Fuzzy Authorization for Cloud Storage

by

Shasha Zhu

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Applied Science
in
Electrical and Computer Engineering

Waterloo, Ontario, Canada, 2013

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

It is widely accepted that OAuth is the most popular authorization scheme adopted and implemented by industrial and academic world, however, it is difficult to adapt OAuth to the situation in which online applications registered with one cloud party intends to access data residing in another cloud party. In this thesis, by leveraging Ciphertext-Policy Attribute Based Encryption technique and Elgamal-like mask over the protocol, we propose a reading authorization scheme among diverse clouds, which is called *fuzzy authorization*, to facilitate an application registered with one cloud party to access to data residing in another cloud party. More importantly, we enable the fuzziness of authorization thus to enhance the scalability and flexibility of file sharing by taking advantage of the innate connections of Linear Secret-Sharing Scheme and Generalized Reed Solomon code. Furthermore, by conducting error checking and error correction, we eliminate operation of satisfying a access tree. In addition, the automatic revocation is realized with update of TimeSlot attribute when data owner modifies the data. We prove the security of our schemes under the selective-attribute security model. The protocol flow of fuzzy authorization is implemented with OMNET++ 4.2.2 and the bi-linear pairing is realized with PBC library. Simulation results show that our scheme can achieve fuzzy authorization among heterogeneous clouds with security and efficiency.

Acknowledgements

First I express my appreciation and gratitude to Prof. Gong for her patient guidance throughout the MASc program. I could not have finished the project without Prof. Gong's scrupulous supervision. Prof. Gong organizes the weekly group meeting from which I learned a lot and she spurs everybody to share the news from both academia and industrial world at the beginning of the meeting. She is an excellent advisor who always sparkle her wisdom and ignite the potentialities of me. Also I must thank my co-supervisor Prof. Bosco Leung, from whom I gain the knowledge of integrated circuits.

And then the thanks are given to my committee members, Professor Mahesh Tripunitara and Professor Oussama Damen, for their valuable comments and suggestions on my thesis.

Also, I would like to thank Dr. Yin Tan who have given me valuable advice about the format and content of the thesis. I must express my appreciation to Bo Zhu who is a talented colleague and always share germinal ideas and most up-to-date news to me. Of course thanks must go to Dr. Xinxin Fan, who aided me to understand the elliptic curve better and provided many practical suggestions.

In addition, I am grateful to our internal reading group. Together, we read the book The Theory of Error-Correcting Codes and share the knowledge that we have mastered. From the reading group, I learned self-motivation and collaboration.

Moreover, special thanks to all my colleagues and friends: Kalikinkar Mandal, Teng Wu, Yang Yang, Tassanaviboon Anuchart, Muhammad Khizer Kaleem, Yao Chen, Fei Huo and Gangqiang Yang for their continuous supports and suggestions on my research. I was having a great time with them, we studied together, we discuss together and we play together. I cherish the friendship with all the time.

Besides the people from Communication Security group, I would like to thank Xiaohui Liang and Dr. Hao Liang both from Broad Band Communication Research group. I probed and examined our scheme to make it more reliable through discussing with them.

Dedication

to my parents and my brother

for their endless love and support

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Cloud storage is a model of networked enterprise storage where data is stored in virtualized pools of storage which are generally hosted by third parties. The third parties, or cloud service providers (CSPs) operate large data centres. Clients of cloud service providers who require their data to be hosted buy or lease storage capacity from CSPs. Those clients are called data owners, or owners for short. The data center operators, in the background, virtualize the resources according to the requirements of the customer and expose them as storage pools, which the customers can themselves use to store files or data objects. Data hosted in the cloud is referred as outsourced data. Physically, the resource may span across multiple servers. The safety of the files depends upon the hosting websites.

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 service provided by companies like JustCloud, Google Drive and so on. In the meantime, cloud computing services are boosting as well. There were 360 million-plus users and 32 thousand-plus applications merely in Google Chrome Web Store by the middle April 2013 [1]. 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 [2]. 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 [3]. 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. Since the direct authorization solution outweighs the downloading and uploading operations in perspective of flexibility, scalability, efficiency and convenience, a proper and secure authorization protocol is needed.

One of the main obstacles is that it is tough to build trust between owner and cloud application, e.g., pdfmerge, because they are residing in diverse cloud parties. Another unwieldy issue is that multiple access tokens and secret keys are needed rather than one if owner wants to authorize access right of several files. Therefore, a scheme that builds the trust between owner and applications and reduces the number of access tokens and secret keys is required.

It is widely accepted that OAuth [4] is the most widely-adopted authorization scheme, unfortunately, it is infeasible to address the situation mentioned above. This is because Oauth protocol requires both resource data and accessing application to be in the same domain. For example, http://pixlr.com, a web-application targeting on editing pictures online, registered with Google Chrome Web Store which can easily access to data residing in Google Drive, but can hardly edit pictures from JustCloud. By introducing a trusted organization Authority which maintains the integrity of cloud application service provider, AAuth proposed by Tassanaviboon *et al.* addressed a similar situation in which owner and consumer are not the same domain [5]. Unfortunately, the feeble scalability of authorization in AAuth does not fix multiple authorizations required by the situation mentioned above.

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) [6] and Oauth. Moreover, FA is suitable for owner to share encrypted data with others and keep the content of data from being known by the cloud storage provider.

The term *fuzzy* indicates that our authorization scheme has attribute-discrepancy tolerance. Depending on where the checking nodes are added, different attribute sets will possess error-tolerance ability and hence different functionality can be achieved. For example, if

the checking nodes are inserted into the sub-tree of file attributes, the file-attribute set will gain error-tolerance property. Therefore one secret key issued to an application could be used to access multiple files which share a large enough overlap on the file-attribute set. While if we add the checking nodes into the sub-tree of application attributes, the archive without changing or updating the access policy could be accessed by multiple applications as well as enough numbers of attributes are shared among the applications.

Authorization schemes supporting fuzziness can also be derived from Fuzzy IBE. However, the leakage of file attributes to ASPs is needed. Knowing the file attributes, one can easily deduce a certain amount of private information of owner and hence jeopardize the privacy of owner. For example, last modified time of file alludes owner's activity time. Comparing to the authorization schemes derived from Fuzzy IBE, our scheme avoids file attributes' leakage and protects owner's privacy thoroughly.

The key features of FA include:

- Fuzzy authorization enables data owner to share their data with applications from a different cloud party.
- By exploiting the transformation from Linear Secret-Sharing Scheme (LSSS) to General Reed Solomon (GRS) code and inserting checking nodes into the access tree, FA enhances the scalability and flexibility of file-sharing. Moreover, through error detection and error correction, FA avoids owner sending file attributes to application service providers and eliminates the procedure of satisfying access tree [7].
- 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 proposes a new secure authorization scheme for cloud storage providing error tolerance, called fuzzy authorization (FA).
2. The security analysis shows that our FA scheme provides a thorough security of outsourced data, including confidentiality, integrity and secure access control.

3. The simulation results demonstrates that FA reduces the storage consumption compare to other similar possible authorization schemes. Simulation of FA protocol also suggests that our scheme could achieve the error tolerance and realize fuzzy authorization

The rest of the thesis is organized as follows: The literature survey is discussed in the Chapter 2 and preliminaries are introduced in Chapter 3. While in Chapter 4, we present the constructions of our scheme and the protocol procedures. Then detailed security reduction and analysis are then given in Chapter 5. In Chapter 6, we demonstrate implementation environment, optimizations and experience results. Finally, in Chapter 7, we make a conclusion and give out the future work.

Chapter 2

Literature Survey

Cloud storage has grown to become popular and is adopted by many individuals and organizations. The widely adoption of cloud storage raised several security concerns about the outsourced data, such as confidentiality, integrity and access control of the data. Both academic and industrial world are making efforts to maintain the security of the outsourced data.

2.1 Encryption Solutions for Cloud Storage

Cloud storage providers are neither considered as fully trustworthy nor are resistant to attacks because they have access to the storage infrastructure. So the encryption of owner's data seems to be necessary. A plenty of encryption solutions are devised and introduced into cloud computing environment. For the confidentiality of the outsourced data, Agudo *et al.* suggested several encryption schemes that can be adopted in cloud storage environment [8]. Xu *et al.* [9] adopt the traditional AES encryption for their scheme and introduce the access policy on the top of this encryption.

On the premise that individuals previously possessed the plain data M and stored the secret key k derived from M in their local storage, Davida *et al.* introduce their encryption solution for cloud-based storage [10]. Since the secret key is derived as the hash value of M,

the requirement of pre-sharing key among individuals is avoided. Moreover, by leveraging Error Correction Code (ECC) encoding and decoding, they suggest a scheme to achieve compressed encryption for slightly different messages. In their compressed encryption construction, messages M_1 and M_2 are similar. Message M_1 is first decoded as canonical M', and the difference vector $\delta_1 = M_1 \bigoplus M'$ is computed. For the similar message M_2, $\delta_2 = M_2 \bigoplus M'$ is calculated as well. In order to reduce the storage consumption, the cipher-text M' and the compressed δ_1 and δ_2 are stored into the cloud rather than cipher-texts of both M_1 and M_2.

Vimercati *et al.* propose an encryption scheme relying on the translation from the access control policy to an equivalent encryption policy which will reduce the number of keys and amount of encryption [11]. In order to enable a cloud storage user to authorize the limited access right to a desired group of other users, an external honest-but-curious service is introduced to manage the authorization policies. But the external service is unable to approach the plain data and prevent authorized user to access data. The plain data is encrypted with a symmetric key. By exploiting a Diffie-Hellman key agreement method, the symmetric key is derived from a secret held by each group user. So only the users who hold the appointed secrets can obtain the key and perform decryption.

2.2 Integrity Schemes for Cloud Storage

Besides confidentiality, integrity is another significant security concern for cloud storage. As the outsourced data is in control of a cloud storage provider rather than owner, the data can be easily tampered due to intentional or unintentional reasons.

Several researchers suggests to adopt a third party auditor (TPA) to maintain the integrity of owner's data stored in cloud [12] [13]. Zhu *et al.* introduce a dynamic audit services for integrity verification [13], in which TPA regularly audit the integrity and availability of the outsourced data with index-hash table (IHT) and public verification parameters (PVP) that are previously stored in TPA . In addition, authorized entities with secret key *sk* have the ability to dynamically update IHT and PVP stored in TPA. Wang *et al.* suggest a TPA leveraging the homomorphic linear authenticator [14] to reduce the

communication and computation overhead compared to the straightforward data auditing approaches.

Rather than relying on a TPA, Bowers, Juels *et al.* devise High-Availability and Integrity Layer (HAIL) [15] for cloud storage to enhance the availability and integrity of data residing in cloud. In this paper, they combine the proof of retrievability (POR) [16] and proof of data possession (PDP) [17]. No third party auditor is needed because a single trusted verifier is attached with outsourced data and will be verified by a client or a service acting on behalf of a client.

2.3 Access Control for Cloud Storage

Works have been done as to migrate and adapt the mature traditional authorization management to cloud computing [18]. Besides that, a series of new access control schemes and solutions have been researched and devised for cloud environment based on the general access control solutions.

Of all the access control architectures, Attribute-Based Encryption (ABE) schemes are the most popular ones due to its scalability and security. Unlike Access Control List (ACL) only defines which entities have the access right, ABE schemes encrypt the data under the access policy which only ensure the eligible entities to do decryption. A distinguished work Fuzzy Identity-Based Encryption (IBE) [19] was introduced by Sahai and Waters in 2005. In Fuzzy IBE scheme, a private key for an identity set ω, can be used to decrypt a cipher-text encrypted with an 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. Later based on Fuzzy IBE, Goyal *et al.* present Keypolicy-Attribute Based Encryption (KP-ABE) [20] in which cipher-texts are labelled with sets of attributes and private keys are associated with access structures that control which cipher-texts a user can decrypt. Bethencourt *et al.* then introduce a complementary scheme to KP-ABE, called Ciphertext-Policy Attribute-Based-Encryption (CP-ABE) [6] in which attributes are used to describe the user's credentials and the formulas over these credentials are attached to the cipher-text by the encrypting party. Waters supplies more concrete and general CP-ABE construc-

tions in later papers [21] [22]. Boneh constructed BB1 and BB2 approaches [7] to build Identity-Based Encryption. The hierarchical construction within BB1 and BB2 can be efficiently secured against chosen-ciphertext attack. More importantly, Boneh extended the underlying Diffie-Hellman assumption to asymmetric pairing which is more advantageous and practical. Both CP-ABE and KP-ABE can be easily adapted to cloud environment and hence a lot of research work are founded on them [5] [23] [24] [25].

Key to Cloud (K2C), is realized by Zarandioon *et al.* through Attribute Based-Hierarchical Key Updating (AB-HKU) [23]. Built on top of KP-ABE with an access tree, AB-HKU scheme supports efficient delegation and revocation of privileges for hierarchies as well as eliminates the requirements of complex cryptographic data structures. AB-HKU is especially convenient and efficient in revocation through one increment of the root threshold value.

Tassanaviboon *et al.* proposes an OAuth and ABE based authorization in semi-trusted cloud computing called AAuth [5]. 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. In AAuth, owner's data is first encrypted by a symmetric key; then the symmetric key is encrypted under modified CP-ABE. To ensure the integrity of the outsourced data, integrity tag is computed and attached with the cipher-texts. Only authorized consumer is granted with secret key to decrypt for the symmetric key.

A cryptographic-based access control [24] 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 addressed fine-grained data access control, efficient key/user management, user accountability and etc., for cloud storage in his dissertation [25].

A solution to address the proof of ownership and eliminate the unnecessary client-side duplication of users sensitive data files is devised by Xu *et al.* [9]. To protect data privacy from both outside adversaries and the honest-but-curious cloud storage server, they encrypts the sensitive data with AES method and introduces their own hash function and constructs Merkle Hash Tree (MHT) to provide *hash-as-a-proof* functionality. During the

process of *proof-of-ownership*, a cross-user provides the digest hash value and the random leaf node value of MHT required by the cloud server can be proved as owner and access to the data.

Due to the reason of economy and simplicity, most cloud environment tends to utilize the mature method or standardized method to handle the security concerns in the cloud storage. Google Drive, for example, authorize the access right based on OAuth standard [4].

From industrial aspect, Cloud Data Management Interface (CDMI) [26], was standardized by Storage Networking Industry Association (SNIA) specifying a protocol for self-provisioning, administering and accessing cloud storage. In CDMI, access control comprises the mechanisms by which various types of access to data are authorized and permitted or denied. CDMI uses the well-known mechanism of an ACL as defined in the NFSv4 standard [27].

2.4 Other Security Concerns for Cloud Storage

Besides the security of outsourced data, there are several other issues that might be considered.

Targeted on protecting users' consumption pattern of cloud computing resources, such as CPU time, storage space etc., anonymous yet authorized and bounded cloud resource schemes [28] are introduced by Slamanig. In the anonymous yet authorized and bounded cloud resource schemes, a *partially blindly signed* token comprising the setting where users should be able to register and obtain a resource bound from a cloud provider is granted to user. Convinced that the anonymous user's request for resource, computing or storage resource, does not exceed the limit, cloud provider grants the request. Therefore, there is no way for cloud provider to figure out the consumption pattern of a particular user due to the anonymity and unlink-ability.

Chapter 3

Preliminaries

In this chapter, we primarily introduce Shamir's $(K,\ N)$ threshold scheme in Section 3.1. Then some background about GRS encoding, error checking and decoding are reviewed in Section 3.2. Then in Section 3.3, the fundamental information of CP-ABE is given. At last the asymmetric bilinear pairing is given in Section 3.4 and the security assumption is demonstrated in Section 3.5.

3.1 Shamir's (K, N) Threshold Scheme

Secret sharing acts as an critical part in CP-ABE and hence Waters gives out the denifinition of a general Linear Secret-Sharing Scheme (LSSS) [21]. Shamir's $(K,\ N)$ threshold scheme is a typical LSSS which plays an essential role in constructing the access policy tree and the recovery of the top secret s.

3.1.1 Distribute the Shares of Top Secret

In order to share a top secret $s \in \mathbb{Z}_q$, we divide it into N pieces $s_i \in \mathbb{Z}_q$, $i \in U$ where U is an index set $\{1, 2, ..., N\}$. Given $p(x) = s + p_1x + p_2x^2 + ... + p_{K-1}x^{K-1}$, where p_i are

randomly selected from $\mathbb{Z}_q$. The secret shares are evaluated as

$$s_i = p(x_i) \tag{3.1}$$

where $x_i \in \mathbb{Z}_q$ are distinct non-zero numbers.

3.1.2 Reconstruct the Top Secret

Given an index set $U' = \{i_1,\ i_2,\ ...,\ i_K\}$ and K distinct points in the 2-dimensional plane $(x_{i_1},\ y_{i_1})$, $(x_{i_2},\ y_{i_2})$, ..., $(x_{i_K},\ y_{i_K})$, by interpolation, an unique polynomial

$$\begin{aligned} p(x) &= p_0 + p_1 x + p_2 x^2 +, ..., + p_{K-1} x^{K-1}, \\ \text{where } p_k(x) &= \prod_{j \in U', j \neq i_k} \frac{x - x_j}{x_{i_k} - x_j}; \end{aligned} \tag{3.2}$$

$$\text{or equivalently, } p(x) = \sum_{k=1}^{K} \prod_{j \in U', j \neq i_k} \frac{x - x_j}{x_{i_k} - x_j} y_{i_k} \tag{3.3}$$

can be reconstructed. Hence given any different K out N points, the top secret s can be recovered as

$$s = p(0) = \sum_{k=1}^{K} \prod_{j \in U', j \neq i_k} \frac{0 - x_j}{x_{i_k} - x_j} y_{i_k}\ . \tag{3.4}$$

3.2 Generalized Reed Solomon (GRS) Code Encoding and Decoding

3.2.1 GRS Code Encoding

Let $\mathbb{F}$ be a finite field with q elements, vector of code locators $\boldsymbol{\gamma} = (\gamma_1, \gamma_2, ..., \gamma_N) \in \mathbb{F}^{\mathbb{N}}$, where γ_i are distinct and vector of column multipliers $\boldsymbol{v} = (v_1, v_2, ..., v_N) \in \mathbb{F}^{\mathbb{N}}$ where $v_i \neq 0$. Let $\boldsymbol{p} = (p_0, p_1, ..., p_{K-1}), p_i \in \mathbb{F}$ be a message vector to be encoded and the

message polynomial is $p(x) = \sum_{i=0}^{K-1} p_i x^i$. Then the corresponding codeword vector is presented as

$$\begin{aligned} \mathbf{c} &= (c_1, c_2, ..., c_N) \\ &= (v_1 p(\gamma_1), v_2 p(\gamma_2), ..., v_N p(\gamma_N)). \end{aligned} \tag{3.5}$$

3.2.2 GRS Code Error Checking

GRS code is a linear $[N, K, d]$ code, where $d = N - K + 1$, with error correction ability $e = \lfloor \frac{N-K}{2} \rfloor$. The parity check matrix is defined as

$$\mathcal{H} \triangleq \begin{bmatrix} 1 & 1 & \cdots & 1 \\ \gamma_1 & \gamma_2 & \cdots & \gamma_N \\ \gamma_1^2 & \gamma_2^2 & \cdots & \gamma_N^2 \\ \vdots & \vdots & \ddots & \vdots \\ \gamma_1^{N-K-1} & \gamma_2^{N-K-1} & \cdots & \gamma_N^{N-K-1} \end{bmatrix} \begin{bmatrix} v_1 & 0 & 0 & \cdots & 0 \\ 0 & v_2 & 0 & \cdots & 0 \\ 0 & 0 & v_3 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & v_N \end{bmatrix}. \tag{3.6}$$

Suppose vector $\boldsymbol{r} = (r_1, r_2, ..., r_N)$ is received. We denote the error vector as $\boldsymbol{e} = (e_1, e_2, ..., e_N)$. Decoder computes the syndrome vector as

$$\begin{aligned} \boldsymbol{s} &= (s_1, s_2, ..., s_{N-K}) \\ &= \mathcal{H}\boldsymbol{r}^\top \\ &= \mathcal{H}(\boldsymbol{c}^\top + \boldsymbol{e}^\top) \\ &= \mathcal{H}\boldsymbol{e}^\top. \end{aligned} \tag{3.7}$$

This yields

$$s_{l+1} = \sum_{j=1}^{N} e_j v_j \gamma_j^l, \; l = 0, \; 1, \; ..., \; N - K - 1. \tag{3.8}$$

An all-zero vector $\boldsymbol{s}$ indicates that there is no error. Otherwise, error(s) exists and further error correction process must be performed.

3.2.3 GRS Code Decoding

In this subsection, we briefly review the three decoding algorithms of GRS code. Recall that $\boldsymbol{r} = (r_1, r_2, ..., r_N)$ is the received vector, $\boldsymbol{c} = (c_1, c_2, ..., c_N)$ is the codeword, $\boldsymbol{\gamma} =$

$(\gamma_1, \gamma_2, ..., \gamma_N)$ is the code locator vector and $\boldsymbol{v} = (v_1, v_2, ..., v_N)$ is the column multiplier vector.

Interpolation-based decoding and syndrome-based decoding are two well-known decoding types of GRS codes. Berlekamp-Welch algorithm [29], a typical interpolation-based decoding algorithm, and Peterson-Gorenstein-Zierler(PGZ) algorithm [30], a classical syndrome-based decoding procedure are reviewed here. Both algorithms are well known for their efficiency. Intuitively, one of these algorithms should be adopted to perform decoding. Unfortunately, none of them, nor the other advanced decoding algorithm is applicable in our system. A detailed representation of how these algorithms fail to decoding in our system is shown in the next chapter Section 4.5.5. Fortunately, Reed-Solomon's original decoding method can be adapted in our situation.

Berlekamp-Welch Algorithm

Let two vectors $\boldsymbol{x} = \{x_1, x_2, \ldots, x_n\}$ and $\boldsymbol{y} = \{y_1, y_2, \ldots, y_n\}$, $x_i, y_i \in \mathbb{F}$, the distance of $\boldsymbol{x}$ and $\boldsymbol{y}$ is defined as $d(x, y) = \{i | x_i \neq y_i, 1 \leq i \leq n\}$. We define $E(x)$ an error locator polynomial over $\mathbb{F}$ such that

$$E(\gamma_i) = 0 \text{ where } r_i \neq c_i \text{ and } deg(E(x)) = e. \tag{3.9}$$

That is

$$E(x) = \prod_{\gamma_i \in J} (x - \gamma_i), \text{ where } J = \{\gamma_i | r_i \neq c_i\} \tag{3.10}$$

and

$$|J| = e \leq \frac{N - K}{2}. \tag{3.11}$$

From equation (3.9), it is easy to check equation

$$r_i E(\gamma_i) = v_i P(\gamma_i) E(\gamma_i), i = 1, 2, ...N \tag{3.12}$$

will always hold. We now define a polynomial $Q(x)$ over $\mathbb{F}$ as

$$Q(x) = P(x)E(x). \tag{3.13}$$

From equations (3.11) and (3.12), it follows

$$deg(Q(x)) \leq \frac{N-K}{2} + K - 1 \tag{3.14}$$

and

$$Q(\gamma_i) = \frac{E(\gamma_i) r_i}{v_i}, i = 1, 2, ...N. \tag{3.15}$$

Berlekamp-Welch decoder takes the codeword length N, the number of errors e, and the received word $\boldsymbol{r}$ as input, and outputs either $P(x)$ or failure. The decoder contains two main steps.

1. By interpolation, decoder computes a non zero polynomial $E(x)$ of degree e such that (3.11), (3.14) and (3.15) hold. Failure will be outputted if there is no such polynomials $E(x)$ or $Q(x)$ satisfying those conditions.

2. Let $P'(x) = \frac{Q(x)}{E(x)}$, and $\boldsymbol{c'} = (c'_1, c'_2, ..., c'_N)$ where $c'_i = v_i P'(\gamma_i)$. Let $d(\boldsymbol{c'}, \boldsymbol{r})$ denote the distance between codeword derived from $P'(x)$ and the received codeword. If $d(\boldsymbol{c'}, \boldsymbol{r}) \leq e$, sets $P(x) = P'(x)$.

Peterson-Gorenstein-Zierler (PGZ) algorithm

In PGZ algorithm, syndrome polynomial $S(x) = 1 + \sum_{i=1}^{N-K-1} s_i x_i$ is defined based on vector $\boldsymbol{s}$. Error locator polynomial is represented as $\Lambda(x) = 1 + \sum_{i=1}^{e} \Lambda_i x^i$. Expanding the equation (3.12), a certain connection between coefficients of $\Lambda(x)$ and $S(x)$ can be deducted and expressed as

$$\begin{bmatrix} s_1 & s_2 & \cdots & s_e \\ s_2 & s_3 & \cdots & s_{e+1} \\ \vdots & \vdots & \ddots & \vdots \\ s_e & s_{e+1} & \cdots & s_{2e-1} \end{bmatrix} \begin{bmatrix} \Lambda_e \\ \Lambda_{e-1} \\ \vdots \\ \Lambda_1 \end{bmatrix} = \begin{bmatrix} -s_{e+1} \\ -s_{e+2} \\ \vdots \\ -s_{2e} \end{bmatrix}. \tag{3.16}$$

Denote the left-most matrix as $S_{e \times e}$ and the invert of this matrix as $S^{\top}_{e \times e}$. Solving the above equation will give us the coefficients of error locator polynomial. With further factorization of $\Lambda(x)$, set $\boldsymbol{J}$, i.e., the locations of where went wrong will be identified. From equation (3.10), error polynomial $E(x)$ can be obtained.

Original Decoding Algorithm

Given received vector $\boldsymbol{r} = (r_1, r_2, ..., r_N)$ and index set $U = \{1,\ 2,\ ...,\ N\}$, decoder selects K out of N indices in all possible ways to form subset $U'' = \{i_1,\ i_2,\ ...,\ i_K\}$ of U. Decoder then interpolates a potential message polynomial $p'(x)$ of degree $K-1$ as

$$p'(x) = p'_0 + p'_1 x + p'_2 x^2 +, ..., + p'_{K-1} x^{K-1} = \sum_{t=1}^{K} \prod_{j \in U'', j \neq i_t} \frac{x - \gamma_j}{\gamma_{i_t} - \gamma_j} r_{i_t}. \tag{3.17}$$

Consequently, the potential message is given as the vector of coefficients of $p'(x)$, i.e., $\boldsymbol{p'} = (p'_0, p'_1, ..., p'_{K-1})$. Since all possible selections are made, the most often occurring potential message polynomial gives a codeword closest to the received word [31]. However, we need to re-encode the message by evaluating $p'(x)$ at γ_i to get codeword $\boldsymbol{c'}$. Denote $d(\boldsymbol{c'}, \boldsymbol{r})$ as the distance of $\boldsymbol{c'}$ and $\boldsymbol{r}$, if $d(\boldsymbol{c'}, \boldsymbol{r}) \leq e$, $\boldsymbol{c} = \boldsymbol{c'}$ is the canonical codeword and $s = p'(0)$ is the top secret. If not, decoding fails.

Despite the fact that Reed-Solomon's original decoding method's inefficiency, in terms of small size messages and codewords, it is still useful and practical. Moreover, unlike the other advanced and efficient decoding algorithms, the original decoding method helps us circumvent the discrete logarithm problem in CP-ABE scheme. A detailed description of why advanced algorithm like Berlekamp-Welch algorithm and PGZ algorithm cannot be used is given in the next chapter. The original GRS decoding procedure brought up by I. S. Reed and G. Solomon [31] serves our purpose and hence is adopted here.

3.3 CP-ABE

CP-ABE method is conceptually close to traditional Role-Based Access Control (RBAC). In this section, we present the construction of access tree, procedure of satisfying an access tree, and the four main algorithms of CP-ABE.

3.3.1 CP-ABE Model

A sensitive message is encrypted under the access tree and a private key used to decrypt must have an attribute set S satisfying the access tree.

Access Tree $\mathcal{T}$

An access tree $\mathcal{T}$ is constructed with AND and OR gates. Each internal node x of $\mathcal{T}$ is a threshold gate attached with a threshold value k_x. Assume there are num_x children nodes of the internal node x, CP-ABE assigns indexes of the children nodes from 1 to num_x. When $k_x = num_x$, the threshold gate is an AND gate and when $k_x = 1$, the threshold gate is an OR gate.

Several functions are defined to facilitate the working with access trees. Like function $\mathsf{parent}(x)$ returns the parent of node x. Function $\mathsf{att}(x)$ represents the attribute that attached with node x when x is a leaf node. The number associated with each node is represented as $\mathsf{index}(x)$. Where the index values are uniquely assigned to nodes in the access structure for a given key in an arbitrary manner. Those functions are also utilized in our FA scheme.

Satisfying Access Tree

Denote r the root node of access tree. For an arbitrary node x in the access tree, $\mathcal{T}_x$ represents a sub-tree rooted at node x. A special case is when $x = r$, $\mathcal{T}_x$ is the access tree itself. If an attribute set S satisfy $\mathcal{T}_x$, we set $\mathcal{T}_x(S) = 1$.

Satisfying access tree is a recursive procedure starting from the root node. If x has no child, then $\mathcal{T}_x(S) = 1$ if and only if $\mathsf{att}(x) \in S$. If x has children, we set $\mathcal{T}_x(S)$ to 1 if and only if at least k_x children return 1 where k_x is the threshold value associated with x.

The procedure of *satisfying access tree* is used to select the matching secret key components when decrypt the cipher-text.

3.3.2 Construction of CP-ABE

CP-ABE construction is based on a symmetric bilinear pairing. There are four main procedures of CP-ABE, Setup, Encrypt, Delegate and Decrypt. The detailed discription of the four procedures can be found in [6] and hence is omitted in the thesis.

3.4 Bilinear Maps

Benefits such as a broader choice of elliptic curve implementations and more compact representations of group elements make asymmetric bilinear pairing more favourable if the symmetry is not explicitly required by a cryptographic scheme [7]. Hence, an asymmetric bilinear pairing is adopted in our cryptographic scheme. Some basic definitions and denotations about groups with efficient computable bilinear maps are introduced below.

Denote $\mathbb{G}_1, \mathbb{G}_2$ and $\mathbb{G}_T$ 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 bilinear pairing or bilinear map 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 $(q,\ g_1,\ g_2,\ \mathbb{G}_1,\ \mathbb{G}_2,\ \mathbb{G}_T)$ is called an asymmetric bilinear setting when $\mathbb{G}_1 \neq \mathbb{G}_2$. If $\mathbb{G}_1 = \mathbb{G}_2 = \mathbb{G}$, and g is a generator of $\mathbb{G}$, then the tuple $(q,\ g,\ \mathbb{G},\ \mathbb{G}_T)$ is a symmetric bilinear setting.

3.5 Decisional Bilinear Diffie-Hellman Exponent Assumption

Waters proposes the decisional parallel Bilinear Diffie-Hellman Exponent assumption [21] and introduces the security of CP-ABE on this assumption. Under a generalization for

asymmetric pairings, we introduce the computational Bilinear Diffie-Hellman Exponent assumption as follows:

We continue to use the notations of bilinear pairing from section 3.4. Let $s \in \mathbb{Z}_q$ be the target secret that adversary intends to recover and K be the threshold value attached with the target node. Denote $\widetilde{W}$ as an index set of secret shares and sets $\widetilde{W}'$ and $\widetilde{W}''$, where $|\widetilde{W}''| < K$, are two disjoint subsets of $\widetilde{W}$. Random numbers r, a, s, β, x_1, x_2, ..., x_N, y_1, y_2, ..., y_N, μ_1, μ_2, ..., μ_N, r_1, r_2, ..., r_N are chosen from $\mathbb{Z}_q$. Tuple

$$\begin{aligned}\bar{y} =&(g_1, g_2, g_2^{y_1}, g_2^{y_2}, ..., g_2^{y_N}, g_1^{\mu_1 y_1}, g_1^{\mu_2 y_2}, ..., g_1^{\mu_N y_N},\\ &g_1^{\mu_t r'_t},\ g_2^{r'_t}, g_1^{ra+\mu_t r''_t},\ g_2^{r''_t})|\forall t' \in \widetilde{W}', \forall t'' \in \widetilde{W}''\end{aligned} \tag{3.18}$$

is given. To distinguish a random element $T \in \mathbb{G}_T$ from $e(g_1, g_2)^{ras}$ is referred to as the decisional Bilinear Diffie-Hellman Exponent problem (d-BDHE). Let an algorithm $\mathcal{B}$ outputting $z \in \{0, 1\}$ has advantage ϵ in solving d-BDHE in ($\mathbb{G}_1$, $\mathbb{G}_2$) if

$$|Pr[\mathcal{B}(\bar{y}, e(g_1,\ g_2)^{ras}) = 0] - Pr[\mathcal{B}(\bar{y}, T) = 0]| \geq \epsilon. \tag{3.19}$$

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

Chapter 4

Fuzzy Authorization

We present the construction of fuzzy authorization (FA) in this chapter. First, we present the system model and overview of our protocol in Section 4.1. The access tree structure is established in Section 4.2. Then in Section 4.3, the archive format is introduced. The transformation from LSSS to GRS code is introduced in Section 4.4. We provide the main procedures and algorithms of FA in Section 4.5. At last, a comparison of FA and Fuzzy IBE adapted in authorization is demonstrated in Section 4.6.

4.1 System Model and Overview of FA

In this section, we present an overview of the system architecture, the compendium of protocol procedure and notations of our system as shown in Table 4.1. After that, several possible adversary models are demonstrated.

4.1.1 Overview of Protocol

There are four main parties in the system as displayed in Fig. 4.1. We assume that all parties hold validate public-key certificates from Certificate Authorities and communications among the four parties are protected by Transport Layer Security (TLS) channels.

Table 4.1: Notations In The Thesis.

Notations	Descriptions
ω	The overall attribute set
ω'	The attribute set of files
ω''	The attribute set of ASPs
e	Maximum number of error
$\boldsymbol{e}$	Error vector
$\boldsymbol{s}$	Syndrome vector
$\boldsymbol{\gamma}$	Vector of code locators
$\boldsymbol{v}$	Vector of column multipliers
$\widetilde{W}$	Index set of secret shares
$\widetilde{W'}$	Subset of $\widetilde{W}$; $\forall t' \in \widetilde{W'}, g_1^{u_t r'_t}$ and $g_2^{r'_t}$ are known by adversary
$\widetilde{W''}$	Subset of $\widetilde{W}$; $\forall t'' \in \widetilde{W''}, g_1^{ra+u_t r'_t}$ and $g_2^{r''_t}$ are known by adversary
$P_f(x)$	Polynomial attached to file sub-tree
$P_a(x)$	Polynomial attached to application sub-tree
U	The index set $\{1, 2, ..., N\}$
$\mathcal{Y}$	The set of all the leaf nodes in $\mathcal{T}$
$\mathcal{Y}'$	Subset of $\mathcal{Y}$; contains all the leaf nodes in F-subtree
$\mathcal{Y}'_s$	Subset of $\mathcal{Y}'$; a selected set of leaf nodes to perform interpolation

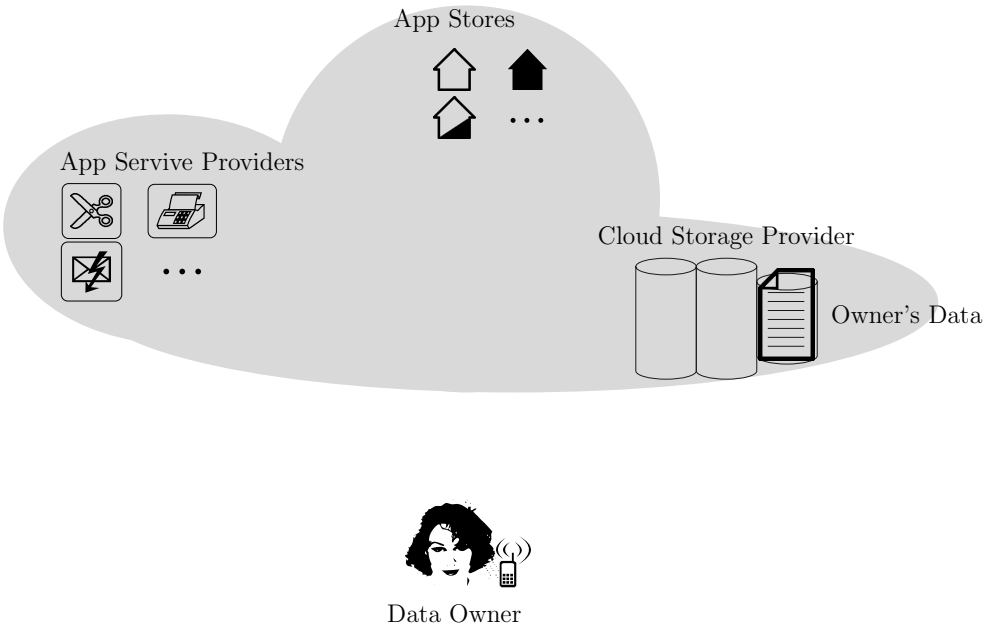


Figure 4.1: Example of System Model

Session tokens are adopted against replay attack during authentication. A final assumption is that only owner has writing permission to cloud storage while application service providers merely have a permission of reading.

- Application service provider (ASP): an application software resides on the vendor's system and is accessed by users through a web browser or through a special purpose client software provided by the vendor. For example, *http://pdfmerge.w69b.com/* is a website to merge several pdf files into one pdf file online. ASP and application are interchangeable in this thesis.

- Cloud storage provider (CSP): the entity which supplies storage as service to its clients and also provides access Application Platform Interfaces (APIs) to ASPs when ASPs hold an access token. Dropbox and JustCloud mentioned previously are such entities.

- Application store (AS): an entity with which the application service providers must

be registered to ensure the integrity of the applications. Google Chrome Web Store is a typical application store.

- Data owner: an entity who stores his or her data inside cloud storage and wishes to utilize cloud application services.

The protocol contains two phases, offline phase and running phase. In the offline phase, data owner encrypts his or her data with a random symmetric key KE and encrypts KE with our modified CP-ABE scheme, see details in Section 4.6. Then owner encapsulates cipher-text of KE and cipher-text of data as an archive file and stores the archive in the cloud. Format of the archive is defined in Section 4.3.

In the protocol running phase, when owner needs to share data with an ASP, 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. Indirect share means that the genuine secret share is an exponent or a part of exponent of a group elements. For example, when s_1 is known as a secret share and g_1 is a group element, $g_1^{s_1 r}$ is an indirect secret share.

In this thesis, we emphasize the flexibility of multiple-file sharing and therefore in our construction, the fuzziness is realized for the file attributes. As soon as ASP gets all the indirect secret shares, it will send a request to CSP for a formatted archive and then perform the decryption of archive header for KE. With KE, ASP decrypts the data cipher-text. The main objective of this thesis is to propose a secure and feasible way to address file-sharing issue with high scalability and flexibility in cloud storage, the method of owner accessing the resource data is not included in the scheme.

4.1.2 Adversary Models

Although entities do not trust each other, we assume that every entity will execute the protocol honestly. We consider the following five adversary situations.

1. CSP is trusted to provide storage services properly but may wants to access owner's

data illegally. CSP may take advantage of the indirect shares that he possesses and query the other indirect shares so as to reconstruct the top secret.

2. ASP may try to decrypt the unauthorized files by utilizing the previous indirect shares that issued to him. ASP is allowed to query for the indirect shares that he does not possess.

3. Application store which is in involved in issuing the indirect application secret shares may try to access to owner's data in the name of an ASP. Since he knows part of the indirect shares of application attributes, he may desire to query about the rest of indirect shares of application attributes and obtain the complete indirect shares of application attributes.

4. An adversary owner may impersonate other owners to construct the indirect secret shares with its own secret key.

5. Targeting on the secret keys and access tokens, general network attacks might be launched by internet hackers.

Figure 4.2 shows an example of authorization. Owner wishes to use pdfmerge, a cloud service provider to merge several pdf files stored in Dropbox into one pdf file. Instead of sharing the symmetric key KE directly with pdfmerge, owner encrypts the KE with modified CP-ABE and issues the secret key SK of CP-ABE to pdfmerge. Owner and Dropbox co-work together to issue the first part of SK and the common part D to pdfmerge. Then the owner and Google Chrome Web Store collaborate to issue the second part of SK to pdfmerge. After receiving the SK, pdfmerge requests for the encrypted file directly from Dropbox and Dropbox transmit the encrypted files to pdfmerge.

4.2 Access Tree Structure

Properly arranging access policy and inserting additional nodes at suitable places when authorize will help us achieve scalability and flexibility.

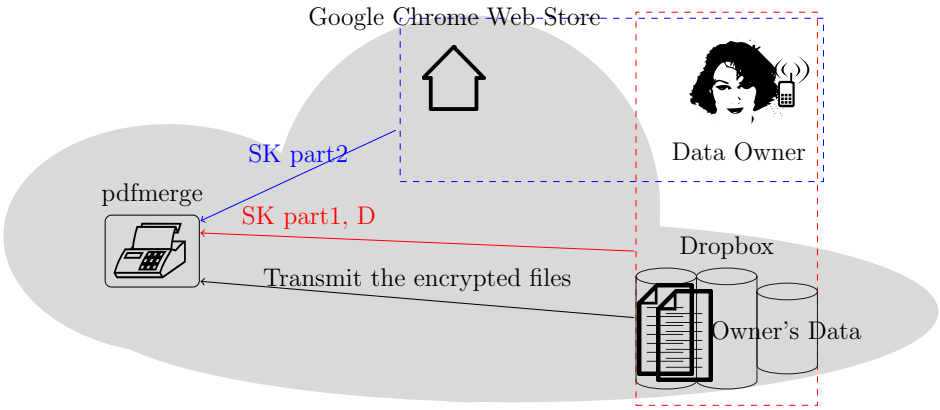
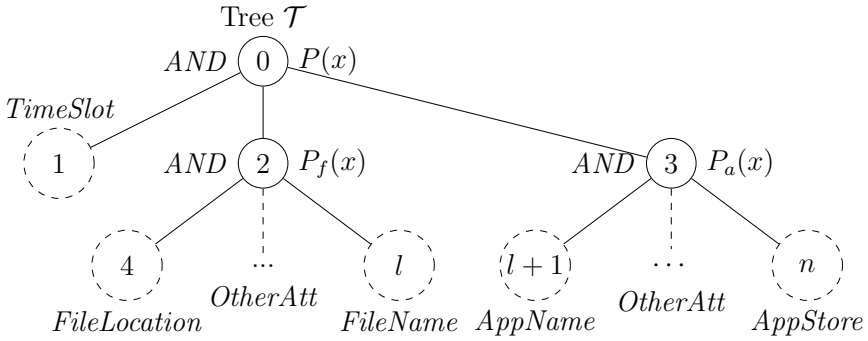
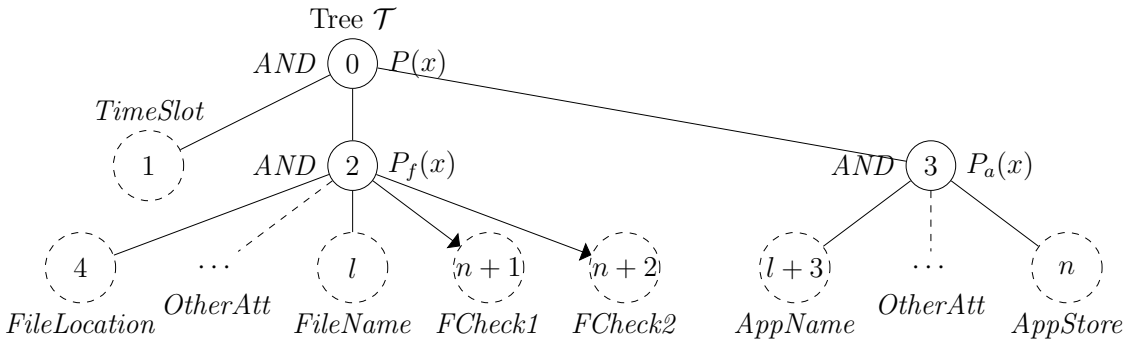


Figure 4.2: System Model



(a) Access tree $\mathcal{T}$ without checking nodes



(b) Access tree $\mathcal{T}$ with checking nodes in subtree of confined attributes

Figure 4.3: Access Tress Structure

4.2.1 Construction of Access Tree

Access tree structures are the same for all the files, but we assign different polynomials for the root nodes of access trees. The symmetric key KE used to encrypt the sensitive data is then encrypted under the access tree. Access trees are constructed with standard techniques [6] through ANDing operation. Sub-tree of file attributes, sub-tree of application attributes and the TimeSlot attribute are ANDed at the tree root node, as shown in Fig. 4.3(a). For abbreviation, let us call sub-tree of file attributes as F-sub-tree and sub-tree of application attributes as A-subtree. All file attributes, such as *FileName, FileLocation, FileType, FileOwner, FilePermission* etc. are ANDed at the root node of F-subtree. While A-subtree contains attributes like *AppStore, AppName, AppExpireDate, AppFunctionality, AppAuthor, AppAddress* etc. All the attributes are attached with leaf nodes which are drawn as dashed circles in Fig. 4.3(a) and Fig. 4.3(b). Each node in the tree is labelled with one index number. From now on, we will use indexes to represent the nodes. A polynomial attached with F-subtree root node is denoted as $P_f(x)$ and a polynomial attached with root node of A-subtree is called $P_a(x)$.

4.2.2 Adding Checking Nodes into the Tree

Before each authorization, owner chooses to enable the checking nodes or to disable the usage of checking nodes. If no redundant node is inserted, the issued secret key could only decrypt one single file without any security loss. However, in many occasions, applications need to access more than just one archive. For example, pdfmerge needs to access several pdf files to perform merging. By inserting appropriate number of redundant checking nodes into F-subtree, a token issued to the application could be used to decrypt different archives. Fig. 4.3(a) displays an example of adding two redundant nodes in the F-subtree which gives us one error tolerance. In Fig. 4.3(b), values of additional nodes are evaluated as $P_f(n+1)$ and $P_f(n+1)$. The new cipher components of the additional nodes are computed and appended to the archive.

Similarly, owner could insert the additional nodes in the A-subtree to empower one token to be used by several application. Further more, adding additional nodes in both

sub-trees will result in multiple applications gain access to multiple files. For simplicity, we only consider inserting redundant nodes in the F-subtree in this thesis.

4.3 Archive Format

In this section, we first present the archive format which supports the fuzzy authorization. The archive file mainly contains three parts, header, encrypted data, and the integrity tag. The format of the archive file is shown as follows.

$$< Archive > = < Header >_{ABE} || < Data >_{KE} || < InteTag > \tag{4.1}$$

where $< Data >_{KE}$ is the protected data encrypted with symmetric key KE, $< InteTag >$ is the integrity tag generated from $< Header >_{ABE} || < Data >_{KE}$. The structure $< Header >_{ABE}$ is relatively complex which is given as

$$\begin{aligned} < Header > = \ & < FileDesc > || < EncryptionMeth > || < InteMeth > \\ & || < KE > || < KV > || < \mathbb{A} >, \end{aligned} \tag{4.2}$$

where $< FileDesc >$ represents the description of protected-file content. $< EncryptionMeth >$ denotes the symmetric-key algorithm used to encrypted the data, $< InteTMeth >$ is a set of algorithm used to generate an integrity tag, such as RSA-MD5, RSA-SHA1, DSA-MD5, DSA-SHA1 etc., and $< KV >$ is the asymmetric key used to verify an integrity tag.

In our authorization, with the insertion of additional nodes, extra ciphertext contents need to be added as well. Hence the previous $< InteTag >$ is replaced with a new $< InteTag >$. Alternatively, owner computes several $< InteTag >$s beforehand, and uses the right $< InteTag >$ when authorize.

4.4 Transformation from Shamir's Linear Secret Sharing Scheme to GRS

From Shamir's (K, N) threshold scheme and GRS encoding and decoding algorithms, there is a transformation from secret distributing to GRS encoding and from secret recovery to

GRS decoding [32]. A detailed transformations from one to another is provided here.

4.4.1 Transformation From Secret Distributing to GRS Encoding

By setting column multipliers vector $\boldsymbol{v}$ to (1, 1, ..., 1) and the code locator vector $\boldsymbol{\gamma} = (x_1, x_2, ..., x_N)$ where x_i are the indexes of the nodes, the process of GRS encoding is basicaly the secret distributing procedure.

4.4.2 Transformation From Secret Recovery to GRS Decoding

As shown in equations (3.3) and (3.17), interpolation is the kernel part of both secret recovery and GRS decoding. The difference is that GRS codeword has N coordinates, of which $N - K$ are redundant and hence are used for error correction. So in order to take advantage of error correction ability from GRS, we will add some checking nodes into the access tree as redundant nodes in our scheme.

4.5 Main Procedures of Fuzzy Authorization

In lieu of using symmetric pairing which can be instantiated with merely suitable supersingular elliptic curves, we adopt asymmetric pairing which will allow a greater variety of constructed and ordinary curves to be used. A Type 2 bilinear pairing [33] 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, will determine 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 will map any binary string to a random element from $\mathbb{G}_1$ [34].

4.5.1 Setup(k)

The setup algorithm, is first initiated by CSP. CSP chooses generators $g_1,\ g_2$ of $\mathbb{G}_1$ and $\mathbb{G}_2$ and a bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ of prime order q according to the input security parameter k. Next CSP chooses a random exponent β and publishes the public key as:

$$CPK = \langle \mathbb{G}_1,\ \mathbb{G}_2,\ g,\ h = g_1^{\beta},\ f = g_2^{1/\beta} \rangle. \tag{4.3}$$

CSP's keeps $CSK = \langle \beta, g_2^{\gamma} \rangle$ as its secret key.

Later, each owner chooses a random exponent α and computes its public key and private key separately as

$$OPK = \langle e(g_1,\ g_2)^{\alpha} \rangle \text{ and } \ OSK = \langle g_2^{\alpha} \rangle. \tag{4.4}$$

4.5.2 Encrypt(CPK, OPK, m, $\mathcal{T}$)

Performed by owner, this algorithm encrypts a secret key KE under the access tree $\mathcal{T}$. Let $\mathcal{Y}$ denote the set of all the leaf nodes of $\mathcal{T}$ and $p_y(x)$ be the polynomial that assigned to a leaf node y. Then the cipher-text CT is given by

$$\begin{aligned} CT = \langle \mathcal{T},\ \widetilde{C} = KE \cdot e(g_1,\ g_2)^{\alpha s},\ C = h^s, \\ \forall y \in \mathcal{Y} : C_y = g_2^{p_y(0)},\ C'_y = H(att(y))^{p_y(0)} \rangle. \end{aligned} \tag{4.5}$$

If later $2e$ checking nodes are added, where $e > 0$, owner also computes the cipher components of checking nodes as $C_{n+1} = g_2^{p_{n+1}(0)}$, $C_{n+1} = g_2^{p_{n+2}(0)}$, ..., $C_{n+2e} = g_2^{p_{n+2e}(0)}$ and $C'_{n+1} = H(att(n+1))^{p_{n+1}(0)}$, $C'_{n+2} = H(att(n+2))^{p_{n+2}(0)}$, ..., $C'_{n+2e} = H(att(n+2e))^{p_{n+2e}(0)}$ where $n + i$ are the indexes of checking nodes.

4.5.3 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 will output common part D and a set of indirect secret shares of secret key.

First, Owner and CSP work together to compute $D = g_2^{(\alpha+ra)/\beta}$ in which $r \in \mathbb{Z}_q$ is chosen by CSP and $a \in \mathbb{Z}_q$ is selected by owner. The sequence of interactions ensures that owner only knows g_2^{ra} and CSP is merely aware of $g_2^{(\alpha+ra)/\beta}$ using method in [5]. 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 and time slot attribute, i.e., $\omega' \cup \{TimeSlot\}$ from owner, for any $i \in \omega' \cup \{TimeSlot\}$, CSP randomly chooses $r_i \in \mathbb{Z}_q$ and computes $H(i)^{r_i}$. Then owner computes $g_2^{ra} H(i)^{r_i}$ and sends them to ASP. ASP then authenticates itself to AS and presents the attributes of ω''. If authentication succeeds, for all $j \in \omega''$, AS will choose $r_j \in \omega''$ and compute $H(j)^{r_j}$. Again owner computes $g_2^{ra} H(j)^{r_j}$ and sends them to ASP. This algorithm ends up with ASP getting the SK which is represented as

$$SK = \langle D = g_2^{(\alpha+ra)/\beta},\ \forall t \in \omega : D_t = g_1^{ra} H(t)^{r_t},\ D'_t = g_2^{r_t} \rangle. \tag{4.6}$$

4.5.4 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 $\widetilde{r} \in \mathcal{Z}_q$ and for all$l \in \widetilde{\omega}, \widetilde{r}_l \in \mathcal{Z}_q$ are randomly picked. After that, a new private key $\widetilde{SK}$ for an attribute set $\widetilde{\omega}$ is generated as

$$\widetilde{SK} = \{\widetilde{D} = D f^{\widetilde{r}}, \forall k \in \widetilde{\omega} : \widetilde{D}_k = D_k g_1^{\widetilde{r}a} H(k)^{\widetilde{r}_k}, \widetilde{D}'_k = D'_k g_2^{\widetilde{r}_k}\}. \tag{4.7}$$

4.5.5 DecryptandErrorCorrect(CT, SK, $\mathcal{T}$)

The decryption algorithm is a recursive procedure over the access structure $\mathcal{T}$ comprising four steps. The algorithm is conducted by ASP.

Decryption on the Node

Let DecryptNode(CT, SK, x) denote the function that takes ciphertext CT, secret key SK and the node x in the tree as input. If x is a leaf node of the tree,

$$\begin{aligned} DecryptNode(CT, SK, x) &= \frac{e(D_i,\ C_x)}{e(C'_x,\ D'_i)} \\ &= \frac{e(g_1^{ra} H(i)^{r_i},\ g_2^{P'_x(0)})}{e(H(i)^{q_y(0)},\ g_2^{r_i})} \\ &= e(g_1,\ g_2)^{raP'_x(0)}. \end{aligned} \tag{4.8}$$

If x is the root node of F-subtree where the additional nodes are added, then for all child nodes z of x, the algorithm calls $DecryptNode(CT, SK, z)$ and stores the result as

$$f_z = e(g_1, g_2)^{raP'_z(0)}. \tag{4.9}$$

If the secret key issued is not designed to decrypt this file, i.e., the attributes set based on which the secret key is issued does not satisfy the access tree, error checking and error correction is needed. Note that, the attribute sets attached with access trees do not have any errors. We adopt the terminology of error correcting code. Here we use GRS for reconstructing the top secret. Thus, a single key can decrypt multiple files for which the attribute sets have distance less than or equal to η.

Error Checking

Let e be the maximum number of errors that can be tolerated. Then at least $2e$ additional nodes are added in the sub-tree. Error checking will be enforced and further error correction may be performed depends on the result of error checking.

According to the way we construct the access structure, the code locator vector is $\boldsymbol{\gamma} = (4,\ 5,\ ...,\ l,\ n+1,\ n+2,\ ...,\ n+2e)$ and column multiplier vector is $\boldsymbol{v} = (1,\ 1, ...,\ 1)$ with the length of $N = l - 3 + 2e$ which is also the number of leaf nodes in the sub-tree. Parity check matrix $\mathcal{H}$ can be easily obtained through equation (3.6) with $\boldsymbol{\gamma}$ and $\boldsymbol{v}$. Because the symbols of codeword or secret shares are exponents of $e(g_1,\ g_2)$, resembling

computing syndromes in equations (3.7) and (3.8), the checking procedure is performed over the exponent of $e(g_1,\ g_2)$ and the derived syndrome vector can be represented as

$$\begin{aligned}\boldsymbol{s'} =&(s'_1,\ s'_2,\ ...,\ s'_N)\\ =&(e(g_1,\ g_2)^{ra\sum_{j=0}^{n-1} e_j v_j \gamma_j^0},\ e(g_1,\ g_2)^{ra\sum_{j=0}^{n-1} e_j v_j \gamma_j 1},\ ...,\ e(g_1,\ g_2)^{ra\sum_{j=0}^{n-1} e_j v_j \gamma_j^{N-1}}).\end{aligned} \tag{4.10}$$

An all-one vector $\boldsymbol{s'}$ indicates no error, therefore no further correction will be needed. In this case, the decryption procedure continues to interpolate $F_x = e(g_1,\ g_2)^{P_f(0)}$. On the other hand, a non-all-one vector of $\boldsymbol{s'}$ indicates that error(s) does exist and the algorithm tries to correct the error(s).

Error Correction

Given leaf nodes set $\mathcal{Y}' = \{4, 5, ..., l, n+1, n+2, ..., n+2e\}$, construct the subset $\mathcal{Y}'_s = \{j_1,\ j_2,\ ...,\ j_K\} \subset \mathcal{Y}'$ which will give us $\binom{N}{K}$ different $\mathcal{Y}'_s$. For each set $\mathcal{Y}'_s$, do interpolation as

$$e(g_1,\ g_2)^{raP'_f(x)} = e(g_1,\ g_2)^{ra\sum_{t=1}^{K}\prod_{\forall i\in\mathcal{Y}', i\neq j_t}\frac{(x-i)f_{j_t}}{j_t-i}}. \tag{4.11}$$

$\binom{N}{K}$ results will be obtained. According to [31], the most often occurring result shall be selected and used to obtain another vector

$$\begin{aligned}\boldsymbol{f'_z} =&(e(g_1,\ g_2)^{raP'_f(4)},\ e(g_1,\ g_2)^{raP'_f(5)},\ ...,\ e(g_1,\ g_2)^{raP'_f(l)},\\ &e(g_1,\ g_2)^{raP'_f(n+1)},\ e(g_1,\ g_2)^{raP'_f(n+2)},\ ...,\ e(g_1,\ g_2)^{raP'_f(n+2e)}).\end{aligned} \tag{4.12}$$

Let $d(\boldsymbol{f_z},\ \boldsymbol{f'_z})$ denote the distance between $\boldsymbol{f_z}$ and $\boldsymbol{f'_z}$. If $d(\boldsymbol{f_z},\ \boldsymbol{f'_z}) \leq \lfloor\frac{N-K}{2}\rfloor = e$, set $e(g_1,\ g_2)^{raP_f(0)} = e(g_1,\ g_2)^{raP'_f(0)}$. Otherwise, error correction procedure fails and the decryption aborts. The reason why decoding process cannot be replaced by Berlekamp-Welch algorithm or PGZ decoding algorithm will be discussed later.

Final Decryption

If error checking and decoding go smoothly, we will get $A = DecryptNode(CT, SK, r) = e(g_1,\ g_2)^{ras}$. The encrypted message KE can be computed by

$$KE = Decrypt(CT,\ SK) = \frac{\widetilde{C}}{e(C, D)/A} = \frac{\widetilde{C}}{e(g_1^{\beta s},\ g_2^{(\alpha+ra)/\beta})/e(g_1,\ g_2)^{ras}} \tag{4.13}$$

Analysis of Other Decoding Algorithms

Decoding with Berlekamp-Welch Algorithm

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

1. Given $F_{Zi}= e(g_1, g_2)^{raP(\alpha_i)}, i = 1, 2, ..., N$, the decoder tries to interpolate $e(g_1, g_2)^{raQ(x)}$ and $e(g_1, g_2)^{raE(x)}$ under some confinements. Since the elements of codeword are given as exponents, the interpolation would be similar to regular operation as soon as we replace the regular summation with multiplication and change normal multiplication to power operation.

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 $e(g_1,\ g_2)^{raQ(x)}$ and $e(g_1,\ g_2)^{raE(x)}$, given $e(g_1,\ g_2)^{raE(0)}$, there is no efficient way to compute $e(g_1,\ g_2)^{\frac{1}{E(0)}}$ over $\mathbb{Z}_q$. As a result, decoding with Berlekamp-Welch algorithm will be hardly fulfilled in this scenario.

Decoding with PGZ Algorithm

In order to find the coefficients of error locator polynomial, equation (3.16) must be solved for regular GRS decoding. While in our case, the equation (3.16) can be decomposed and derived into the following equation set:

$$\begin{cases} e(g_1,\ g_2)^{ra\sum_{i=0}^{e-1} s_i E_{e-1-i}} & = e(g_1,\ g_2)^{-ras_e} \\ e(g_1,\ g_2)^{ra\sum_{i=1}^{e} s_i E_{e-i}} & = e(g_1,\ g_2)^{-ras_{e+1}} \\ & \vdots \\ e(g_1,\ g_2)^{ra\sum_{i=e-1}^{2e-2} s_i E_{2e-2-i}} & = e(g_1,\ g_2)^{-ras_{2e-1}} \end{cases} \tag{4.14}$$

where 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 this scenario.

4.5.6 Time Slot Synchronization

We divide time zone into small 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 [23] is used in our system regarding to re-encrypt sensitive data. When an application's access right is revoked, lazy revocation allows to postpone the update of polynomials and re-encryption of sensitive data until writing action has happened. Because 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 needs to collaborate together to re-encrypt the header file. Let $\mathcal{M}$ denote the children set of the root node, and $|\mathcal{M}| = m$. Assume the time-slot attribute is attached with node i^*, $i^* = 1$ in Fig.2, timeslot synchronization procedure initiates with owner chooses a random value $\widetilde{s_t}$, set the most up-to-date TimeSlot share as

$$P(i^*)_t = P_{TS}(0)_t = P_{TS}(0)_{t-1} + \widetilde{s_t}. \tag{4.15}$$

New cipher-text components for a new time slot can be obtained as $C_{TS_t} = g^{P(i^*)_t}$ and $C'_{TS_t} = H(TimeSlot)^{P(i^*)_t}$. From equation (3.4), the top secret can be obtained. More precisely,

$$\begin{aligned} s_t &= P(0)_t \\ &= \sum_{u=1}^{i^*-1} \prod_{\forall i \in \mathcal{M}, i \neq u} \frac{(0-i)P(u)}{u-i} \\ &+ \prod_{\forall i \in \mathcal{M}, i \neq i^*} \frac{(0-i)P(i^*)_t}{i^*-i} \\ &+ \sum_{u=i^*+1}^{m} \prod_{\forall i \in \mathcal{M}, i \neq u} \frac{(0-i)P(u)}{u-i}. \end{aligned} \tag{4.16}$$

So the discrepancy between s_t and s_{t-1} can be obtained based on equations (4.15) and (4.16)

$$\begin{aligned} \Delta_s =& s_t - s_{t-1} \\ =& \prod_{\forall i \in \mathcal{M}, i \neq i^*} \frac{(0-i)P(i^*)_t}{i^*-i} - \prod_{\forall i \in \mathcal{M}, i \neq i^*} \frac{(0-i)P(i^*)_{t-1}}{i^*-i} \\ =& \prod_{\forall i \in \mathcal{M}, i \neq i^*} \frac{0-i}{i^*-i} \widetilde{s_t}. \end{aligned} \tag{4.17}$$

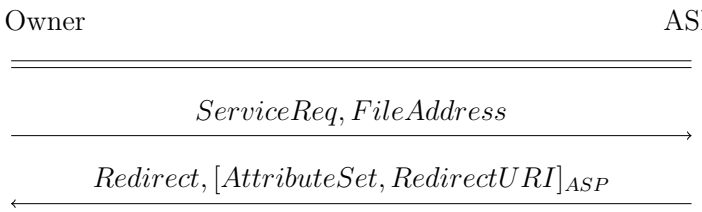


Figure 4.4: Service Request Flow

Number i^* is constant, so as $\prod_{\forall i \in \mathcal{M}, i \neq i^*} \frac{0-i}{i^*-i}$, the most left part of Δ_s. Hence the new ciphertext's main component $\widetilde{C}$ can be updated as $\widetilde{C} = m \cdot e(g_1,\ g_2)^{\alpha(s+\Delta_s)}$ and C as $C = h^{(s+\Delta_s)}$.

4.6 Fuzzy Authorization Protocol Flow

The flow of Fuzzy Authorization (FA) involves in four on-line protocols. With the requirement of adding checking nodes, FA protocol needs to re-encapsulate the archive file.

4.6.1 Service Request

1. Owner initiates by sending a request *ServiceReq* along with the files' addresses to ASP, for example, pdfmerge, *http://www.pdfmerge.com/*.

2. ASP redirects data owner's user agent to the authorization endpoint of CSP. ASP includes its identifier, functionality, local state and a redirection URI attributes to which the authorization server will send the user-agent back once access is granted(or denied).

4.6.2 Token and Secret Key Issuing

1. From the redirect command, owner's user agent passes the redirect command containing a targeted application attribute set and the redirect URI to CSP. An example

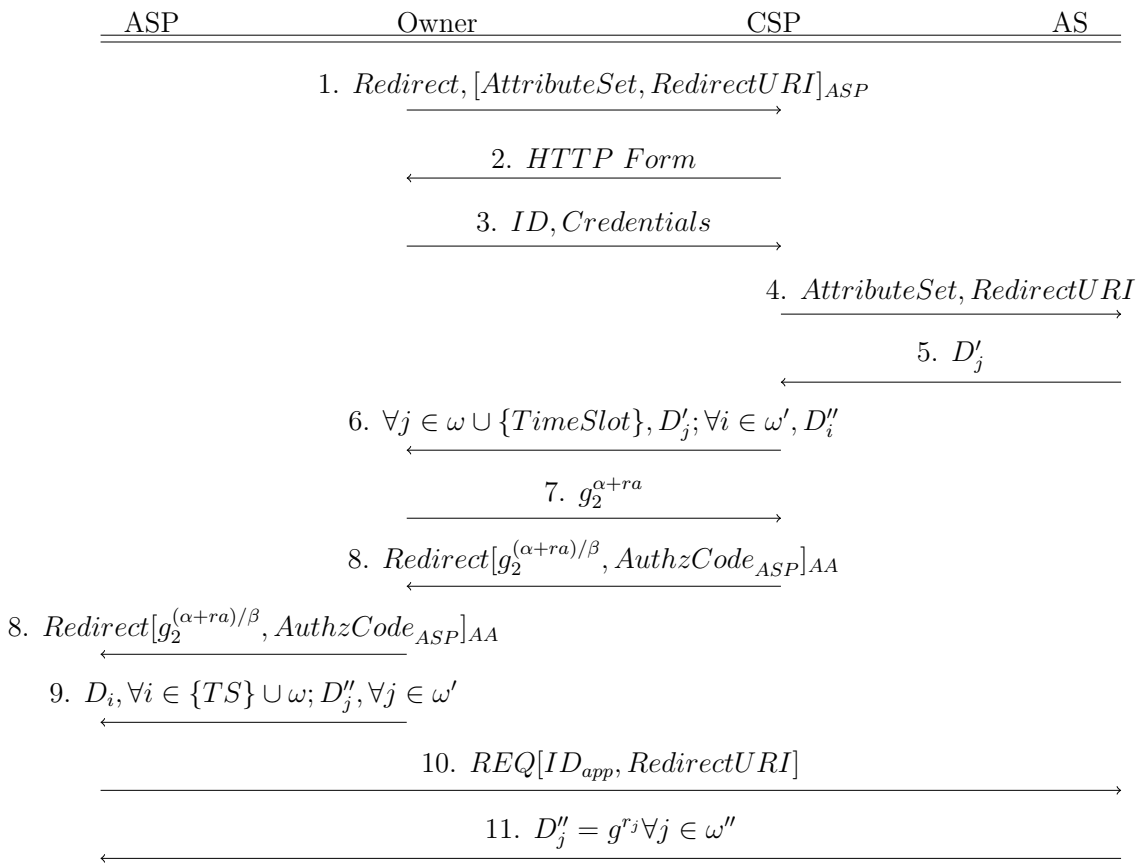


Figure 4.5: Token and Secret Key Issuing Flow

of redirection command and attribute set is given as follows.

$$\begin{aligned} Redirect1 =& [AttributeSet, RedirectURI]_{ASP} \\ \text{where } AttributeSet =& \{AppStore, AppID, AppExpireDate, \\ & AppFunctionality, AppAuthor, AppAddress\}. \end{aligned} \tag{4.18}$$

The subscript ASP means that content in the square brackets are digitally signed by ASP.

2. CSP sends owner the authentication $HTTP\ Form$.

3. Owner fills her ID and credentials and submits the form to authenticate herself to CSP.

4. If the authentication succeeds, CSP generates an authorization code $AutzCode$ (a nonce) and sends AS a command including ASP attribute set and redirect URI to request for the partial application key components from AS. The request command is shown as follows.

$$ReqAppAttPart1 = [AttributeSet, RedirectURI]_{CSP}$$

5. Application store retrieves applications attributes from $AttributeSet$, then $\forall j \in AttributeSet$, application store generates partial part-1 $D'_j = H(j)^{r_j}$ and replies it to CSP. The part-2 $D''_j = g_2^{r_j}$ is sent to ASP when ASP authenticate itself to AS in the later step.

6. The CSP, on the other hand, $\forall i \in \omega' \cup \{TimeSlot\}$ generates partial part-1 $D'_i = H(i)^{r_i}$ and part-2 $D''_i = g^{r_i}$. And CSP randomly selects $r \in \mathbb{Z}_q$. Then the part-1 and the part-2 signed by the CSP, g^r , and the partial part-1 $D_j{}'$ of ASP received earlier are sent to the owner.

7. Owner verifies whether D'_j, D''_j where $\forall i \in \omega' \cup \{TimeSlot\}$ are valid by computing bilinear pairing $e(D'_i, g) = e(D''_i, H(i))$. If the verification succeeds, the owner randomly chooses a, computes $g_2^{\alpha+ra}$ from a, g_2^r and the owner secret key g_2^α. Owner then replies to the CSP with the result $g_2^{\alpha+ra}$.

8. CSP generates the common part $D = g_2^{(\alpha+ra)/\beta}$ from the MSK β and $g_2^{\alpha+ra}$ received from owner. Hence, with ElGamal-like mask, we prevent CSP from knowing about owner secret key g_2^{α}, and owner from knowing CSP's master secret key β. CSP encrypts the common part D and the authorization code $AutzCode$ with ASPs public key and signs with the its private key, before sending the owner a redirect command $Redirect[g_2^{(\alpha+ra)/\beta}, AuthzCode_{ASP}]_{AA}$ to redirect the user-agent back to the consumer with the redirection URI received earlier.

9. The owner binds all partial key components by multiplying them with g_1^{ra} and sends all key components $D_i = g_1^{ra} D'_i, \forall i \in \omega \cup \{TimeSlot\}$ and $D_j = g_1^{ra} D''_j, \forall j \in \omega'$ to the application service provider.

10. The application service provider sends the authority a command

$$REQ[ID_{app}, RedirectURI]$$

to authenticate itself and to request the partial part-2 key components of application attributes.

11. If authentication succeeds, application store replies to the application service provider with the partial part-2 $D''_j = g_2^{r_j}, \forall j \in \omega''$ of application components.

Note that, the detail of how application service providers are authenticated by application store are beyond the scope. However, readers can get a general idea of this process through Google Accounts Authentication and Authorization [35].

4.6.3 File Access

The file access protocol is similar as OAuth 2.0 [4]. Also, compare to other protocol procedures, it is rather easy and simple.

1. The application accesses archive files by presenting the access token to the CSP.

2. The CSP validates the access token and ensures that it has not expired and that its scope covers the requested resource.

3. If the access token is valid, then CSP will transmit the required archive file to ASP.

4.6.4 TimeSlot Synchronization

The procedure of time slot synchronization is presented below.

1. Assume owner updates the plain data at time t, he chooses a new random value s_t and computes $e(g,g)^{\alpha\Delta_s}$ using its $OPK = e(g,g)^{\alpha}$. Owner now sends $e(g,g)^{\alpha\Delta_s}$ and s_t to the CSP.

2. CSP computes the new $TimeSlot$ share as $P_{TS}(0)_t = P_{TS}(0)_{t-1} + s_t$. Then the new cipher-text components for the new time slot can be computed as $C_{TS_t} = g^{P_{TS}(0)_t}$ and $C'_{TS_t} = H(TimeSlot)^{P_{TS}(0)_t}$ where $TimeSlot$ is the string of the t_{th} time slot. Also CSP updates $C = h^{s_t}$ from $MPK = g^{\beta}$.

3. Owner computes $e(g,g)^{\alpha\Delta_s}$ using its $OPK = e(g,g)^{\alpha}$ and send it to the CSP.

4. CSP replaces two cipher-text components C_{TS_t}, C'_{TS_t} and $C = h^{s_t}$ with the received components according to the current time slot. CSP also computes the new cipher $\widetilde{C}_t = \widetilde{C}_{t-1} \cdot e(g,g)^{\alpha\Delta_s}$.

4.7 Difference Between Fuzzy Authorization and Other Solutions

Fuzzy Authorization (FA) maintains the confidentiality of data with symmetric encryption and encrypts the symmetric key with modified CP-ABE. Integrity tags are computed so that it is convenient for data owner and authorized parties to check the integrity without any TPAs. Especially, FA provides an scalable, efficient and flexible access control by exploiting the modified CP-ABE to adapt to the cloud storage environment. Requiring no third authority parties, FA is totally practical and feasible in the industrial world for all the entities involved already exist. Moreover, we enable the fuzziness of authorization

by transforming secret reconstruction to GRS decoding to take advantage of GRS error correction ability. Through assembling fuzziness functionality into system, we enhance scalability and flexibility at the price of minor security loss.

Chapter 5

Security Analysis

In this section, our system is analyzed from perspectives of internal and external adversaries. 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 to external adversaries, they may not run the protocol but try to launch general attacks to compromise the security of data. We will first give security analysis for internal adversary models provided that adversaries can get the cipher-text $CT = \langle \mathcal{T}, \widetilde{C} = m \cdot e(g_1, g_2)^{\alpha s}, C = h^s, \forall y \in \mathcal{Y} : C_y = g_2^{q_y(0)}, C'_y = H(att(y))^{q_y(0)} \rangle$. According to our access structure, in order to recover the top secret s, the adverse party has to recover $P_{TS}(0), P_f(0)$ and $P_a(0)$ in the first place.

5.1 CSP Tries To Illegally Access or Modify Owner's Plain Data

Without any collusion with other parties, cloud server is able to get the $TimeSlot$ and file attributes, that is $\{TimeSlot\} \cup \omega'$. For any $t \in \{TimeSlot\} \cup \omega'$, cloud server can obtain $H(t)^{r_t}$ and $g_2^{r_t}$, however, not $g_2^{r_a} H(t)^{r_t}$.

1. Scenario 1. Let the target secret be $P_{TS}(0)$ and the threshold value is $K = 1$, then we can settle set $\widetilde{W} = \{TimeSlot\}, \widetilde{W'} = \widetilde{W}$ and $\widetilde{W''} = \emptyset$ where $\widetilde{W}, \widetilde{W'}$ and $\widetilde{W''}$ are

notations defined in the d-BDHE assumption. Clearly, $|\widetilde{W}''| = 0 < K = 1$ satisfies and therefore the d-BDHE assumption holds.

2. Scenario 2. Let the target secret be $P_f(0)$ and the threshold value of F-subtree's root node be K, then $\widetilde{W} = \omega', \widetilde{W}' = \widetilde{W} = \omega'$ and $\widetilde{W}'' = \emptyset$. Again, inequity $|\widetilde{W}''| = 0 < K$ satisfies and our assumption still holds in this scenario. Since CSP has no information about application attributes, it is impossible for it to guess $P_a(0)$.

3. Scenario 3. Setting our target secret as the top secret s and the threshold value $K = 3$, then $\widetilde{W} = \{1, 2, 3\}$ in which the set elements are indexes of children of the root node. Also, $\widetilde{W}' = \emptyset$ and $\widetilde{W}'' = \emptyset$ and $|\widetilde{W}''| = 0 < K = 3$. As a consequence, it remains difficult for CSP to get the top secret s.

5.2 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 owner has updated the file. The second occasion is more severe since the ASP holds indirect file attributes shares.

1. Scenario 4. In the first occasion, TimeSlot is not known by ASP, neither attributes of F-subtree. 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 our target secret to $P_{TS}(0)$ and $P_f(0)$ will leads us to Scenario 1 and Scenario 2 separately. If ASP sets the target secret as $P_a(0)$ and the threshold of A-subtree, a positive integer, to K. Then we will have $\widetilde{W} = \omega', \widetilde{W}' = \widetilde{W}$ and $\widetilde{W}'' = \emptyset$ with $|\widetilde{W}''| = 0 < K$. Once again, the d-BDHE assumption holds.

2. Scenario 5. In the second occasion, not only sub-trees of file attributes and application attributes are acquainted to ASP, but also $P_f(0)$ and $P_a(0)$. Even though, ASP could forge the random exponent r_t for TimeSlot attribute and get $H(TS)^{r_t}$, $g_2^{r_t}$, there is no way for ASP to guess $g_1^{ra}H(TS)^{r_t}$. Setting the target secret as the top secret s and threshold value $K = 3$, $\widetilde{W} = \{1, 2, 3\}, \widetilde{W}' = \{TimeSlot\}$ and $\widetilde{W}'' = \{2, 3\}$ where $|\widetilde{W}''| = 2 < K = 3$. Hence, the d-BDHE assumption applies to this scenario as well.

5.3 AS Tries to Access Owner's Data Illegally

It is clear to find that attributes exposed to AS are application attributes and thus $\widetilde{W} = \omega''$. Similar as we reduce the adverse CSP model to d-BDHE assumption, we can also reduce this adverse model to our assumption. Hence it is impractical for AS to recover the top secret s.

5.4 Owner Propose Tokens to Access Other Owners' File

A vicious owner may either pretend to be an innocent owner to issue tokens or she may fabricate the tokens in place of another owner. The former case is unlikely for the vicious owner has to authenticate herself to CSP. As to the latter case, the vicious owner may fabricate the partial components of indirect secret shares attached with file attributes and application attributes and multiply them with her own $g_1^{ra'}$. Alternatively put, for any $t \in \omega(\omega$ 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 ominous owner will get $e(g_1, g_2)^{ra's}$ 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 $e(g_1, g_2)^{ras}$, the problem will be reduced to a discrete logarithm problem and hence the fabrication is unsuccessful.

Chapter 6

Implementation of Fuzzy Authorization

In this chapter, we present an implementation of FA protocol and their performance. The implementation environment and parameters chosen for communication among four parties are first introduced in Section 6.1. Some optimizations of the implementation is presented in Section 6.2. Comparisons with Fuzzy IBE adapted in authorization is demonstrated in Section 6.3. Then measurements of performance is demonstrated in Section 6.4.

6.1 Parameter Selection and Simulation Environment

6.1.1 Pairing Implementation

Our implementation uses symmetric bilinear pairing which was implemented with pairing-based cryptography (PBC) library [36] from Stanford University. 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 $\mathbb{Z}_q$ where

q is the order of group $\mathbb{G}$. Using a computer with 4 Intel(R) Core(TM) i3-2130 CPUs running at 3.40GHz.

6.1.2 Implementation of FA Protocol with OMNETPP

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 project, we adopt *handleMessage()* function due to its convenience of co-working with library PBC.

However, in our implementation, simulation time does not elapse in the function, in other words, bilinear pairing and other relative computing time will not be counted in the FA protocol, the experiment time we collected is simply the protocol running time.

6.1.3 Parameter Selection for Communication

FA protocol mainly facilitates user 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 for simulation, we monitored communications between a smart phone and online websites in real life with WebSitePulse [37] a tool used to monitor internet communications. Depending on the websites smart phone accessed and the situation of WiFi to which smart phone connected, connection time and responding time varies. The effective upload bandwidth of the WiFi is 500Kbps and download speed is 65KBps. Under this circumstance, and after one thousand test 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 6.1 Bandwidth of cloud storage provider is unlimited just as most cloud storage providers set in real life [38] and so as bandwidth of application store. Upload bandwidth

Table 6.1: Response Delay Parameters

Dropbox	Chrome Web store	Owner Device (Android)	pdfmerge
15ms	10ms	49ms	20ms

of owner is 500Kpbs and download bandwidth is 65KBps, the same as the parameters of real life smart phone communication.

6.2 Optimizations

For error correction, the original decoding algorithm is introduced in Chapter 3. Before each interpolation, a set $\mathcal{Y}'_s$ of K indexes is chosen for all possibilities. Let k range from 1 to $\binom{N}{K}$ and $\mathcal{Y}'_{s_k}$ be the kth set. Combination in lexicographical order algorithm [39] is used and further optimization could be done on top of it. In lexicographical order combination, the next combination is constructed based on current combination and the difference between them is only one component. So instead of conducting $\binom{N}{K}$ complete interpolations, the optimized procedure will perform the first complete interpolation and rest $\binom{N}{K} - 1$ partial interpolations.

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

$$w_{k,j_u}(x) = e(g_1,\ g_2)^{ra \prod_{\forall i \in \mathcal{Y}'_{s_k}, i \neq j_u} \frac{(x-i)P_f(j_u)}{j_u - i}}. \tag{6.1}$$

Then we will obtain set $\mathcal{W}_k = \{w_{k,1}(x), w_{k,2}(x), ..., w_{k,K}(x)\}$ where $w_{k,j_u}(x)$ is defined as equation (6.1).

Denote the complementary set of $\mathcal{Y}'_{s_k}$ as $\mathcal{Y}^C_{s_k} = \{1, 2, ..., N\} \setminus \mathcal{Y}'_{s_k}$. An index $i_{old} \in \mathcal{Y}'_{s_k}$ is the old index to be replaced by the new index $i_{new} \in \mathcal{Y}^C_{s_k}$. Then the kth set $\mathcal{W}_k$ is updated to $\mathcal{W}_{k+1}$ as following:

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

$$w_{k+1,j_u} = w_{k,j_u}; \tag{6.2}$$

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

$$w_{k+1,j_u} = w_{k,j_u}^{\frac{i_{new}-j_u}{i_{old}-j_u}}. \tag{6.3}$$

By keeping $\mathcal{W}_k$ up-to-date, the interpolation will always be

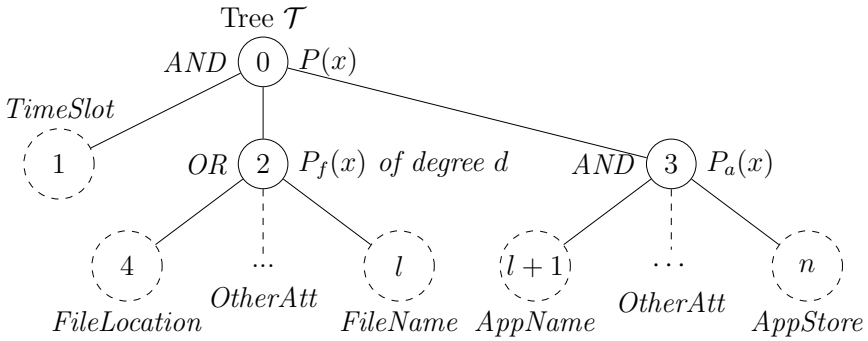
$$e(g_1,\ g_2)^{raP_f(x)} = e(g_1,\ g_2)^{ra\ \sum_{u=1}^{K} w_{k,j_u}}, u = 1, 2, ..., N. \tag{6.4}$$

Before the optimization, for each set $\mathcal{Y}'_{s_k}$, interpolation will cost $1 + 2(K-1)$ exponential operations on the element from group $\mathbb{G}_T$ and $\binom{N}{K}[1 + 2(K-1)]$ exponentiation overall. The optimization reduced $1 + 2(K-1)$ exponential operations to 1 exponential operation and the overall number of exponential operation is reduced from $\binom{N}{K}[1 + 2(K-1)]$ to $2(K-1) + \binom{N}{K}$.

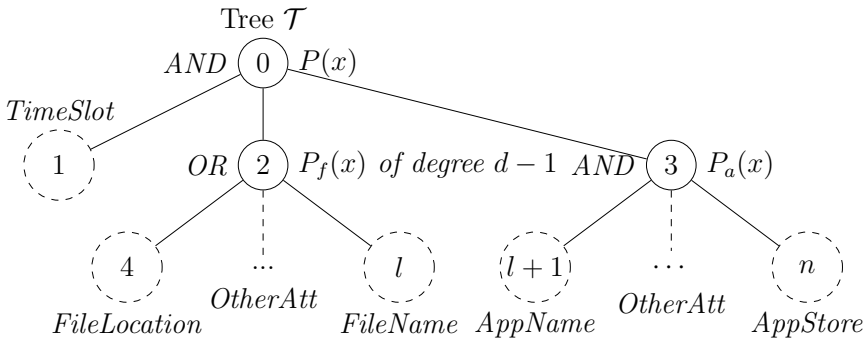
Another optimization can be adopted when performs the decryption over the root of sub-tree where the checking nodes are added, e.g., F-subtree in our case. Instead of computing the exponential polynomial $P_f(x)$, unknown x can be replaced by node index number. Thus the interpolation result is a potential indirect share. Replacing x with indexes of root's children nodes in turn, a set of new indirect secret shares will be obtained. Instead of choosing the most frequently occurring polynomial, the advantage of parity check matrix $\mathcal{H}$ could be used. For each new set of share components obtained, equations (3.7) and (3.8) can be applied to check whether they are the correct share components. If (3.7) and (3.8) are satisfied for a certain set of potential share components, stop interpolation and set the unknown x to 0 to obtain $e(g_1, g_2)^{raP_f(0)}$.

6.3 Fuzzy IBE Adapted in Cloud Storage Authorization

In order to compare the performance of FA, we apply two simple methods derived from Fuzzy IBE proposed in [19]. The first solution of Fuzzy IBE is referred as Fuzzy IBE1 and the second solution is denoted as Fuzzy IBE2.



(a) $P_f(x)$ of degree d



(b) $P_f(x)$ of degree $d-1$

$\vdots$

Figure 6.1: Access Trees of Fuzzy IBE1

6.3.1 First Solution of Fuzzy IBE

As shown in Fig. 6.1, for each file, owner creates multiple access trees with distinct threshold values of F-subtrees. Different threshold values indicates different degrees of $P_f(x)$ attached with F-subtree. A smaller threshold gives us larger error-tolerant ability. Therefore, for each access tree owner has to reconstruct the polynomial $P_f(x)$ and compute the corresponding $P_f(i)$ where i is the node index number.

Then owner encrypts the symmetric key KE under these different trees to obtain different cipher-texts. As a result, different cipher-texts matched with different access

trees are stored together in the cloud server. At the beginning of authorization, owner determines the error-tolerant ability and assigns one of the cipher-texts that will be sent to ASP. Since ASP has no idea about the file attribute set, there is no way for ASP to perform the *satisfying an access tree* procedure over F-sub-tree. Hence owner has to transmit the file attribute set of application to ASP as well.

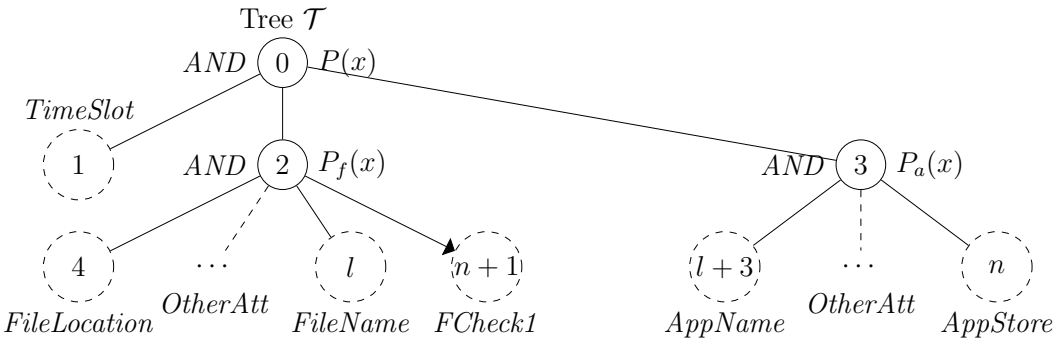
After receiving the file attribute set, cipher-text and the secret key, ASP first applies the *satisfying an access tree* procedure to determine which key components of the attributes are correct and can be used to perform decryption. Then, by making use of the Decryption procedure [19], ASP can obtain K correct indirect shares and recover the top secret, finally get symmetric key KE. If the *satisfying an access tree* procedure fails, ASP quits decryption and requests the secret key again.

Fuzzy IBE1 is simple, however a large amount of extra space and computation of encryption are needed. In addition, owner has to transmit ASP the file attribute set. Moreover, ASP also needs to run *satisfying an access tree* procedure to select the correct key components so as to decrypt.

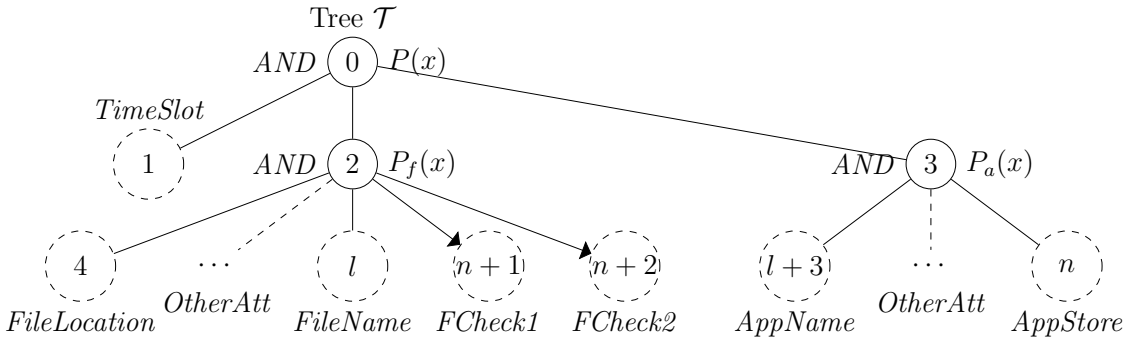
6.3.2 Second Solution of Fuzzy IBE

The second solution is that owner reserves some default attributes in the F-subtrees of all the files and maintains the threshold values. This solution is denoted as Fuzzy IBE2 shown in Fig. 6.2. By increasing the number of these default attributes, owner enhances the tolerance of error. The additional nodes' values, i.e., $P_f(i)$ where i is the index number of additional nodes, are computed with polynomial $P_f(x)$. Unlike Fuzzy IBE1 above, the solution two does not require the reconstruction of the polynomials attached with extra access trees. There is only one access tree for each file. Still the cipher-text components of the additional default nodes are computed and inserted into the cipher-text.

Since ASP is not aware of the file attributes, owner has to send application the file attributes along with the secret key, which is the same as Fuzzy IBE1. Before performing decryption with the secret key, ASP needs to carry out *satisfying an access tree* procedure with the received file attributes to determine which attributes are matched with the



(a) Add One Default Node



(b) Add Two Default Nodes

$\vdots$

Figure 6.2: Access Trees of Fuzzy IBE2

attributes attached with tree leaf nodes. Founded on the matched attributes, the corresponding components of the secret key are selected to do decryption.

6.3.3 Comparisons of FA to Fuzzy IBE1 and Fuzzy IBE2

Similar to the second solution, our fuzzy authorization scheme adds additional attributes into F-subtree. However, Fuzzy IBE2 requires owner to send the attribute set of file which may result leakage of owner's privacy. Also in Fuzzy IBE2, ASP has to perform *satisfying access tree* procedure to determine which indirect shares can be utilized. According to the property of Maximum Distance Separable (MDS) code, by adding $2e$ additional attributes in the tree, where e is the maximum number of errors that could be tolerated, fuzzy authorization has the ability to check and correct errors. Then fuzzy authorization is able to perform error correction when some of the attributes are not matched. When every indirect shares are correct, the reconstruction of top secret key can be performed on arbitrarily K out of N shares.

Thus unlike the two solutions derived from Fuzzy IBE, FA begins error correction at the indirect share level while the two former solutions select the right key components at attributes level. Therefore, fuzzy authorization avoids owner from sending file attributes to ASP and eliminating the necessity of carrying out the *satisfying an access tree* procedure by ASP. More importantly, FA avoids the privacy leakage of data owner.

The requirements of the three solutions are summarized in Table 6.2. In Table 6.2, notations are given.

- R_1 represents the operation that owner needs to send file attributes to ASP.
- R_2 denotes that ASP needs to performs procedure of *satisfying access tree*.
- R_3 stands for the leakage information of owner.
- R_4 means the error checking operation should be performed.
- R_5 means that error creation operation is required.

Table 6.2: Comparison of FA, Fuzzy IBE1, and Fuzzy IBE2

Properties	FA	Fuzzy IBE1	Fuzzy IBE2
R_1	No	Yes	Yes
R_2	No	Yes	Yes
R_3	No	Yes	Yes
R_4	Yes	No	No
R_5	Yes	No	No

6.4 Performance Measurements

6.4.1 Time Consumption

For single computation, the time is collected as following.

1. On average, it costs 1.14ms to compute bilinear pairing.
2. It costs 1.51ms and 0.14ms on average to complete exponentiation in $\mathbb{G}$ and $\mathbb{G}_T$ receptively.
3. Adding and multiplication operations cost 0.001ms and 0.09ms which are relatively small.
4. For one authorization operation of 3 different files with one error tolerance, the average overall time of our simulation is 1.187s.

Compare to other authorization schemes, FA utilizes error checking and correction. The simulation results show that error checking and correction is not very time consuming. The average time consumption of error checking and correction is shown in the following table. Table 6.3 shows the time consumption of error checking and correction.

As to the transmission of file attribute set, at east one *round trip time* (RTT) is needed. In the most commonly used 3G and 4G networks, the average RTT of these networks are around or over 100ms [40]. Compare with the communication overhead cost by transmission of file attribute set, error checking and correction is more efficient.

Table 6.3: Time Consumption of Error Checking and Correction

Time Consumption	Attribute Number in F-subtree	Error Number
$47.45ms$	6	1
$56.02ms$	4	2
$68.63ms$	8	1
$79.91ms$	6	2

6.4.2 Extra Space Consumption

In the access tree, each leaf node is attached with an attribute y for which two cipher components $C_y = g_2^{P_y(0)}, C'_y = H(y)^{P_y(0)}$ must be added into the cipher-text. Assume total number of leaf nodes of the access tree is n, and the number of F-subtree leaf nodes is $\frac{n}{2}$. Let l be the number of archives that could be decrypted with the same KE and e be the maximum number of errors that can be tolerated. In our simulation, $n = 16$ and l ranges from 1 to 10. Since *FileName* and *FileLocation* are the two attributes that most likely to be different, two typical values of $e = 1$ and $e = 1$ are simulated.

For Fuzzy IBE1, fuzziness of authorization can be achieved by changing the threshold value of F-subtree. Then polynomial $P_f(x)$ and values of leaf nodes have to be recomputed. As a consequence, the cipher components for F-subtree leaf nodes must be updated accordingly. Then at least $2 * \frac{n}{2}$ extra elements from group $\mathbb{G}$ are required. The extra storage required is referred as Fuzzy IBE1 in Fig. 6.3.

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

In fuzzy authorization, we insert checking nodes in the access trees and compute extra cipher components. According to the property of MDS code, for error correction ability of e, at least $2e$ checking nodes are required. The number of extra elements from group $\mathbb{G}$ is $4e$. The storage consumption of these three solutions are demonstrated in Fig. 6.3.

From Fig. 6.3, we can observe that extra storage consumption of fuzzy authorization is always less than that of Fuzzy IBE1. In addition, when $e = 1, k < 10$ and $e = 2, k < 6$,

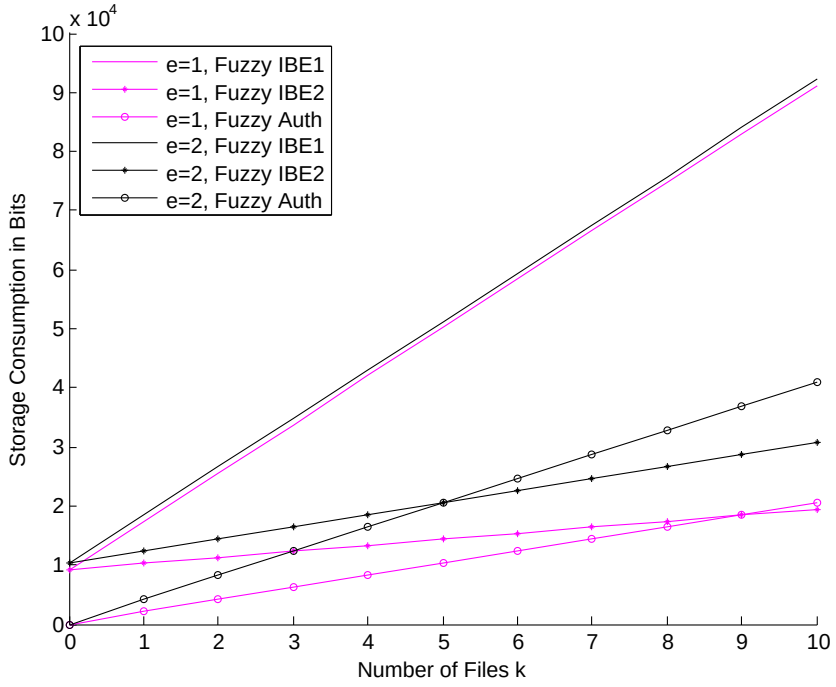


Figure 6.3: Comparison of Storage Consumption

fuzzy authorization has an advantage in storage consumption than Fuzzy IBE2 as well.

6.4.3 Revocation

Currently, most authorization schemes utilize manually revocation. As the backgrounds of owners vary large and for a less-cared owner, he or she may easily forget the revocation. We assume that once owner remembers, he or she will revoke. Therefore, based on Ebbinghaus Forgetting Curve, the probability of revocation failure is demonstrated in Fig. 6.4. Assume owner updates the original data at time t_{change}, then in FA, the non-revocation probability before t_{change} is 100% and after t_{change} is 0%. As manifested in Fig. 6.4, the uncertainty of human brain may result in higher probability of failure while revocation in fuzzy authorization is more determinate.

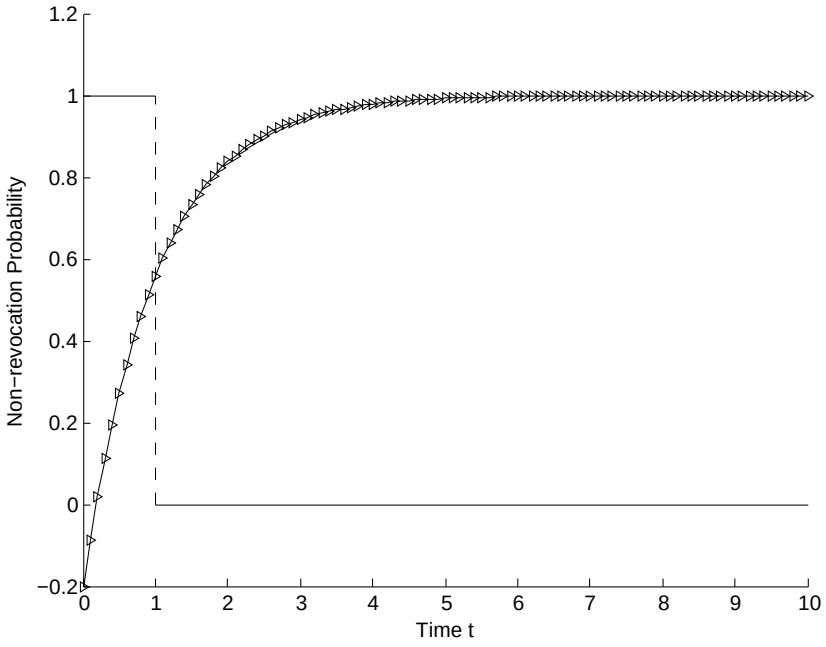


Figure 6.4: Non-revocation Probability of Manually and Fuzzy Authorization

6.4.4 Time Cost for Protocol Procedures

Since the response delay of each parties varies, the time consumption of each protocol procedure varies as well. By keeping the response delay of one party as a variant and fixing the delay response of the other three parties, we show how one party's response delay affects the overall time consumption.

Time Consumption of Service Request Protocol

The service request protocol basically contains two steps. Owner send a request to ASP asking for service and ASP responses with a redirect command that redirect owner to cloud storage provider. So the overall procedure contains propagation delay from owner to ASP, ASP to owner and owner to cloud storage provider and also the response delay between owner and ASP.

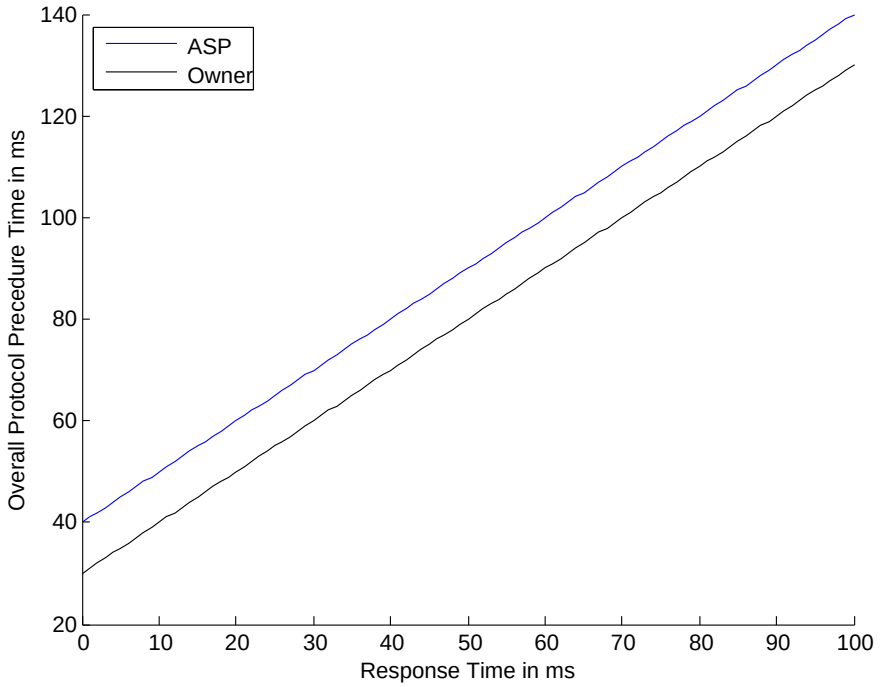


Figure 6.5: Time Consumption of Service Request Protocol

So there are two major factors affect the overall time. One is the response delay of ASP and the other is response delay of owner. Through testing with WebSitePulse, we fix the response time of owner as 40ms when set the response delay of ASP as a variant. In Fig. 6.5, blue line shows that the overall time consumption starts at 40ms and is linearly increased with gradient one as the response delay ascending. In Fig. 6.5, the black line indicates the entire time consumption growth based on the increase of the response delay of owner. The black line shows that, the entire time consumption grows with gradient one and starts at 30ms.

Time Consumption of Token Issuing Protocol

Token issuing protocol is much more complex and involves four parties. So there are at least four major factors that influence the overall time. Let us fix the response delay of AS as 20ms and response delay of CSP as 50ms where these two factors are not variants. We

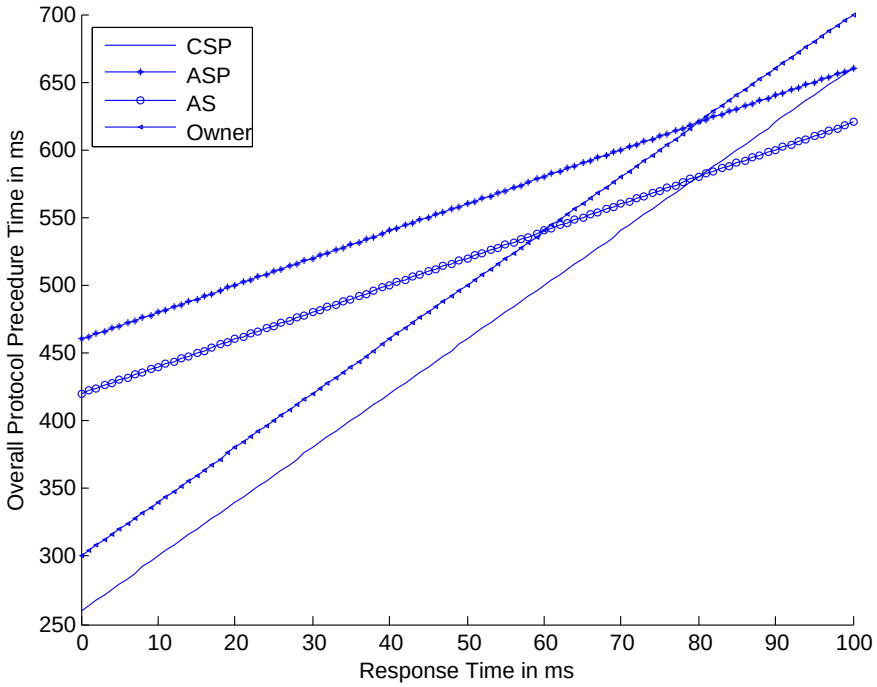


Figure 6.6: Time Consumption of Token Issuing Protocol

shall take a look at the Fig. 6.6 and analyze how each response delay impacts the entire time of the protocol.

The plain blue line shows how the response delay impacts the overall time consumption. Blue line with asterisks demonstrates the way the response delay of application service provider impacts the overall time. Blue line embedded with circles indicates response delay of application store influences the overall time consumption. The way how response time of owner affects the overall time is shown with the blue line embedded with triangles. For simplicity of notation, let us denote the lines as lines 1, 2, 3 and 4 respectively.

As demonstrated in Fig. 6.6, line 1 indicates overall time increases with gradient of four starting from 260ms as the response time of CSP ascending. Line 2 shows that overall time initiating from 460ms and grows with gradient of two. Line 3 demonstrates that overall time ascends along with the increase of response delay of application store. Line 4 shows that increases of the response delay of owner influences the overall time to ascend four

times. Also, it is obvious to notice that the response delay of owner and CSP cast larger influence to the overall time consumption because the increase gradients are larger.

Time Consumption of File Access Protocol

The file access protocol is relatively simpler where ASP request for the archive file and show the access token to CSP. CSP examines the expiration time and access scope of the access token before sending file to ASP. So the overall time contains propagation and the response delay from ASP to CSP and the transmission time of the archive file.

As a result, there are two major factors affect the overall time. They are the response delay and propagation time of ASP and the response delay and propagation time of CSP. In Fig. 6.7, the black line indicates the entire time consumption growth based on the increase of the response delay of ASP. The black line shows that, the entire time consumption grows with gradient one and starts at 30ms. The blue line of CSP, on the other hand, display that entire time consumption grows with gradient one and starts at 30ms.

6.4.5 Algorithm Complexity Analysis

Assume the number of files that about to be authorized to ASP is k and the number of error that can be tolerated is e. We denote N as the number of leaf nodes in F-subtree, $O(Exp)$ the computing time of exponentiation of group elements, and $O(Mul)$ be the time of multiplication of two group elements.

Ciphertext Computing

In Fuzzy IBE adapted in authorization 1, access trees with another threshold value of all the files that are about to be authorized to ASP are created. Furthermore, the cipher-text components matched with F-sub-tree are recomputed. Let the complexity of computing the attributes of A-Fubtree and $TimeSlot$ be $O(Basic)$. So the complexity of Fuzzy IBE adapted in authorization 1 is $2NK \cdot O(Exp) + O(Basic)$. As to Fuzzy IBE adapted in authorization 2, there are e extra nodes added in the F-sub-tree and then the complexity

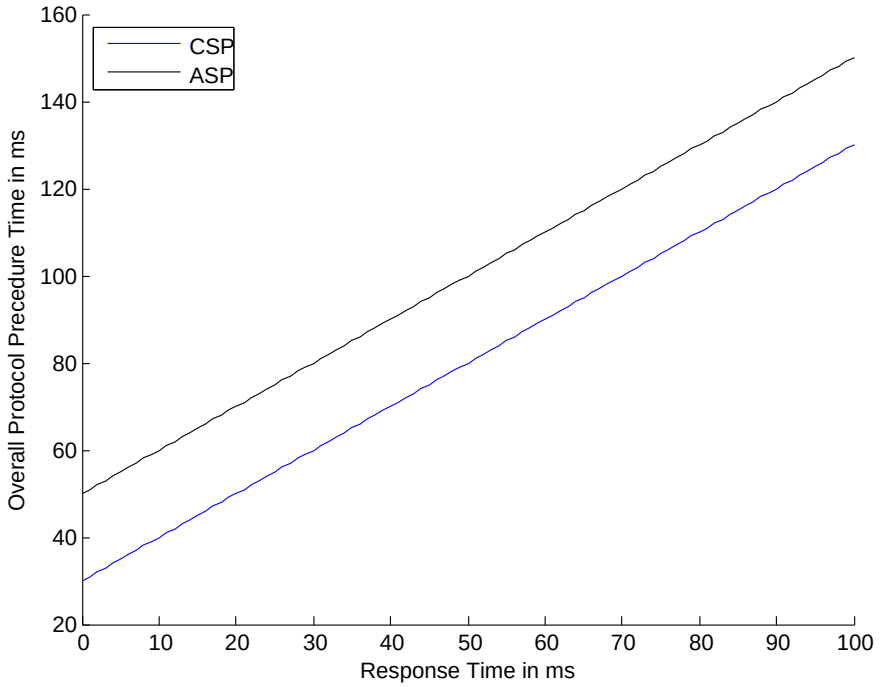


Figure 6.7: Time Consumption of File Access Protocol

of computing is $2e \cdot O(Exp) + O(Basic)$. Similarly, Fuzzy Authorization's computing complexity is $4e \cdot O(Exp) + O(Basic)$. The computing complexity of three schemes is shown in Table 6.4.

Table 6.4: Ciphertext Computing

Schemes	Complexity of ciphertext computing
Fuzzy IBE1	$2NK \cdot O(Exp) + O(Basic)$
Fuzzy IBE2	$2e \cdot O(Exp) + O(Basic)$
FA	$4e \cdot O(Exp) + O(Basic)$

Table 6.5: Secret Key Issuing Complexity

Schemes	Complexity of secret key issuing
Fuzzy IBE1	$O(Fundamental)$
Fuzzy IBE2	$2d \cdot O(Exp) + d \cdot O(Mul) + O(Fundamental)$
FA	$4d \cdot O(Exp) + 2d \cdot O(Mul) + O(Fundamental)$

Secret Key Issuing Complexity

Let $O(Fundamental)$ be the computing complexity of the original access tree. In Fuzzy IBE1, there is no extra computing while in Fuzzy IBE2, extra $2d \cdot O(Exp) + d \cdot O(Mul)$ computing is needed. Besides the fundamental computing, in FA, extra $4d \cdot O(Exp) + 2d \cdot O(Mul)$ computing is needed. The computing complexity of three schemes is shown in Table 6.5.

Decryption Complexity

The main procedure of decryption is to perform interpolation. In both Fuzzy IBE1 and Fuzzy IBE2, decryption needs to call *satisfying an access tree* procedure to determine which of the cipher-text components and secret key components are selected to perform interpolation. The number of leaf nodes in F-subtree is N in Fuzzy IBE1 and $N + e$ in Fuzzy IBE2. Hence the complexity of *satisfying an access tree* of Fuzzy IBE1 is N string comparisons and of Fuzzy IBE2 is $N+e$ string comparisons. In fuzzy authorization, instead of calling *satisfying an access tree* procedure, the error checking procedure is required which has a cost of $N(N-K) \cdot O(Exp) + (N-1)(N-K) \cdot O(Mul)$. If there are errors exist, correction procedure is then conducted. The complexity of error correction is $[2(K-1) + \binom{N}{K}]O(Exp) + \binom{N}{K}K \cdot O(Mul)$. Let $O(DecInter)$ denote the rest interpolations for decryption. Then the complexities of decryption of three schemes are displayed in Table 6.6.

Table 6.6: Decryption Complexity

Schemes	Complexity of secret key issuing
Fuzzy IBE1	N string comparisons
Fuzzy IBE2	$N + e$ string comparisons
FA	$[2(K-1) + \binom{N}{K}]O(Exp) + \binom{N}{K}K \cdot O(Mul)$

Chapter 7

Conclusion and Future Work

In this chapter, we present the conclusion in Section 7.1 and in Section 7.2, we demonstrate the future work.

7.1 Conclusion

In this thesis, we propose a new authorization scheme, fuzzy authorization, which carries out a flexible file sharing between owner who stores her data in one cloud party and applications who registered within another cloud party. In addition, in fuzzy authorization, the confidentiality of data is maintained through symmetric encryption and attribute based encryption; the integrity of data is checked with integrity tag by owner or ASP; and the access control is securely implemented with modified CP-ABE.

By tampering one of the components of secret key to be incorrect, the simulation of FA protocol suggests that our authorization scheme successfully corrects the unmatched indirect secret share, recovers the top secret and performs the decryption for KE. FA's self-error-checking ability eliminates the requirement of sending the file attribute to ASP and error-correcing ability omits necessity of performing *satisfying the access tree* procedure are proved by our simulation at a minor tradeoff in efficiency. Further more, the simulation indicates that with the update of TimeSlot attribute, our authorization scheme

automatically invalidates the authorized reading right from ASP. Comparing to Fuzzy IBE1 and Fuzzy IBE2, the simulation results show that our solution FA reduces the storage consumption when $e = 1$ 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 scheme is feasible and acceptable.

7.2 Future Work

While this thesis addresses the reading right authorization on cloud storage, our future work will aim to resolve the writing right accreditation. Since the writing authorization accreditation other parties to change owner's data, trust between owner and other parties must be built on a more rigorous authentication.

Also, we adopt the lazy revocation for our scheme which requires the owner to choose random increase value s_t and compute $e(g, g)^{\alpha s_t}$ at time t when he updates the archive file. The cryptographic data structure is updated so as to revoke. Compare to revocation at the beginning of equivalent time slot, lazy revocation keeps owner from extra re-encryption. Unfortunately, lazy revocation slightly lowers the security [23]. In the future work, we will work on another revocation scheme that eliminates the updating of cryptographic data structure and enhance the security.

Besides, for we are using the original GRS decoding method, the complexity of error correction grows exponentially when the number of file attributes increases. A more efficient way to detect and correct the errors should be considered and tested in later work.

References

[1] http://www.chromeosapps.org/, 2013.

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

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

[4] E. Hammer-Lahav, D. Recordon, and D. Hardt, "The oauth 2.0 authorization protocol," *Network Working Group Internet-Draft*, 2011, available at http://tools.ietf.org/html/rfc6749.

[5] A. Tassanaviboon and G. Gong, "Oauth and abe based authorization in semi-trusted cloud computing," in *Data intensive computing in the clouds - DataCloud-SC '11, second international workshop, Proceedings.* ACM, 2011, pp. 41–50.

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

[7] D. Boneh and X. Boyen, "Efficient selective-ID secure identity based encryption without random oracles," in *Advances in Cryptology—EUROCRYPT 2004*, ser. Lecture Notes in Computer Science, vol. 3027. Springer Berlin Heidelberg, 2004, pp. 223–238, available at http://www.cs.stanford.edu/~xb/eurocrypt04b/.

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

[9] J. Xu, E.-C. Chang, and J. Zhou, "Leakage-resilient client-side deduplication of encrypted data in cloud storage," Cryptology ePrint Archive, Report 2011/538, 2011, http://eprint.iacr.org/.

[10] G. Davida and Y. Frankel, "Efficient encryption and storage of close distance messages with applications to cloud storage," in *Cryptography and Security: From Theory to Applications*, ser. Lecture Notes in Computer Science, D. Naccache, Ed. Springer Berlin Heidelberg, 2012, vol. 6805, pp. 465–473. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-28368-0_29

[11] S. D. C. di Vimercati, S. Foresti, S. Jajodia, S. Paraboschi, G. Pelosi, and P. Samarati, "Encryption-based policy enforcement for cloud storage," in *Proceedings of the 2010 IEEE 30th International Conference on Distributed Computing Systems Workshop*. ACM, 2010, pp. 42–51.

[12] 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, pp. 362–375, February 2013.

[13] Y. Zhu, H. Wang, Z. Hu, G.-J. Ahn, H. Hu, and S. S. Yau, "Dynamic audit services for integrity verification of outsourced storages in clouds," in *IEEE Transactions on Services Computing*. IEEE, 2011, pp. 227–238.

[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*. ACM, 2008, pp. 90–107.

[15] K. D. Bowers, A. Juels, and A. Oprea, "Hail: a high-availability and integrity layer for cloud storage," in *Proceedings of the 16th ACM conference on Computer and communications security*. ACM, 2009, pp. 187–198.

[16] A. Juels and B. Kaliski, "Pors: Proofs of retrievability for large files," in *ACM Conference on Computer and Communications Security*. ACM, 2007, p. 584597.

[17] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, and D. Song, "Provable data possession at untrusted stores," in *ACM Conference on Computer and Communications Security*. ACM, 2007, p. 598609.

[18] U. Lang, "Openpmf scaas: Authorization as a service for cloud & soa applications," in *Cloud Computing Technology and Science (CloudCom)*. IEEE, 2010, pp. 634–643.

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

[20] V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-based encryption for fine-grained access control of encrypted data," 2006, extended abstract to appear in ACM CCS 2006. This is the full version. vipul@cs.ucla.edu 13428 received 31 Aug 2006, last revised 7 Oct 2006. [Online]. Available: http://eprint.iacr.org/2006/309

[21] B. Waters, "Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization," Cryptology ePrint Archive, Report 2008/290, 2008, http://eprint.iacr.org/.

[22] ——, "Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization," in *Public Key Cryptography PKC 2011*, ser. Lecture Notes in Computer Science, vol. 6571. Springer Berlin Heidelberg, 2011, pp. 53–70.

[23] S. Zarandioon, D. D. Yao, and V. Ganapathy, "K2c: Cryptographic cloud storage with lazy revocation and anonymous accessn," in *Security and Privacy in Communication Networks*, ser. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, vol. 96. Springer Berlin Heidelberg, 2012, pp. 59–76.

[24] 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.

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

[26] S. T. Position, "Cloud data management interface," *Advancing Storage and Information Technology*, 2012, available at http://snia.org/sites/default/files/CDMI%20v1.0.2.pdf.

[27] S. Shepler, "Network file system (nfs) version 4 minor version 1 protocol," *Network Working Group Internet-Draft*, 2010, available at http://tools.ietf.org/html/rfc5661.

[28] D. Slamanig, "Efficient schemes for anonymous yet authorized and bounded use of cloud resources," in *Selected Areas in Cryptography*, ser. Lecture Notes in Computer Science, A. Miri and S. Vaudenay, Eds. Springer Berlin Heidelberg, 2012, vol. 7118, pp. 73–91. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-28496-0_5

[29] E. R. Berlekamp and L. R. Welch, "Error correction for algebraic block codes," U.S. Patent US 4 633 470 A, Septemper, 1983.

[30] D. Gorenstein, W. W. Peterson, and N. Zierler, "Two-error correcting bose-chaudhuri codes are quasi-perfect," *Information and Control*, vol. 3, pp. 291–294, 1960.

[31] I. Reed and G. Solomon, "Polynomial codes over certain finite fields," *Journal of Society for Industrial and Applied Mathematics*, vol. 8, pp. 300–304, 1960.

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

[33] S. Chatterjee and A. Menezes, "On cryptographic protocols employing asymmetric pairings the role of ψ revisited," Cryptology ePrint Archive, Report 2009/480, 2009, http://eprint.iacr.org/.

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

[35] "Google accounts authentication and authorization," https://developers.google.com/accounts/docs/OAuth2Login#registeringyourapp, April 2013.

[36] B. Lynn, *PBC Library Manual*, http://crypto.stanford.edu/pbc/manual.pdf, 2006.

[37] "Website test tools," http://www.websitepulse.com/help/tools.php, 2013.

[38] http://support.justcloud.com/question/205/is-there-a-bandwidth-limit, justCloud.

[39] C. J. Mifsud, "Algorithm 154: Combination in lexicographical osrder," *Commun. ACM*, vol. 6, no. 3, p. 103, Mar. 1963. [Online]. Available: http://dx.doi.org/10.1145/366274.366309

[40] 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," *UMass Technical Report*, 2012.

[41] C. Cocks, "An identity based encryption scheme based on quadratic residues," in *Cryptography and Coding*, ser. Lecture Notes in Computer Science, vol. 2260. Springer Berlin Heidelberg, 2001, pp. 360–363.

[42] A. O. Michael Backes, Christian Cachin, "Secure key-updating for lazy revocation," in *Computer Security ESORICS 2006*, ser. Lecture Notes in Computer Science, vol. 4189. Springer Berlin Heidelberg, 2006, pp. 41–50.

[43] E. Fujisaki and T. Okamoto, "Secure integration of asymmetric and symmetric encryption schemes," in *Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings*, ser. Lecture Notes in Computer Science, vol. 1666. Springer, 1999, pp. 537–554.

[44] T. O. Eiichiro Fujisaki, "Secure integration of asymmetric and symmetric encryption schemes," *Journal of Cryptology*, vol. 26, pp. 80–101, 2013.

[45] D. Boneh and B. Waters, "Conjunctive, subset, and range queries on encrypted data," in *Theory of Cryptography*, ser. Lecture Notes in Computer Science, vol. 4392. Springer Berlin Heidelberg, 2007, pp. 535–554.

[46] D. Boneh and X. Boyen, "Efficient selective-ID secure identity based encryption without random oracles," *Journal of Cryptology*, vol. 24, pp. 659–693, 2011.

[47] R. S. Kumar and A. Saxena, "Data integrity proofs in cloud storage." in *COMSNETS*, D. B. Johnson and A. Kumar, Eds. IEEE, 2011, pp. 1–4. [Online]. Available: http://dblp.uni-trier.de/db/conf/comsnets/comsnets2011.html#KumarS11

[48] S. Yu, C. Wang, K. Ren, and W. Lou, "Achieving secure, scalable, and fine-grained data sccess control in cloud computing," in *INFOCOM'10 Proceedings of the 29th conference on Information communications*. IEEE, 2010, pp. 534–542.

[49] D. Recordon and J. Hoyt, "Openid authentication 2.0 - final," *Network Working Group Internet-Draft*, 2007, available at http://openid.net/specs/openid-authentication-2_0.html.

[50] A. Shamir, "How to share a secret," *Communications of the ACM*, vol. 22, pp. 612–613, 1979.

[51] http://www.facebook.com/help/405977429438260/, 2013.

[52] A. Beimel, "Secure schemes for secret sharing and key distribution," Ph.D. dissertation, Israel Institute of Technology, Technion, Haifa, Israel, 1996.

[53] L. R. Welch, "The original view of reed-solomon codes," http://csi.usc.edu/PDF/RSoriginal.pdf, 1997.