} \prod_{\substack{j \in U', \\ j \neq i_k}} \frac{x - x_j}{x_{i_k} - x_j} \zeta_{i_k} \tag{41}$$

is obtained. Hence given any different K out of N secret shares, the top secret ζ is recovered as

$$\zeta = p_0 = \sum_{k=0}^{K-1} \prod_{\substack{j \in U', \\ j \neq i_k}} \frac{0 - x_j}{x_{i_k} - x_j} \zeta_{i_k} \ . \tag{42}$$

REFERENCES

[1] http://www.thetop10bestonlinebackup.com/cloud-storage, 2013.

[2] http://www.pdfmerge.com/, 2013.

[3] D. Balfanz, B. de Medeiros, D. Recordon, J. Smarr, and A. Tom, "The oauth 2.0 authorization protocol," Internet Draft, 2011.

[4] A. Tassanaviboon and G. Gong, "Oauth and abe based authorization in semi-trusted cloud computing," in *Proceedings of the second international workshop on Data intensive computing in the clouds*. ACM, 2011, pp. 41–50.

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

[6] R. McEliece and D. Sarwate, "On sharing secrets and reed-solomon codes," *Communications of the ACM*, vol. 24, no. 9, pp. 583–584, 1981.

[7] K. Ren, C. Wang, and Q. Wang, "Security challenges for the public cloud," *IEEE Internet Computing*, vol. 16, no. 1, pp. 69–73, 2012.

[8] P. Samarati and S. D. C. di Vimercati, "Data protection in outsourcing scenarios: issues and directions," in *Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security*. ACM, 2010, pp. 1–14.

[9] I. Agudo, "Cryptography goes to the cloud," in *Secure and Trust Computing, Data Management, and Applications*, vol. 187. Springer Berlin Heidelberg, 2011, pp. 190–197.

[10] J. Xu, E.-C. Chang, and J. Zhou, "Weak leakage-resilient client-side deduplication of encrypted data in cloud storage," in *Proceedings of the 8th ACM SIGSAC symposium on Information, computer and communications security*. ACM, 2013.

[11] C. Wang, S. S. Chow, Q. Wang, K. Ren, and W. Lou, "Privacy-preserving public auditing for secure cloud storage," *IEEE Transaction on Computers*, vol. 62, no. 2, pp. 362–375, 2013.

[12] Y. Zhu, H. Wang, Z. Hu, G.-J. Ahn, H. Hu, and S. S. Yau, "Dynamic audit services for outsourced storages in clouds," *IEEE Transactions on Services Computing*, vol. 6, pp. 227–238, 2013.

[13] Q. Wang, C. Wang, K. Ren, W. Lou, and J. Li, "Enabling public auditability and data dynamics for storage security in cloud computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 22, no. 5, pp. 847–859, 2011.

[14] H. Shacham and B. Waters, "Compact proofs of retrievability," in *Proceedings of the 14th International Conference on the Theory and Application of Cryptology and Information Security: Advances in Cryptology*. Springer-Verlag Berlin, 2008, pp. 90–107.

[15] C. C. Erway, A. Kp, C. Papamanthou, and R. Tamassia, "Dynamic provable data possession," in *Proceedings of the 16th ACM conference on Computer and communications security*. ACM, 2009, pp. 213–222.

[16] V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-based encryption for fine-grained access control of encrypted data," in *IACR Eprint archive*. ACM, 2006, pp. 89–98.

[17] A. Sahai and B. Waters, "Fuzzy identity-based encryption," in *Advances in Cryptology - EUROCRYPT 2005*, vol. 3494. Springer Berlin Heidelberg, 2005, pp. 457–473.

[18] B. Waters, "Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization," in *Public Key Cryptography - PKC 2011*, vol. 6571. Springer Berlin Heidelberg, 2011, pp. 53–70.

[19] D. Boneh and X. Boyen, "Efficient selective-ID secure identity based encryption without random oracles," in *Advances in Cryptology-EUROCRYPT 2004*, vol. 3027. Springer Berlin Heidelberg, 2004, pp. 223–238.

[20] W. Wang, Z. Li, R. Owens, and B. Bhargava, "Secure and efficient access to outsourced data," in *Proceedings of the 2009 ACM workshop on Cloud computing security*. ACM, 2009, pp. 55–65.

[21] S. Yu, "Data sharing on untrusted storage with attribute-based encryption," Ph.D. dissertation, Worcester Polytechnic Institute, MA, USA, July 2010.

[22] S. Yu, C. Wang, K. Ren, and W. Lou, "Attribute based data sharing with attribute revocation," in *Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security*. ACM, 2010, pp. 261–270.

[23] ——, "Achieving secure, scalable, and fine-grained data access control in cloud computing," in *Proceedings of the 29th conference on Information communications*. IEEE, 2010, pp. 534–542.

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

[25] T. O. Eiichiro Fujisaki, "Secure integration of asymmetric and symmetric encryption schemes," *Journal of Cryptology*, vol. 26, no. 1, pp. 80–101, 2013.

[26] S. Chatterjee and A. Menezes, "On cryptographic protocols employing asymmetric pairings - the role of ψ revisited," *Discrete Applied Mathematics*, vol. 159, pp. 1311–1322, 2011.

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

June 2007.
[28] E. R. Berlekamp and L. R. Welch, “Error correction for algebraic block codes,” U.S. Patent US 4 633 470 A, september, 1983.
[29] D. Gorenstein, W. W. Peterson, and N. Zierler, “Two-error correcting bose-chaudhuri codes are quasi-perfect,” *Information and Control*, vol. 3, no. 3, pp. 291–294, 1960.
[30] I. Reed and G. Solomon, “Polynomial codes over certain finite fields,” *Journal of Society for Industrial and Applied Mathematics*, vol. 8, no. 2, pp. 300–304, 1960.
[31] L. R. Welch, “The original view of reed-solomon codes,” http://csi.usc.edu/PDF/RSoriginal.pdf, 1997.
[32] M. Backes, C. Cachin, and A. Oprea, “Secure key-updating for lazy revocation,” in *Proceedings of the 11th European conference on Research in Computer Security*. Springer Berlin Heidelberg, 2006, pp. 327–346.
[33] B. Lynn, *PBC Library Manual*, http://crypto.stanford.edu/pbc/manual.pdf, 2006.
[34] “Website test tools,” http://www.websitepulse.com/help/tools.php, 2013.
[35] http://support.justcloud.com/question/205/is-there-a-bandwidth-limit, justCloud.
[36] C. J. Mifsud, “Algorithm 154: Combination in lexicographical order,” *Communications of the ACM*, vol. 6, no. 3, p. 103, 1963.
[37] Y.-C. Chen, E. M. Nahum, R. J. Gibbens, D. Towsley, and Y. sup Lim, “Characterizing 4g and 3g networks: Supporting mobility with multi-path tcp,” University of Massachusetts Amherst, Tech. Rep., 2012.
[38] A. Shamir, “How to share a secret,” *Communications of the ACM*, vol. 22, no. 11, pp. 612–613, 1979.

Shasha Zhu received the BSc degree (2011) of Information Security from Wuhan University and MSc degree (2013) from Electrical and Computer Engineering, University of Waterloo. She is the recipient of first prize in VMware Chinese University Students Cloud Computing Originality Contest. Her research interests include communication security and cloud computing.

Guang Gong received a B.S. degree in Mathematics in 1981, an M.S. degree in Applied Mathematics in 1985, and a Ph.D. degree in Electrical Engineering in 1990, from Universities in China. She received a Postdoctoral Fellowship from the Fondazione Ugo Bordoni, in Rome, Italy, and spent the following year there. She was promoted to an Associate Professor at the University of Electrical Science and Technology of China. During 1995-1998, Dr. Gong worked with several internationally recognized, outstanding coding experts and cryptographers, including Dr. Solomon W. Golomb, at the University of Southern California. Dr. Gong joined the University of Waterloo, Canada in 1998, as an Associate Professor in the Dept. of Electrical and Computer Engineering in September 2000. She has been a full Professor since 2004. Dr. Gongs research interests are in the areas of sequence design, cryptography, and communication security. She has authored or co-authored extensive amount of technical papers and two books, Signal Design for Good Correlation for Wireless Communication, Cryptography and Radar (2005), co-authored with Dr. Golomb, Communication System Security (2012), coauthored with Dr. Lidong Chen. Dr. Gong serves/served as Associate Editors for several journals including Associate Editor for Sequences for IEEE Transactions on Information Theory, and served on a number of technical program committees and conferences as co-chairs or committee members. Dr. Gong has received several awards including the Best Paper Award from the Chinese Institute of Electronics in 1984, Outstanding Doctorate Faculty Award of Sichuan Province, China, in 1991, the Premiers Research Excellence Award, Ontario, Canada, in 2001, NSERC Discovery Accelerator Supplement Award, 2009, Canada, and Ontario Research Fund - Research Excellence Award, 2010, Canada, Best Paper Award of IEEE ICC 2012, and IEEE Fellow 2014.