博士論文

Achieving Expressive, Scalable, and Efficiently Revocable Collaborative Data Access Control in Multi-Authority Cloud

(マルチオーソリティクラウドにおけるデータアクセス制御の表現性、

スケーラビリティ、効率的な失効処理の実現)

Supervisor: Associate Professor Dr. Hiroyuki SATO

This dissertation is submitted in partial fulfillment of the requirements

for the degree of Doctor of Philosophy

Department of Electrical Engineering and Information Systems,

Graduate School of Engineering

The University of Tokyo

37-147256

ファクキアウ ソムチャート　Somchart Fugkeaw

1 June 2017

Abstract

Cryptographic-based access control is a palpable solution for supporting flexible and secure data access control in data outsourcing environment. However, dealing with both access control enforcement and complexity of cryptographic keys is non-trivial. Among the models employed, Ciphertext Policy-Attribute-based Encryption (CP-ABE) is regarded as a suitable solution and mostly adopted since it reduces key management overhead compared to symmetric and public key encryption. In addition, CP-ABE provides the owners with the full control over their own policies. Nevertheless, there are no works that apply CP-ABE and show the practicality in implementing it in the complex and collaborative data sharing scenario where multiple users in different domains can access the data shared by multiple owners. Furthermore, the major drawbacks of CP-ABE related to the inability to express and enforce write privilege, the high communication and computation costs of key distribution, revocation, and policy updates are still re-current problems that have not been solved in an integrated manner by any previous works.

In this thesis, we propose an expressive, scalable, and efficiently revocable access control solution supporting collaborative data sharing in multi-authority cloud storage systems. Furthermore, all drawbacks of CP-ABE mentioned above are resolved in this thesis based on the following four key technical contributions.

First, we propose the access control scheme called Collaborative Ciphertext-Policy Attribute Role Based Encryption (C-CP-ARBE) based on the combination of Role-based Access Control (RBAC) and a Ciphertext-Policy Attribute-based Encryption (CP-ABE). We adopt a RBAC model to enhance the expressiveness of CP-ABE policy. At a conceptual level, the access control policy is designed to accommodate the privilege information that comprehensively expresses read and write access of each user. In addition, we propose a write access enforcement mechanism to enable write-permitted users to update data and retrieve the policy to re-encrypt it securely and efficiently.

Second, we propose the zero key broadcast encryption technique based on our proposed user decryption key graph and public key encryption for supporting efficient key management and minimizing key distribution cost. The proposed technique enables all generated keys to be securely stored in a cloud server and dynamically invoked upon the user`s request. Compared to the CP-ABE, our scheme provides more scalable and practical in supporting a large number of users.

Third, within the cryptographic process of C-CP-ARBE, we propose a two-layer encryption (2LE) scheme to enable a more efficient user revocation with the computation cost reduction for user key re-generation and file re-encryption. With our 2LE method, a data is encrypted with the CP-ABE and then the ciphertext is encrypted with the AES symmetric encryption. Our proposed scheme minimizes the cost of user revocation in a manner where there is no file re-encryption and key re-generation if there is a user revocation. Our strategy is to update symmetric key based on the re-computation of the public role parameter. Hence, only the symmetric encryption layer is changed while the inner CP-ABE encryption layer is not affected. In addition to the user revocation level, we also investigate the problems of attribute revocation which is a finer level in CP-ABE. Revoking an attribute (s) introduces unavoidable overheads including expensive overheads for key re-generation, data re-encryption, and key re-distribution. To this end, we embed our newly proposed scheme called Very Lightweight Proxy Re-Encryption (VL-PRE) scheme into the C-CP-ARBE. In VL-PRE, size of the re-encryption key is small and key updates method is used instead of key generation. These properties of VL-PRE significantly reduce the computation cost for computing the re-encryption key in all revocation cases.

Another core focus in this thesis is the proposal of an optimized and scalable policy update management scheme. In CP-ABE, the data owner needs to re-encrypt files and send them back to the cloud when the policy is updated. This incurs overheads including computation, communication, and maintenance cost at data owner side. To this end, we develop policy updating algorithms to handle the policy change and employ VL-PRE to optimize the cost of file re-encryption when there is an update of policy. The security proof and policy update validation criteria are given to guarantee the security, correctness, and accountability of the proposed scheme.

To evaluate all the proposed features of C-CP-ARBE, we conducted the experiments by setting the simulations to evaluate the encryption, decryption, and revocation performance. Specifically, we did experiments to compare the performance of our C-CP-ARBE and related works regarding the user revocation, attribute revocation, and policy update performance. The experimental results confirm that our proposed C-CP-ARBE is practical and efficient in practice.

Acknowledgement

There are many great people who contribute to the success of this thesis. Without their support and help, this thesis would not be achieved.

First of all, I would like to express my utmost gratitude to my supervisor, Prof. Hiroyuki Sato, who guided me in both academic and life settings related to Japanese matters throughout my three year of PhD study. I remembered he provided a very warm welcoming and helped me regarding the settle-up in Tokyo since the first day I came to Japan for the study. It has been my long-lasting impression. He gave me the opportunity to choose the research topic regarding the cloud security I loved to do. I do appreciate insightful suggestions and technical guidance he always provided and put on my research production including this thesis and research papers throughout my PhD time. My deep appreciation also goes for his time spending in reviewing and providing constructive comments with his technical merit that greatly contribute to the completion of this thesis. I also thank for his tremendous support that gave me opportunities to travel to many countries for presenting my research papers. Thank you for time he was always available whenever I had problems and needed his helps. This thesis would not have been possible without his time and effort.

I want to thank Prof. Pierangela Samarati who initially guided me the topic about cloud security that is one of the promising areas in information security research. I also thank for her technical review in giving useful and practical comments on my initial research paper. That would be a fortunate starting point of this thesis.

I am grateful to Prof. Hitoshi Aida, Prof. Tomohiro Kudoh, Prof. Hiroyuki Morikawa, Prof. Masaya Nakayama, and Prof. Motonori Nakamura for serving as my dissertation examining committee and for their constructive comments on my research. Their suggestions and comments are greatly valuable to this thesis.

I would also like to thank Ajarn Koong, Dr. Jarernsri Mitrpanont, who was my former M.Sc. advisor at Mahidol University, for her constant support and encouragement for pursuing up my dream in obtaining a PhD. She was the first teacher who taught me how the scientific research is formally conducted and how good technical paper writing is. Her hard working and passion in doing research and teaching are impressive and would be a role model I would like to follow.

I want to thank all the staffs at Department of Electrical Engineering and Information Systems, all network lab members at ITC, UTokyo for providing the resources and good research environment. Also, I thank for their help about any Japanese procedures I had to deal with throughout my study time in Japan.

I thank my current employer, Thai Digital ID Co., Ltd. for giving me an excellent and challenging job for several years, and the time to pursue a PhD and finish this thesis.

My sincere thank further goes to the Otsuka Scholarship Foundation that provided me the scholarship during the AY 2014-2016. The scholarship was a great contribution to this achievement enabling me to live and study in Tokyo in a doable manner.

Finally, I would like to thank my family, my mother, my father, and my sister for their love, constant support, encouragement, and understanding. Their power of cherish makes my PhD path is warm and determined.

Contents

List of Figures

List of Tables

Chapter 1: Introduction

This chapter first provides the definition, background, and security challenges of cloud computing. Then, existing access control solutions and an example of a collaborative data sharing scenario are presented. Furthermore, the motivations, problem statements as well as the contributions of this thesis are depicted. The outline of the overall content of this thesis is also given in the final part of this chapter.

1.1 Background of Cloud Computing and Its Security Challenges

The US National Institute of Standards and Technology (NIST) gives a formal definition of the cloud computing in [50] as follows:

"Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction."

Cloud computing has emerged as a successful paradigm to deliver flexible and convenient data processing and sharing. Many businesses firms and several public enterprises such as hospitals, governmental units etc. have adopted cloud computing solution for their IT operations and project implementations. However, there are still much more enterprises that are considering to employ cloud solution for their strategic data storage and processing. Some of the major concerns for cloud adoption include security and privacy. Accordingly, the provision of security and privacy is a key requirement for the widespread use of the cloud.

According to [50], the following five major characteristics of cloud computing model are discussed with their security challenges:

- On-demand self-service. Cloud users are allowed to access and use the cloud resources as needed without any human intervention. Therefore, the availability of cloud service is normally guaranteed in the service level agreement (SLA). *However, high availability exposes the high chance of unauthorized accesses.*

- Broad network access. Cloud providers usually provide flexible accessibility for users to use the services over the network by accessing through various standard client platforms (e.g., mobile phones, tablets, laptops, and workstations). *Due to heterogeneous access capabilities over the public networks, service vulnerabilities and promising attacks such as man-in-the-middle attack can be taken place anytime.*
- Resource pooling. As a multi-tenant model, computing resources provided by a cloud provider are pooled and shared to the multiple users. The sharing of portion of available resources is dynamically assigned to users based on their requirements. In this sense, users are unable to control the physical location of the resources provided. *Lack of physical control and resources shared among multiple users increases the privacy and security risk of data outsourced in the cloud. This is a major factor for businesses having sensitive data to decide not to adopt the cloud technology.*
- Rapid elasticity. Resources management can be done flexibly and automatically by the provider. Users can also scale up the resources based on their computational demand in any quantity at any time. This characteristic highly benefits the users in terms of scalability and availability of resource provisioning. *Even though enlightening the elasticity and flexibility for scaling up the resources offers great potential for cloud consumer, several security challenges remain. Scaling up computational resources increases a higher chance of the vulnerability of resource control and points of security compromise. Consequently, additional access control and the applicable security policy should be properly adapted and defined to harness the use of data and/or applications run on the cloud. The cost of security management is therefore increased upon the elasticity feature.*
- Measured service. In a cloud operation, resource usage can be monitored, controlled, and reported. To provide transparency for both the provider and cloud users, trust and appropriate monitoring tools need to be available. Also, the applicable SLA is normally used to warrant the service quality compared to the quality agreed. *With this characteristic, the challenges on authentication and accountability will arise. To this end, secure and efficient access control to audit logs and reports as well as the integrity of these data are essential.*

Against the above security challenges, strong authentication technologies such as PKI authentication and access control are commonly used techniques to address the security problems. However, only authentication and general access control mechanism alone are not sufficient for effective access control for data outsourced in the cloud. This is because cloud storage server is considered as an "honest but curious" [1] or semi-trusted servers that the data owner cannot fully trust. In addition, the users and cloud service providers are not in the same security domain and sensitive data is in risk of being hacked, modified, and exposed. In many organizations, sensitive data to be outsourced may be very valuable and a compromise on the data will cause severe impact to the data owners or other relying parties.

Various security challenges have been addressed by research solutions in different focuses such as access control [e.g., 1,2,5-7,10, 13-15,22,49], data integrity in cloud [e.g., 51,52, 57], secure query and search over cloud [e.g., 55, 56], and cloud auditing [e.g., 53, 54, 58, 74]. Essentially, strong access control is typically considered as one of the top concerns of the cloud security.

In February 2016, Cloud Security Alliance (CSA) announced the Treacherous Twelve' Cloud Computing Top Threats in 2016 [65]. The CSA working group has identified and ranked the top 12 most salient cloud security concerns based on the survey result. The result of security concerns rated by approximately 270 respondents in the survey indicates the ranking of security issues in cloud computing as follows:

1. Data Breaches
2. Weak Identity, Credential and Access Management
3. Insecure APIs
4. System and Application Vulnerabilities
5. Account Hijacking
6. Malicious Insiders
7. Advanced Persistent Threats (APTs)
8. Data Loss
9. Insufficient Due Diligence
10. Abuse and Nefarious Use of Cloud Services
11. Denial of Service
12. Shared Technology Issues

Among the criteria in the List [65], the top two security concerns are on the data breaches and access management. Therefore, the provision of data confidentiality preservation as well as availability of strong access control mechanism is crucially required to alleviate the concern and to increase the confidence in cloud adoption.

By nature of access control requirement to be performed outside the security and resource ownership together with the requirement of encryption feature, a simple access control list (ACL), the traditional access control models such as discretionary access control (DAC), mandatory access control (MAC), and Role-based access control (RBAC) are not applicable for supporting access control in outsourcing environment such as cloud computing. In cloud computing environment, users, data owners, and service providers are generally not in the same security domain. Besides, users are normally identified by their attributes and not by predefined identities. Therefore, the identity-based access control models such as DAC or MAC cannot be applied in the open cloud environment. RBAC is more suitable to be used in cloud system as its scalability and the access is subject to the roles of users rather than the fixed identity. RBAC can be also employed to implement several important security principles such as least privilege, separation of duties, and data abstraction [75]. RBAC can be effectively used when users can be clearly separated according to their job functions. However, in data outsourcing environment that supports the collaborative use of data, the enforcement of the organization's access policy is not restricted to be enforced to its organization's user only. The RBAC is not practical for such environment since it is possible that there are unknown users from different organizations allowed to access the shared data. For example, users from different organizations may have the same role but they may be granted different permissions for data access. Furthermore, implementing RBAC in a cloud needs additional cryptography mechanism to deliver the confidentiality and privacy of the outsourced data. Such a requirement introduces expensive costs for managing both access control and cryptographic keys.

1.2 Current Data Access Control Solutions

Ensuring data protection in the cloud needs the guarantee of their confidentiality, integrity, and availability [37,38,39]. Cryptography or encryption is one of principal solutions that can be used to address confidentiality, integrity, and availability problems. Several previous research works have either applied traditional cryptographic scheme such as symmetric encryption and public key encryption [1,2,5,6,22,49] or another public key encryption primitive called attribute-based encryption (ABE) [7, 9,10,13-15, 19, 21, 22] to be incorporated into their access control solution for the cloud. However, the traditional cryptography methods based on symmetric encryption or public key encryption introduce high key management cost and high maintenance cost in handling multiple copies of ciphertext when they are used to support secure data sharing in cloud environment. In the symmetric key encryption, cost of key change-over and key distribution is very expensive when there are frequent revocations of users and are a high number of users in the system. Also, it is highly vulnerable that the same key is shared among multiple users. For the public key encryption, data is typically encrypted with the public key of the user who is authorized to gain access to the data. Thus, if there are multiple users authorized to access the same data, a number of cihphertexts will be produced with respect to the number of those authorized users. The maintenance of multiple copies of ciphertext and data update makes this approach impractical.

Regarding the ABE solution, even though it offers the lightweight encryption and supports fine-grained access control for data outsourcing scenario, the lack of capability for write access enforcement and expensive overheads for key distribution and revocation limit its scalability and efficiency in large-scale environment. In addition, the access control solution dedicated to collaborative and cross-domain data sharing in multi-authority cloud has not been addressed by any work. In the collaborative data sharing scenario supporting multiple users, each user must have a key(s) to decrypt the encrypted files. In this scenario, users from different organizations may be allowed to collaboratively access the data shared by another organization. To this end, the access control mechanism should manage cross authentication and the access control policy needs to accommodate the user's attributes issued by multi-attribute authority and can enforce the access control dynamically. A concrete scenario of the collaborative data sharing is illustrated in the next section.

In this dissertation, we emphasize on a collaborative access control solution based on the attribute-based encryption scheme in which the scalable, fine-grained and expressive access control are the core focus.

1.3 Access Control for Collaborative Data Sharing in the Cloud

In a dynamic and large scale data sharing, an efficient support of collaborative data usage among multiple relying parties is crucially required. Multiple data owners can share their data deliberately to any business users from different organizations as long as the shared data is used by the legitimate users. Therefore, providing security and access control that supports collaborative use of outsourced data is a real challenge that will be a primary focus of this research.

Several web-based groupwares are available to support collaboration among users and even to allow users to share data connected to their own database based on the available features of the applications. However, groupware applications usually do not support encryption feature. In practice, outsourced data is generally encrypted and the collaborative users must have a key(s) to decrypt with respect to a given access control policy. This is to guarantee the confidentiality and privacy of outsourced data shared among multiple groups of users.

To illustrate the collaborative access control for data sharing in multi-authority cloud scenario that this thesis will entail, we use healthcare data sharing as a running example throughout this thesis. Figure 1.1 presents a sample scenario of collaborative healthcare data sharing.

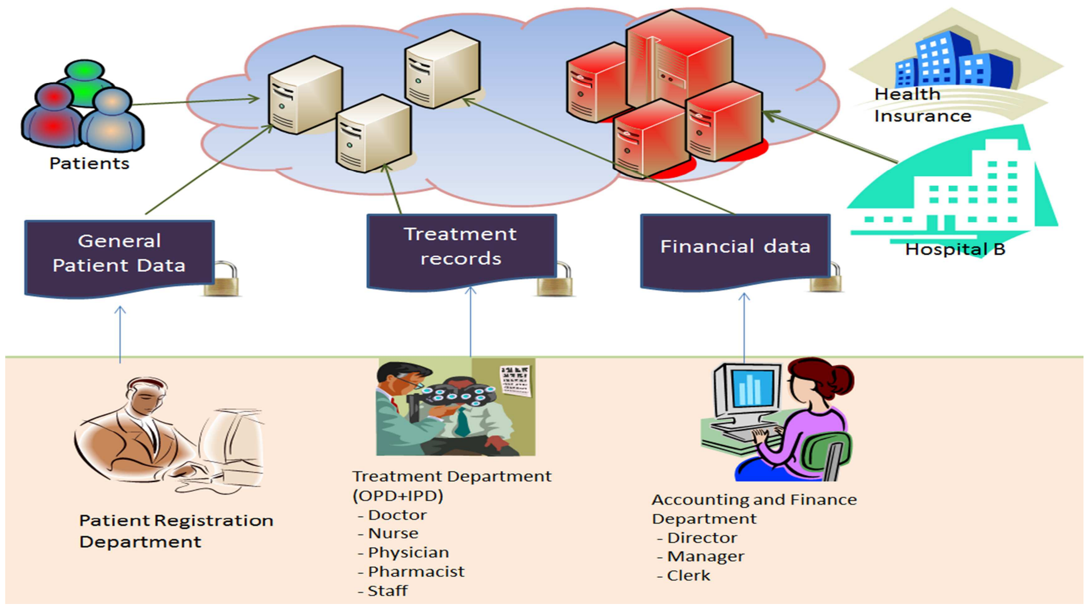


Figure 1.1: A Healthcare Data Sharing Scenario in Cloud Computing

To outsource medical or healthcare data to the cloud, departments (data owners) in the hospital encrypt data and uploaded them to the cloud. In the collaborative use of data shared in the cloud, the data are not only shared among the users in their departments, but they can be also shared to other users who work in different departments of the hospital or those who are from the relying parties such as partner hospital B, and health insurance company. The enforcement of appropriate access permissions (read or write) is also essential for the real-world data sharing. For example, the patient registration records generated by the patient registration department can be shared with write access to the nurse for recording the appointment in the registration file. Also, the record of patient registration may be shared with read only to the insurance company for verifying the visit histories of patients who claim for the payment from the insurance.

Accordingly, access control mechanism used in such scenario must be fine-grained in enforcing least privileges to individual users, scalable in supporting a large number of users, and flexible in managing the policies to any changes of access control elements. Furthermore, a cryptography feature is required for the access control mechanism for data outsourcing environment. However, implementing key management in cloud is not simple since a large number of keys may need to be generated and distributed to users for decrypting the data residing in the cloud. The computation and communication overheads in managing keys in the cloud environment are usually expensive. Therefore, practical and cost-optimized approach for managing cryptography operations such as key management (key issuance, key distribution, key revocation) is a crucial issue that naturally extends the challenge of access control in the collaborative use of data outsourced in cloud computing.

Among the cryptography-based approaches, attribute-based encryption (ABE) is considered as a suitable solution to be used as the access control model for outsourced data. In [18], Sahai and Waters introduced the original ABE scheme called a Threshold ABE system. In this scheme, ciphertexts are labeled with a set of attributes S and a user's private key is associated with threshold parameter k and another set of attributes S'. The user is able to decrypt the ciphertext if at least k attributes must overlap between the ciphertext and his private key. However, this technique has the limitation for designing more general system because of inexpensive threshold semantics. Hereafter, Goyal et al. proposed a more general ABE model called Key Policy –ABE (KP-ABE). In KP-ABE, a ciphertext is associated with a set of attributes and user key is associated with the access structure. User can decrypt the ciphertext if the attributes associated in the ciphertext match the access structure of a user’s key. Hence, KP-ABE encryption is subject to the key generator. In 2007, Bethencourt et al. [16] introduced Cipheretext Policy-ABE (CP-ABE to allow the data owner to define the access structure and use it to encrypt the data. A user’s key is constructed by a set of attributes. The user can decrypt the ciphertext if their key satisfies the access structure associated to the ciphertext. To this end, CP-ABE has been more adopted by several works for supporting secure access control in data outsourcing environment as it possess the access control with encryption feature and delivers the encryption based on the discretion of data owner.

1.4 Motivation

As a strong requirement for access control solution for the collaborative use of outsourced data, we come up with the following motivating questions to highlight the requirements for enhancing ABE-based access control to support secure data sharing in multi-authority cloud scenario.

1) How can the shared data (e.g OPD file, financial record of treatment) be properly accessed by authorized groups of users (even the different roles, department, organization) with least privilege (read or write) of each individual or groups of user?
2) How can keys be efficiently managed and dynamically assigned to users whom may belong to multiple organizations with different roles (e.g., a doctor James may either belong to Hospital A as a medical doctor or belong to a university as a professor)?
3) How does the attribute or user revocation give minimal impact to the remaining users in the hospital and how the forward security (a newly joined user can use her key to decrypt the previously available ciphertext if she has sufficient attribute satisfying the access policy) and backward security (a revoked user cannot decrypt any new ciphertext requiring the revoked attribute to decrypt) are supported?
4) How can the data owners manage (add, update, delete) the access control policies used to enforce the authorization over the outsourced data with the minimal impact to the users while maintaining the confidentiality of the policies.

To the best of our knowledge, our research is the first attempt in examining the access control solution for collaborative and cross-domain data sharing in the multi-owner and multi-authority cloud system. We propose an integrated access control solution that accommodates the collaboration feature of groupware application with a fine-grained ABE access control management in multi-owner and multi-authority cloud systems.

1.5 Problem Statements

Existing access control solutions in cloud storage systems generally concentrate on minimizing key management cost, reducing computing cost of interaction between data owner and outsourced data storage, and improving the scalability. Ciphtertext-Policy Attribute-based Encryption (CP-ABE) [16] is recognized as one of the most applicable techniques for cryptography-based access control for cloud computing since it reduces key management overhead compared to symmetric and public key encryption. In addition, CP-ABE provides the owners with the full control over their own policies. However, in the data sharing scenario, users may obtain their attributes from multiple authorities to gain access to resources shared by multiple data owners. To solve this kind of problems in this scenario, multi-authority attribute-based encryption (MA-ABE) schemes [7, 9,10,13-15, 21] have been proposed.

Even though CP-ABE provides flexible and secure access control for data outsourcing scenario, there are drawbacks of CP-ABE related to the inability to express and enforce write privilege, expensive overheads for key distribution, and high cost for user and attribute revocation, and policy update. Actually, these problems impede the efficiency of access control and they have not yet been solved in the comprehensive way.

First, CP-ABE is generally designed to support secure data sharing with read access only. Actually, this scheme has no mechanism for write access enforcement. We discuss this problem into two sub-problems.

(1) Policy expressiveness. The CP-ABE policies do not contain the permission of users; only a set of attributes and logical operators is specified. In CP-ABE, data is encrypted with the policy specified by the data owner and only the owner can modify data while all other users can only read them. With the read-only assumption, it is not practical in several scenarios where a data owner may also want to authorize some users to write and update the outsourced data. This limits the expressiveness and manageability of access policies. Hence, *Policy accommodating a comprehensive privilege (read and write) specification with efficient enforcement is truly desirable.*

(2) Data re-encryption cost. If the users are allowed to write or update the data, they need to be given the policies to re-encrypt the data and upload them back to the cloud. *Securely sharing policies among users with write permission in parallel to optimizing data re-encryption cost is the real challenges for implementing write access control in CP-ABE setting.*

Second, all user keys generated by the attribute authority (AA) will be delivered to all users in the way that all of them are encrypted before they are distributed to an individual user. This renders the administrative and communication overheads to the AAs or data owners. The cost is proportional to the number of users in the system. The key generation and key distribution cost are incurred whenever user or attribute is revoked. *Managing multiple keys and minimizing key distribution cost in CP-ABE setting are also paramount importance for the practical deployment of CP-ABE based access control.*

Third, In CP-ABE, the attributes issued by AA can be shared by multiple users. If there is any case of user or attribute revocation, non-revoked users whose keys contain the revoked attribute need to update their keys. Also, if the revoked attribute appears in the access policy, the files need to be re-encrypted under the new policy. Re-encryption is also required to prevent the unauthorized decryption of the keys containing the revoked attributes. Hence, the revocation cost of CP-ABE includes the costs for key re-generation, key re-distribution, and file re-encryption. Also, the communication for retrieving file for re-encryption and loading file back to the cloud is another burden for data owners. Nevertheless, existing works have not provided the solution for solving the revocation in both user and attribute level. *Therefore, designing and developing a revocation scheme delivering the optimized computation cost and communication cost for both user and attribute revocation is truly required and it is the first attempt in resolving the revocable CP-ABE in an effective and comprehensive manner.*

Finally, policy updating in the CP-ABE introduces the cost of file re-encryption similar to a revocation case. If the policy is updated, a data owner needs to download all files encrypted with the before-updated policy from the cloud and then re-encrypts them with the updated policy, and loads new cihertexts back to the cloud. This introduces both computation and communication overhead to data owners. Also, access policies are locally maintained at data owners side. Therefore, data owner must accommodate the request for policy retrieval for data encryption by write-permitted users. This limits the flexibility and availability of data encryption service. *Devising the protocols that optimize policy update cost and guarantee the correct update of the policy content is strongly required for the practical access control function.*

In this thesis, we aim to compliment the above shortfalls of CP-ABE and extend our research vision to deliver the expressive, flexible, scalable, and fine-grained access control for supporting collaborative use of outsourced data in multi-authority cloud environment.

1.6 Contributions of This Dissertation

Our proposed scheme *is* ***the first attempt*** **in solving the major drawbacks of CP-ABE as integrated solution and dealing with collaborative access control** for cross-domain data sharing in cloud scenario where there is multi-owner and multi-authority. Therefore, the technical innovation is reflected in the following contributions.

1. **Expressive, flexible, scalable, and fine-grained access control**

Our proposed access control model is based on the integration of role-based access control (RBAC) into the CP-ABE scheme in order to gain the benefits of these two schemes. Here, the CP-ABE based access policy is empowered with the RBAC policy expression where a set of attributes is grouped and assigned under the specific roles and the privilege information is also included. The access policy is modeled in a tree-based structure and it allows attributes issued by different authorities to be modeled in the same access policy. Furthermore, we accommodate RBAC constraints as the additional access control features supporting a more expressiveness of CP-ABE. Therefore, our tree-based access structure is highly expressive, flexible, and scalable for fine-grained access control enforcement over multi-authority and multi-users federations. Our scheme also supports users having write privilege can update the data and can request for file re-encryption to the proxy without the intervention of data owners.

2. **Key Complexity Reduction**

In the collaborative access control settings where multiple (authorized) users deliberately access the data shared by multiple data owners, key management operations including key generation, key distribution, and key usage (encryption and decryption) are so heavy tasks that needs to be well designed. Less complicated and highly secure key management is desired. The key management must be also scalable in the number of users. In CP-ABE scheme, when the attribute authority generates the keys for users, the generated keys will be encrypted and distributed to users. Hence, key broadcast or key distribution is another major overhead that the authority has to be responsible for key management. This overhead becomes even more tiresome if there are a huge number of users in the system. To this end, we propose the zero key broadcast encryption to fill up this gap of the CP-ABE. Our key management strategy is based on user decryption key graph (UDKG) and public key encryption. Basically, after user decryption keys are generated, they are all encrypted with the individual user's public key and they are then structurally stored in the UDKG in the cloud. These keys will be dynamically invoked upon the user`s request.

3. **Efficient user and attribute revocation management**

Our proposed access control model provides an efficient user and attribute revocation mechanism that is capable to avoid successive revocation impacts to remaining users. The impacts usually are caused by key re-generation and file re-encryption. These subsequent costs of revocation are the common problem of the original CP-ABE scheme. The revocation model should also support backward security that the revoked users should no longer use their keys or attributes to collude with other active users to access the file in the cloud server. We propose a two-layer encryption model and very lightweight proxy re-encryption (VL-PRE) to deliver the optimal cost of user and attribute revocation respectively.

4. **Dynamic, secure, and optimized policy update management**

In the access control, access policy is considered as a critical part of authorization system. During the deployment of access control policies, the changes of policy content or rules may frequent occur. If the policy is updated, data owner is required to re-encrypt the data to reflect the new policy. To reduce the burden regarding the computation and communication cost at data owner side, we introduce a policy updating mechanism based on our proposed VL-PRE to support file re-encryption. The criteria of correctness, completeness, and security are used to control the efficiency and integrity of our algorithms.

1.7 Organization of This Dissertation

The rest of this dissertation is organized as follows.

Chapter 2 provides the literature review of major three research issues: (1) access control solutions for outsourced data, (2) attribute revocation management, (3) access policy update management.

Chapter 3 presents our proposed access control scheme called Collaborative - Ciphertext Policy- Attribute Role Based Encryption (C-CP-ARBE). Theoretical backgrounds of CP-ABE and RBAC are reviewed in details. Then, we give the definition of our proposed access control model based on the integration of CP-ABE and RBAC, present write access enforcement method, and explain the system model. At a core of the chapter, the formal construction of our C-CP-ARBE algorithms, its security proof are then introduced. Finally, the analysis of C-CP-ARBE and the evaluation and experiment part are given.

Chapter 4 presents the attribute revocation method based on our newly proposed Very Lightweight – Proxy Re-Encryption (VL-PRE). We describe VL-PRE protocol and its security proof. Finally, the comparative performance evaluation is conducted to show the substantial improvement of VL-PRE against the previous works.

Chapter 5 proposes the comprehensive policy updating algorithm and how the VL-PRE is used to support secure and efficient policy update of our C-CP-ARBE. Finally, the performance evaluation of policy update is performed to substantiate the efficiency of our proposed solution.

Chapter 2: Literature Review

This chapter discusses the related works in the area of access control of outsourced data. The chapter begins with a review of the concept of cryptographic approaches used to support a secure access control for outsourced data. Non-attribute-based encryption and attribute-based encryption (ABE) solution are first discussed. Then, approaches related to user and attribute revocation, and policy updates in cloud systems are presented.

2.1 Access Control Enforcement of Data Outsourced in Cloud

In this research, we classify the cryptographic-based access control solutions into two categories: a non-attribute based access control and an attribute-based access control.

2.1.1 Solutions Based on Non-Attribute Based Access Control

Traditional access control models can be generally divided into three types: discretionary, mandatory, and role-based [17]. For the discretionary access control (DAC) model, the access control policy is specified based on the data owner's decision. In DAC model, users usually are grouped and member in the same group may have the common permission with others have. Due to the limitation of scalability, DAC is not practical for the environment where there are multi-user and multi-application. The mandatory access control (MAC) model is more suitable for the distributed systems as the model defines the mapping between user and resource, MAC model usually specifies the multi-level security policy for a subject and object e.g., highest security, medium security, low security, and unclassified. For the role-based access control (RBAC) model, there is a clear relationship specified between user, role, permission, and resource. It is widely regarded it is an effective model for general organizational access control since it can significantly reduce the overhead maintenance of the policy and it is flexible in assinging a number of users into specific roles and mapping roles to resources. Thus, RBAC is more scalable than DAC and MAC model.

In cloud computing environment, resources are resorted outside the security domain of the data owner and the access control elements such as users and their job functions can not be fixed in the policy in the open cloud environment. Therefore, the traditional access control like DAC and

MAC are truly not applicable. Even though RBAC seems more suitable to be used as an access control model in the cloud, it lacks the embedded encryption capability which is required for regulating the encryption enforcement over the outsourced data.

For years, there are research works [1,2,4-6,8, 49] that applied encryption and key management techniques to construct an access control model for cloud storage systems.

For example, S. De Capitani di Vimercati et al. [1] proposed an access control in cloud scenario by allowing multiple users to selectively access to the outsourced data. The approach integrates encryption and authorization to support the dynamic changes of access control policies based on their proposed selective encryption. The goal is to translate an authorization policy to be enforced in an equivalent encryption policy based on the hierarchical organization of keys. Here, two-layer encryption is imposed on data. Technically, the inner layer is done by the owner as of the initial encryption while the outer layer is used to enforce the changes of access control policy. Any update to the control rules entails a change in the tree. Access revocation requires re-encryption of the affected files. The authors extended their work to support write access proposed in [6] by designing a key derivation techniques for read and write authorization and introduced together with the HMAC and signature-based approach applied for integrity confirmation.

This paper is the first attempt focusing on addressing the problem of evolution of authorization policy by translating the authorization policy to be enforced in an equivalent encryption policy. The authors introduce the encryption policy for resource encryption and key distribution. Here, a key derivation method adopted from [12] was exploited based on the definition and computation of public tokens. Initially, users are given a single key according their access policy and if there are any updates on the resources or permissions, the resources and policy will be re-encrypted. Users will be assigned a new key with the key derivation method.

The authors defined key and token graph to represent the relationship between users, keys and resources. However, the proposed technique introduces a complexity when there is a creation of new files and user grant/revocation. In addition, users need to hold his individual key and a shared group key associated to the access control lists that they have a privilege to access. In addition, the computation cost for translating the changed policy is high when the environment composes of several users and frequent update on the policy or privilege of users regularly takes place. This makes the solution unscalable in practice. Lastly, the approach does not support the control of multiple accesses in multi-owner data outsourcing scenario.

Raykova et al. [2] employed selective encryption to enforce read and write privileges on outsourced data. The authors introduce a two-level access control model that combines fine-grained access control, which supports the precise granularity for access rules, and coarse-grained access control, which allows the storage provider to manage access requests with only limited information from its inputs. Outsourced resources are arranged into units called access blocks and the access control is enforced in the cloud only at the granularity of blocks. Here, the read and write access are specified in the data block level. The asymmetric encryption is used to specify the enforcement in the way that the private key is used to enforce the read privilege and the public key is used to enforce the write privilege. However, the paper does not demonstrate the performance of a high volume transaction of the read and write access control in their model. In addition, the integrity of the access blocks and ability to update a policy are not addressed. The solution also provides inflexible enforcement of policy where resources with the same read access control policy can be modified by two different sets of users.

In [5], CloudSeal system was proposed and implemented to demonstrate the privacy of content delivery in public cloud. The authors integrate symmetric encryption, proxy-based re-encryption, k-out-of-n secret sharing, and broadcast revocation mechanisms. CloudSeal maintains cache of the major part of a stored cipher content object in the delivery network for content distribution. This optimizes the cost of content delivery and key management. For forward security, a user cannot access content if he/she is not a group member. This is achieved by the encryption key that is only distributed to the authorized users. For backward security, a user cannot access content after he/she leaves or is revoked from the group. This is achieved by the re-encryption process. If user is revoked from the system, a new re-encryption key will be issued to the affected group of user and the content accessed by the group is re-encrypted. Even though this papers fully supports a flexible access control and confidentiality of the content published in the cloud storage, the key generation cost and key update performance expose the efficiency problem. The proxy re-encryption can alleviate the re-encryption cost but the content provider always needs to monitor and stays online to support re-key generation if there are any new users joining the group or leaving the group. In addition, the solution does not support multi-privilege access control. Users in the same group may have different privileges in accessing the content, thus the assignment of the same key to all users in the group cannot be done in this scenario.

In [49], Kaaniche et al. proposed CloudaSec as an access control framework for securely sharing outsourced data in the public cloud. The proposed approach is based on the content hash keying system using the Elliptic Curve Cryptography (ECC). This approach exploits dual encryption. The authors apply symmetric cryptography for data encryption and employ asymmetric encryption for key encryption. CloudaSec delivers flexible access to encrypted contents, by dynamically sharing a group secret key within the group. If any member of the group is revoked, the group secret key is re-generated. However, the approach does not support write access enforcement. Furthermore, the examination of re-computing cost and key re-distribution cost of a new group key is not provided especially when there are frequent user revocation cases. With this scheme, only user revocation level is supported.

In summary, the proposed cryptographic based solutions where users are given different keys encounter a scalability and key management problem. This is because the number of keys grows linearly to the number of users and the overhead in managing keys significantly high especially when there are a large number of users. Even though key distribution and key updates solution are addressed, the complexity of cryptographic operations still exists.

2.1.2 Solutions Based on Attribute-based Encryption

Cryptographic-based access control is a common approach for supporting security and privacy of the data outsourced in third party environment such as cloud storage systems. Existing approaches [1,2,5,6,22,49] based on symmetric key and public key encryption either introduce key management problem or key derivation complexity, or produces multiple copies of ciphertext. This becomes even more impractical for the cloud environment consisting of a large number of users since the overhead for both communication and computation will be tremendously high.

Attribute-based encryption (ABE) turns out to be a suitable and preferred solution [76] for satisfying scalable, flexible, and fine-grained access control solution. The key construction of

ABE is based on bilinear maps or bilinear pairings. In 2006, Goyal et al. proposed key-policy attribute-based encryption (KP-ABE) [11] to serve a more general and richer encrypted access control. In this scheme, the ciphertext is associated with a set of attributes for each of which a public key component is defined. User secret key is constructed to associate with the access structure. However, the KP-ABE scheme does not give the data owner a full control over the access policy. Thus, works based on KP-ABE scheme [3,9] suffer from this limitation.

To address this drawback, the Ciphertext Policy Attribute Based Encryption (CP-ABE) was proposed in [16]. In CP-ABE, each user is assigned a set of attributes which are embedded into the user's secret key and a public key is defined for each user attribute. The ciphertext is associated with the access policy structure in which the encryptor can define the access policy by her own control. Users are able to decrypt a ciphertext if their attributes satisfy the ciphertext access structure. The encryption performance of CP-ABE is generally subject to the size of policy and file size while the decryption performance is dependent on the no. of attribute containing in the user decryption key [69].

Existing works employing CP-ABE model usually concentrate on minimizing key distribution, reducing computing cost of interaction between data owner and outsourced data storage, improving scalability, efficient enforcement of a policy, and addressing the revocation problem.

In [19], Zhou et al. proposed a role-based encryption (RBE) scheme for cloud storage systems. The proposed RBE uses ABE for cryptographic access control and use identity broadcast encryption for key distribution. For the access control, role-based access control (RBAC) policy is enforced through a public parameter of role and a group public to encrypt the data. To decrypt the data, user decryption key containing the public attributes set and public role parameters are used. However, in this system data is encrypted by the data owner to the specific role, several copies of the encrypted data are required for authorized users who belong to different roles.

In 2013, the authors proposed a new RBE scheme [21] to enhance their previous work by enabling the encryption of a particular file can be done by a group of roles. Also, their new scheme eliminates the cost of file re-encryption if there is any user revocation operation. However, the computation cost for storing tuples of ciphertext and cost for computing symmetric key K for every request is impractical for a large number of users having frequent data access over the remote cloud.

In [8], Nabeel and Bertino proposed two-layer of encryption for protecting confidentiality of data in the cloud. They proposed a policy decomposition method for encrypting the data. With this approach, an access control policy (ACP) is decomposed into two pieces: one for course grained access control performed by data owner, and the other piece for fine-grained access control enforced by the cloud. With this scheme, the cloud server supports two services: the storage service, which stores encrypted data, and the access control service, which performs the fine grained encryption. This reduces the computation and communication overhead at data owner in dealing with the re-encrypting file if there is any change of user or policy. However, this scheme reveals a policy content to the cloud. Furthermore, this approach does not support write access enforcement.

To support a more complex environment of cloud computing where there is multi-owner and multi-authority, a multi-authority attribute based encryption (MA-ABE) was recently proposed by several works [7, 9,10,13-15, 21, 68, 72]. In fact, the MA-ABE scheme highlights the practical solution for solving the management of attributes issued by multiple authorities and the scheme is truly essential for today advanced data sharing scenarios where there are multiple owners sharing their resources to multiple users coming from both inside and outside the organization.

In [13,14],Chase et al. proposed multi-authority scheme where each user is given an ID and the attributes can be obtained from different authorities using different pseudonyms. The proposed MA ABE scheme requires a central authority to generate an aggregated secret key by integrating the secret keys from different attribute authorities. This enables the central authority to be able to decrypt all the ciphertexts. Therefore the central authority becomes the bottleneck and causes of performance problem. In [14] Chase et al. proposed an improvement to their previous work by removing the central authority but their scheme still limit the expressiveness of access control policy by supporting only AND formulae.

In [15], Lewko et al. introduced the decentralized multi-authority in the CP-ABE. Instead of the monotone access tree structure, they use a linear secret sharing scheme (LSSS) matrix as an access structure where the AND, OR gates are converted to the LSSS matrix to represent the policy conditions. However, their method is based on the composite order bilinear groups and does not support attribute revocation.

More recent MA-ABE approaches have paid attention on a more practical problem which is the solution for improving the key management and revocation.

In [9] the authors proposed a fine-grained and scalable data access control for personal health records (PHRs) by using attribute-based encryption. The paper addresses the access control in the multi-owners and multi-authority setting. To do so, the users in the PHR system are divided into multiple security domains namely public domains (PUDs) and personal domains (PSDs) to reduce the key management complexity and for a better user management. The PUDs consist of users who have the professional roles such as doctors, nurses, and medical researchers, while the PSDs personally relate to the data owner such as family members, friends who are granted the access by the owner. Two ABE systems are applied: for each PSD, the YWRL's revocable KP-ABE scheme [3] is used; for each PUD, the revocable MA-ABE scheme is used. In this scheme, the immediate revocation is supported since if any attribute is revoked, the affected attribute public keys are recomputed. For the MA-ABE applied in public domain, the PHRs are encrypted with the conjunctive normal form (CNF) among different AAs, i.e., $P:=P_1 \wedge \ldots \wedge P_N$, where P_k corresponds to arbitrary monotonic access structure. The secret key of user u, A_k^u should contain at least one attribute obtained from AA_k. The access control policy is enforced in user's secret key.

Even though the proposed KP-ABE scheme is designed to allow the data owner to specify the access policy but it needs to be predefined and agreed with the users. This is not a direct control over an access policy and it is very inflexible if there is any update to the policy. This scheme supports the write access in the way that the requestors who have an authorized to write the data are granted a time-related signature. The signature is coupled with the time period to limit the write privilege for that time window. However, the data owner has to be available to deliver the signatures to the requestors during the possible time period known by the users.

In [7], the authors proposed hierarchical attribute-set-based encryption (HASBE) by extending ciphertext-policy attribute-set-based encryption (ASBE) with a hierarchical structure of users. In this scheme, a trusted authority is responsible for generating and distributing system parameters and root master keys as well as authorizing the top-level domain authorities. However, the vulnerability of trusted authority of the hierarchical domains would be at risk to all users. Besides, to serve the revocation and re-encryption process, the data owner needs to be online during the period agreed with users.

Kan Yang et al. [10, 22] proposed DAC-MACS (Data Access Control for Multi-Authority Cloud Storage model. The authors apply CP-ABE technique to construct an access control model where there are several multi-authority issuing the attributes. The proposed scheme improves the decryption process and solves revocation problem in ABE by designing the decryption token and key update and ciphertext update algorithms. For the immediate revocation, their scheme reduces cost of data re-encryption since only the ciphertext getting an effect is updated. With the ciphertext update technique, the data owner needs to compute ciphertext components for new attributes. The entire computation cost is subject to the number of attributes appearing in the ciphertext and the type of update operations (i.e. OR, AND) over the access structure. Furthoermoe, this approach does not support write access and its policy is not applicable for collaborative data sharing.

In [40], Fangming et al. combined Attribute-Based Signature (ABS) and CP-ABE to enforce read and write access privileges, respectively. This approach is effective but it requires the presence of a trusted party for managing policy enforcement. Besides, there is no scenario or the evaluation to demonstrate how their approach is deployed in the distributed and large-scale data sharing environment.

Even though, CP-ABE scheme provides a fine-grained access control with cryptographic-based feature, the major drawbacks of CP-ABE related to the inability to express and enforce write privilege, high cost for key distribution and revocation, and lack of flexible and optimized policy update management are still re-current problems that have been not resolved as an integrated solution by any works.

Regarding to the write access enforcement, CP-ABE specification cannot be used to express the write privilege in the policy and this limits the expressiveness of the policy specification. Access control over data encrypted with CP-ABE technique is typically enforced by read access.

Most existing CP-ABE based approaches have not taken this issue into their solutions. However, in a collaborative data sharing, in addition to the data owner, some groups of users may have a privilege to update the data and load the updated files back to the cloud. Therefore, write privilege enforcement is crucially required for supporting the practical data sharing.

Key distribution based on key broadcast is a common operation for delivering user decryption keys to legitimate users in CP-ABE. However, the key distribution cost is proportional to the number of users. If there is any revocation occurred, the non-revoked users who share the attribute revoked needs to get the updated keys redistributed by the AA or owners. This overhead limits the scalability and efficiency of the system.

User and attribute revocation in CP-ABE scheme imposes the cost of file re-encryption and key re-generation. When there is a revocation in CP-ABE setting where the attributes are shared among the users and appeared in the access policy used to encrypt the data files, a file must be re-encrypted, and all non-revoked user keys requires either updated or re-distributed. The computation and communication costs resulting from the revocation is thus non-trivial and it will limit the efficiency and scalability in high user volume environment, such as data sharing in cloud system.

For the policy updating issue, there are very few works that take this issue into their access control solution. In fact, access policy management is one of the crucial features of any typical access control system. In CP-ABE, a number of policies are coupled with the ciphertexts stored in the cloud server. If there any changes on policies, the corresponding ciphertexts needs to be re-encrypted. Hence, the local policy management and cost of policy updates are also the problem that can introduce both communication and computation overheads.

Table 2.1 compares supporting access control features of four related approaches. As of Table 2.1, no works can satisfy all requirements of the desired features. Especially, the capability to support collaborative data sharing and flexibility in managing the policy in cloud storage have not been supported by any MA-ABE works. These requirements become necessary in real-world collaborative data sharing settings.

Table 2.1: Comparison of Access Control Schemes Related to Our Work

	Zhiguo Wan et al.[7]	**Ming Li et al. [9]**	**Kan Yang et al.[10]**	**L. Zhou et al. [19]**
Cryptographic Access Control Model	CP-ABE	KP-ABE	CP-ABE	CP-ABE and Role-based Encryption
Support Multi-Authority	Yes	Yes	Yes	No
Support Multi-Owner	Yes	Yes	Yes	No
Multiple decryption key management on cloud	No	No	Yes	No
Collaborative /cross domain data sharing	No	No	No	No
Support Write Access	No	Yes	No	No
Revocation Support	User Level	Attribute and User Level	Attribute Level	Attribute Level
Support secure policy update in cloud	No	No	No	No

In our research, we share the common research problems of existing works by enhancing the efficiency of secure data access control, key and user management, revocation management, and policy updates in cloud data sharing scenario. Therefore the gaps of research issues that have not yet addressed by the above approaches will be fulfilled by this thesis. In addition, we also extend the scope of our research to entail a more advanced collaborative access control in multi-owner and multi-authority cloud storage systems. To the best of our knowledge, our research is the first attempt that proposes the collaborative access control solution for supporting data sharing in multi-authority cloud environment where data owners share their data to multiple users that may come from different organizations.

2.2 User and Attribute Revocation Management

For the revocation problem in CP-ABE, there are several research works proposing solutions to handle and minimize cost of user revocation [7,10,11,23]. For instance, the works proposed in [7,11] introduce an “expiry time” attribute to a decryption key. Hence the key is effective only for a given period of time. The limitation of this method is that it does not support immediate or real-time user revocation. In [62], Ostrovsky et al. use negative constrains in an access policy to enable the revocation of certain attributes relates to negating the attributes. The system proposed encounters the scalability problem in handling the revocation of individual users, because the revocation requires the encryption of information of all revoked users and each of which is treated as a distinctive attribute. In [64], the authors propose the attribute-based revocation scheme based on the broadcast encryption revocation mechanism [63]. The scheme requires the generation of the public system key of which the size is proportional to the total number of users.

Nevertheless, such approaches focus only the user revocation level while the attribute revocation has not been examined. In [22], Hur and Nor proposed an access control mechanism based on CP-ABE with efficient attribute and user revocation capability. They exploited dual encryption method combining the attribute-based encryption and selective group key distribution in managing attribute group. Nevertheless, this scheme requires a trusted authority to manage all the attributes, issue secret keys to users, and revoke users’ attributes. Thus, it has a vulnerability in security point of view and performance problem [73]. In addition, the revocation of user both user and attribute introduces the costs as CP-ABE does.

Recently, Kan Yang et al. [10] proposed DAC-MACS (Data Access Control for Multi-Authority Cloud Storage model. The authors apply CP-ABE technique to construct an access control model where there are several multi-authority issuing the attributes. For the immediate attribute revocation, their scheme reduces cost of data re-encryption since only the ciphertext getting an effect is updated. However, the revocation cost for ciphertext and key update could be problematic if there are a large number of users as well as frequent and several attributes revoked.

Another solution commonly employed to support revocation management is using proxy re-encryption (PRE) [3, 26-33]. PRE was initially introduced by Mambo and Okamoto [26]. They proposed a technique that uses a concept of delegator to perform re-encryption of the ciphertext sent by the originator. In this scheme, the delegator learns neither the decryption keys nor original plaintext. Generally, the PRE scheme can be classified into two major solutions: traditional PRE and outsourcing PRE.

First, The traditional PRE schemes [6,7,8] were adopted by several both identity based encryption (IBE) and attribute-based encryption (ABE) in supporting the revocation. However, the traditional PRE focuses only on delegating the re-encryption task to the proxy. The data owner or client needs to bear the computation cost for re-encryption key generation and new user key re-distribution. In [3], Yu et al. proposed KP-ABE based access control for data outsourced in the cloud and employed PRE to support attribute revocation. With this scheme, the computation costs including file re-encryption and key update are outsourced to the proxy in cloud server. However, this scheme is based on KP-ABE of which the data owner has no control over the policy. In 2010, the authors extended their scheme [32] to integrate the PRE with CP-ABE. The extended scheme lets the proxy perform both file re-encryption and user key update. However, the proxy deployed is assumed to be honest in their scenario. Hence, it may be risky for real deployment as the proxy is physically located at the cloud.

Second, recent IBE [30] and ABE [29-31] approaches considered that the cost at client in computing the re-encryption key should be optimized by offloading it to the proxy as well so that the client can save the communication and computation cost. This solution is classified as an outsourcing PRE approach. For example, K. Liang et al. [30] introduced re-encryption key update process to be done at a cloud. To do so, the public key generator publishes a constant-size public string at the beginning of each time period. If there is any user revoked, the proxy re-encrypts the ciphertext to the next time period. Hence, revoked users can not decrypt with their existing keys. However, this approach does not support attribute revocation, which is a finer revocation level.

In [27, 31], the techniques for outsourcing the re-encryption key generation for supporting revocation have been proposed. In [27], Tysowski et al. designed a protocol based ABE and PRE for supporting secure data sharing in mobile cloud computing. The PRE is used to reduce the computation workload on the data owner side. Key generation is divided between mobile data owner and trusted authorities. In addition, the authors introduced versioning array and key sharing technique to dynamically support when there is a change on member of mobile users. However, this paper tackles only the user revocation level and the re-encryption key generation is partially outsourced.

In [31], Jason et al. proposed the outsourcing conditional PRE (OC-PRE) scheme with a focus on reducing overhead at client sides in dealing with the initial setup stage and at the decryption of each message from the client to the cloud. In this scheme, the originator computes the conditional value and re-encryption keys and sends to the proxy in cloud to perform re-encryption process. If there is a change (e.g., revoke user) on membership, the originators or clients just re-compute the conditional value changing key (CCK) and the CCK will transform the ciphertext with a new conditional value. This reduces the re-encryption key generation significantly. However, the paper only focuses on the change of user level such as user revocation while an attribute revocation level requires more practical justification and it has not been discussed.

Actually, the existing revocable PRE approaches have not segregated the revocation level between user and attributes. They only consider the re-encryption is a major cost for the revocation. In addition, user decryption key update is also another overhead to be concerned for attribute revocation but this issue is generally overlooked. Besides, the existing works have not provided the practical implementation to show how much the PRE improves the revocation performance. We therefore propose a very lightweight PRE (VL-PRE) to incorporate into our C-CP-ARBE in order to optimize the computational costs caused by attribute revocation and policy update. The detailed solution of VL-PRE is presented in chapter 4.

2.3 Secure Policy Update Management

In the CP-ABE based access control environment, the evolution of access control elements such as change of users or access policy can be generally occurred anytime. Specifically, if there is an update of a policy, the data owner needs to download the encrypted files encrypted with the before-updated policy and re-encrypt them with a new policy. Then, they are uploaded back to the cloud. This introduces the communication and computation cost at data owner. Unfortunately, policy updates in CP-ABE scheme has got less attention by existing research works. Most research works assume that if there is any change of policy, the straightforward update can be applied. However, the overheads causing from policy update will be more substantial high and will degrade the quality of service if there are frequent changes. There are a few approaches so far that propose the specific solution for coping the policy update in CP-ABE.

In [59], the authors introduced a ciphertext delegation method to update the policy of ciphertext in attribute-based access control. Their method can be applied to handle user revocation and policy update based on a re-encryption delegation technique. If there is any change of policies, the re-encryption algorithm is performed to re-encrypt the data with the new policy. Nevertheless, the performance on updating the ciphertext over the complex access policy was not examined by the authors. Recently, Kan Yang et al. [34, 60] proposed a method to outsource a policy update to the cloud server. They proposed policy updating algorithms for adding and removing attributes in the AND, OR, and threshold gate of LSSS policy. They proposed a scheme that is able to update ciphertext instead of re-encrypting the file. However, cost for ciphertext update is linear to the number of attributes updated over the access structure. Besides, the authors have not discussed how updated polices are maintained.

Since the method that deals with the update of ciphertext tightly works on the CP-ABE encryption itself, the complexity and cost of the update are linear to the number of attributes and update operations. Therefore, this method is not applicable well for the unpredicted and frequent policy updates. Another solution is to use a delegated proxy to perform re-encryption. Therefore, in addition to support the revocation problem, PRE concept has been employed to support policy update as well.

For example, in [28], the authors introduced adaptable CP-ABE scheme to handle policy changes in CP-ABE encryption for data outsourced in cloud computing. In this scheme, a trapdoor is generated from the central authority and it is used to transform a ciphertext under one access policy into ciphertexts under any other access policies. This scheme allows the data owner to outsource ciphertext re-encryption task to the proxy, while the proxy cannot learn the content from the plaintext encrypted. However, the trapdoor generation is still the computation burden that the authority has to compute every time whenever the policy is updated.

In [29], Yukata Kawai proposed a flexible CP-ABE proxy re-encryption scheme by combining key randomized and encrypted methodology and adaptive CP-ABE. The proposed scheme focuses on reducing the computation cost at client side by outsourcing the re-encryption key generation to cloud server. The universal re-encryption key (urk) is proposed to be used together with the decryption key (Sk_s) for generating the re-encryption key. The decryption key is protected by randomized parameters and sent to the cloud for computing the re-encryption key. Importantly, Kawai's approach is the first attempt dealing with the outsourcing concept of re-encryption key generation in PRE setting. However, the author did not perform the performance evaluation to substantiate the efficiency of the proposed scheme. Furthermore, the complexity of the scheme is still problematic for a big size of policy.

However, the proposed schemes [28, 29] only provide the security function while the implementation result and performance have not been discussed. Therefore, the efficiency and practicality of the proposed CP-ABE proxy re-encryption in handling the policy changes cannot be justified. In addition, their approaches have not provided the criteria to validate a newly updated policy before it is used to re-encrypt the data.

In [35], we proposed PRE scheme that fully outsources re-encryption key generation to the proxy; the computation cost at data owner is minimized. However, if there are frequent revocations or policy updates, re-encryption key needs to be re-generated in all change events and data owners require to prepare and submit data package to the proxy for computing the re-encryption key.

In essence, PRE allows for a third party to do the computational work to re-encode encrypted files after a policy change. While desirable from a client perspective, it is still non-ideal to have a high proxy-side cost to handling policy updates, particularly if policy updates are frequent.

To the best of our knowledge, existing normal PRE schemes are not practical to support policy updates in large-scale data outsourcing environment where the access control elements may be changed frequently. This is because cost for re-encryption key generation is unpredictable at the data owner side. However, offloading too much computation cost to a proxy may introduce the delay for re-encryption task and thus cause efficiency problem. Besides, this strategy is also not advisable for the cloud model that the cloud provider charges the fee based on CPU usage. Thus optimizing both setup cost at data owner side and re-encryption cost a cloud side is a real challenge. Unfortunately, this computation optimization aspect has not been addressed by the existing PRE schemes. We therefore propose a policy update algorithms and employ VL-PRE proposed in chapter 4 for optimizing the computational costs in both data owner side and cloud side caused by policy updates. The details of the proposed method are discussed in chapter 5.

Chapter 3: Our proposed Access Control Model: C-CP-ARBE

This chapter is organized in two major parts: the theoretical background and our proposed access control model. In the first part, CP-ABE and RBAC which are the key theories used to construct our model are presented. For the second part, we give definitions of our access control model, key management, and the construction of our cryptographic algorithms. This chapter is the extension of our previous paper [43-45].

3.1 Technical Preliminaries

In this section, we present the theoretical background of CP-ABE and RBAC which are the core constructs of our proposed access control model.

3.1.1 Ciphertext Policy-Attribute-Based Encryption (CP-ABE)

Bethencourt et al. [16] proposed the ciphertext policy attribute-based encryption as an extension of attribute-based encryption (ABE) for the access control. Basically, the concept of cryptographic construction for both ABE and CP-ABE is based on the bilinear maps.

The following describes the formal definition of bilinear maps.

Bilinear Maps

Let G_1 and G_2 be two multiplicative cyclic groups of prime order p and e be a bilinear map,

$e : G_1 \times G_1 \rightarrow G_2$. Let g be a generator of G_1. Let H: $\{0,1\}^* \rightarrow G_1$ be a hash function that the security model is in random oracle.

The bilinear map e has the following properties:

1. Bilinearity: for all $u, v \in G_1$ and $\mathbb{Z}_p$, $e(u^a, v^b) = e(u, v)^{ab}$

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

Definition 1: Let a set $\{P_1,P_2,\ldots,P_n\}$ be given. A collection $\mathbb{A} \subset 2^{\{P_1,P_2,\ldots,P_n\}}$ is monotone if

$\forall B,C : if\ B \in \mathbb{A}\ and\ B \subset C \longrightarrow C \in \mathbb{A}$.

An access structure is a monotone collection $\mathbb{A}$ of non-empty subsets of $\{P_1,P_2,\ldots,P_n\}$, i.e. $\mathbb{A} \subset 2^{\{P_1,P_2,\ldots,P_n\}}\setminus\emptyset$.

In the context of CP-ABE, we consider a collection of the set of attributes. Thus, an access structure $\mathbb{A}$ consists of sets of attributes.

Access Tree T [16]. Given an access structure $\mathbb{A}$, we define an access tree on $\mathbb{A}$. Each non-leaf node represents a threshold gate associated by its children and a threshold value. Furthermore, each leaf node is associated with an attribute. We denote that an attribute can be either regular name, number, or a mix of both types by using comparison operators (=, >, >=, <, <=).

Definition 2: A threshold gate is a tree node whose associated value is defined as follows. Let num_x be the number of children of a node x and k_x be its threshold value ($0 < k_x \leq num_x$). When $k_x = 1$, the threshold gate represents an OR gate and when $k_x = num_x$, it represents an AND gate. The k-outof-n threshold gate is also allowed in $T\,\mathbb{A}$, in this case $k_x = k$ and $num_x = n$.

Each leaf node x of the tree is associated with an attribute and its value.

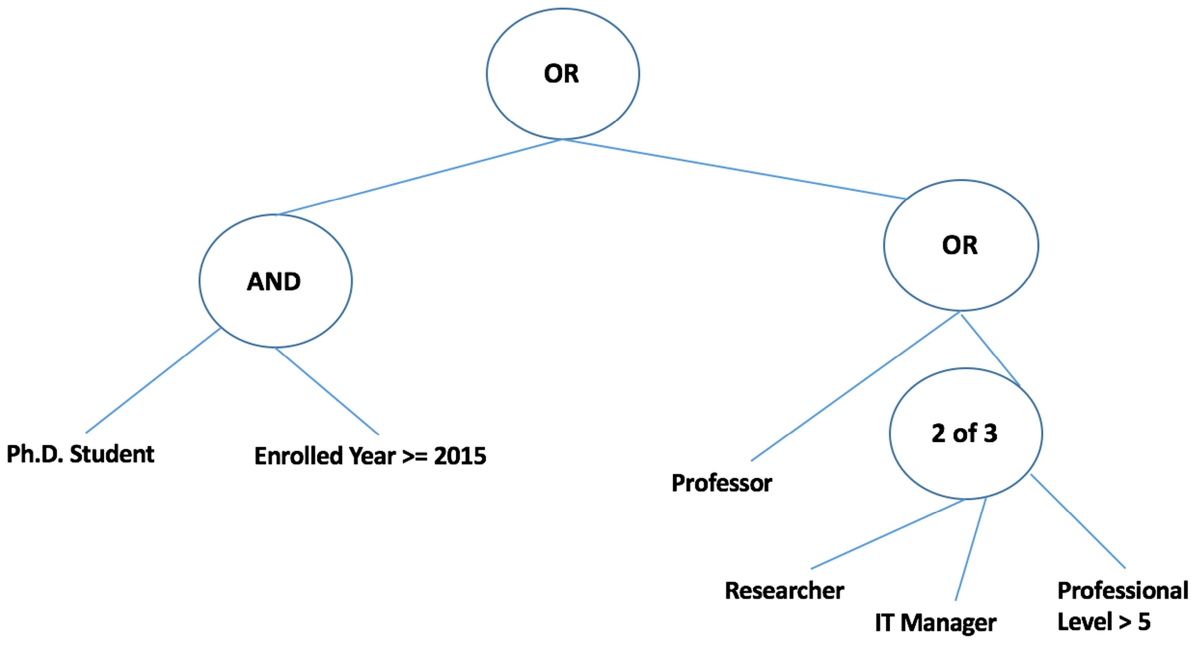


Figure 3.1: Example of CP-ABE Access Tree

Figure 3.1 shows an example of access tree constructed from a set of attributes built from {Ph.D. Student, Enrolled Year, Professor, Researcher, IT Manager, Professional level}. The nodes of OR, AND, OR, and "2 of 3" are threshold gates. The access tree above contains as nodes Ph.D. Student, Enrolled Year >=2015, Professor, Researcher, IT Manager, Professional level >5, and the threshold gates mentioned before.

In CP-ABE, a message is encrypted by using a policy which is an access tree constructed from a set of attributes. A user is able to decrypt a ciphertext only if she has a secret key that is constructed from a set of attributes *S* that satisfies the policy.

CP-ABE[16] is composed of four fundamental algorithms as follows:

Setup. This algorithm takes initial parameters as input. It generates the public parameter *PK* and a master key *MK*.

Encrypt(*PK, M*, *T)* . This algorithm takes as input the public parameter *PK*, message *M*, and access tree *T*. The message *M* will be encrypted with public parameter *PK* and the access tree *T*. It outputs a ciphertext *CT*.

KeyGeneration(*MK*, *S*) This algorithm takes the master key *MK* and a set of attribute *S* describing the key to be produced. Then, it produces the decryption key (private key or secret key) *SK* which is used in decryption.

Decrypt(*PK, CT, SK*). This algorithm takes as input the public parameters *PK*, a ciphertext *CT*, and a user's private key *SK*. If *CT* is produced by using an access tree *T*, it will be decrypted if the set of attributes *S* in the private key *SK* satisfies the access tree *T*.

Because CP-ABE possesses the encryption feature with the flexible use of attributes for the access authorization decision, it is thus adopted by several works [3,7, 9, 10] that develop the access control solution for data outsourced in cloud storage systems.

In CP-ABE, the encryption performance is subject to the policy size and size of data to be encrypted, while the decryption efficiency is subject to the size of a key which grows linearly in the number of attributes.

3.1.2 Role-Based Access Control Model (RBAC)

RBAC is an access control model that provides the relationships of user's role, permission, and objects or resources.

Definition 3: RBAC is a tuple of (*U, R, P, UA, PA, RH*) where**:**

(1) *U, R, P* are given sets that represent a set of users, roles, and permission, respectively.

(2) $UA \subseteq U \times R$, a many-to-many user-to-role assignment relations;

(3) $PA \subseteq P \times R$, a many-to-many permission-to-role assignment relation;

(4) $RH \subseteq R \times R$ is a partial order on *R* representing a role hierarchy.

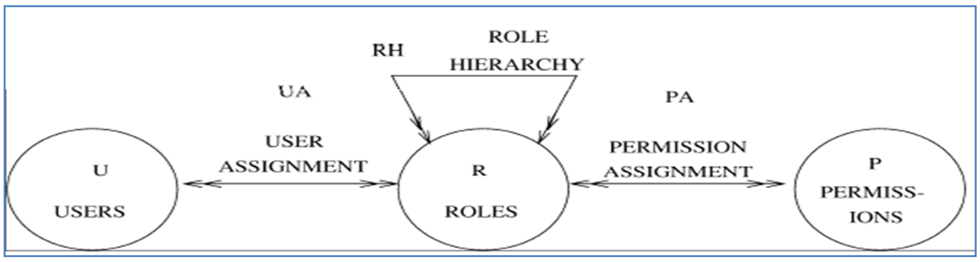


Figure 3.2: Hierarchical RBAC [24]

Figure 3.2 shows the hierarchical RBAC [24] which is generally used to refer to RBAC. The model supports role hierarchy where the role can be structured in hierarchy according to the position of the users in their organization. The top role is on the top of hierarchy while the less-powerful rule lays on the bottom. Usually, the permission of lower roles is inherited to the roles above them. Besides, the constraints are allowed to be modelled and are enforced to the access session in RBAC. In the typical RBAC model, the role and privilege can be flexibly specified and enforced based on the organization's access control policy. Generally, RBAC delivers expressive and fine-grained access control specification as it specifies what user can perform what action over the given resource. Also, it increases the scalable management of a number of users in the organization since a group of user is assigned to a specific role; it thus far more flexible in defining the policy onto the role rather than onto individual user. However, applying RBAC directly to cloud environment may not be practical since users accessing resources may be unknown and may come from different organizations and the access control model needs to be cryptographic-embedded.

3.2 Our Proposed C-CP-ARBE Model

In this section, we first give the definitions of the system model, access control policy, and user decryption key. Then, we describe the cryptographic protocol in overall, and provide details of algorithms, and discuss security analysis. Finally, the analysis of the C-CP-ARBE is given.

3.2.1 Extension of Access Tree

To integrate the RBAC model into CP-ABE to enhance the expressiveness and scalability, we define attribute-role based (A-RBAC) access control model as follows:

Definition4: A-RBAC is a tuple of ($U, R, P, UA, RH, D, APA, Attr, C$) where:

- U, R, P, UA, RH are as in the RBAC model,
- $D \subseteq R \times Attr$ is a many-to-many permission-to-role assignment relation.
- $APA \subseteq Attr \times P$

We define $PA \subseteq R \times P$ as for r $\in R$, p $\in P$, $PA(r,p)$ iff there exists an attribute a such that $D(r,a)$ and $APA(a,p)$. Hence, A-RBAC is an extension of RBAC.

In cloud computing environment, a user u ($\in U$) is an entity to whom assigned a specific role r_u ($\in R$) and she requests to access the resource with a permission p ($\in P$) i.e. read or write. Attributes $Attr$ are a set of attributes used to characterize a role r_u. A set of attributes is issued by attribute authority AA.

- C is RBAC constraint. Definitions 5 , 6, and 7 give examples of the constraints employed in our access control model.

Definition 5: Separation of Duty Constraint over Role (**SoD-R Constraint**). SoD-R is the constraint over a role set R which specifies that the exclusive roles cannot be assigned to the same user at the same time.

Let U, R be a user set, and role set respectively. Let r_1 and r_2 be two roles $\in R$ such that $r_1 \neq r_2$. We also define predicate UserRoleAssign(u,r) that means to $u \in U$ is assigned $r \in R$. $(r_1, r_2) \in$ SoD-R if the following condition holds:

UserRoleAssign $(u, r_1) \rightarrow \neg$ UserRoleAssign (u, r_2)

Example 1: Let U = {Alice, Bob}, R = {doctor, nurse}. SoD-R(doctor, nurse) is satisfied only if

UserRoleAssign (Alice, doctor) → ¬UserRoleAssign (Alice, nurse).

UserRoleAssign (Alice, nurse) → ¬UserRoleAssign (Alice, doctor).

UserRoleAssign (Bob, doctor) → ¬UserRoleAssign (Bob, nurse).

UserRoleAssign (Bob, doctor) → ¬UserRoleAssign (Bob, nurse).

In our model, the SoD-R constraint helps prevent the occurrence of conflicting roles; that is the conflicting roles cannot be assigned to the same user of the same authority at the same time.

Definition 6: Spatial Constraint. A Spatial Constraint SC is a combination of one or more sets of locations (L).

Location L can be either logical location such as IP address, or organization name, or address of the place specified as an attribute.

Example 2: Any doctors from Hospital A (HosA), Hospital B (HosB), or Hospital C (HosC) can login to the system S.

Constraint expression: $\forall\ u\,(\text{UserRole}\,(u,\ \text{doctor})\ \wedge$

Workedin(u,HosA ∪ HosB ∪ HosC)→Login(u,S).

Definition 7: Temporal Constraint. A Temporal Constraint TC specifies the durations in which a role can be enabled or disabled. TC is a tuple of two elements (I, E), where $I = (t_1,, t_2)$ denotes an interval and E denotes an event that is enabled in the interval I.

Example 3: From 8.00 am to 5.00 pm. is a working time of DayDoctor.
Constraint expression: ((8.00 am, 5.00 pm), enable *DayDoctor*)

Our constraint modeling is expressive and able to support general RBAC constraints in SoD, spatial, and temporal constraint in attribute-based access control model used in the cloud. In our scheme, constraints are deployed in a cloud server as separate execution files and they are signed by the data owner.

Definition 8: Extended Access Tree

We extend the definition of Access Tree so that each non-leaf node has an attribute and threshold gate. The threshold gate rule is the same as an access tree of CP-ABE.

Example 4: Let a set of roles {nurse, MD} be given. Figure 3.3 illustrates collaborative data access control for a hospital information system where professional roles: nurse and medical doctor (MD) are allowed to access the outpatient department (OPD) treatment file stored at a cloud storage. It is assumed that the OPD file is created and shared by Hospital A. Only the users allowed by the Hospital A can access the file.

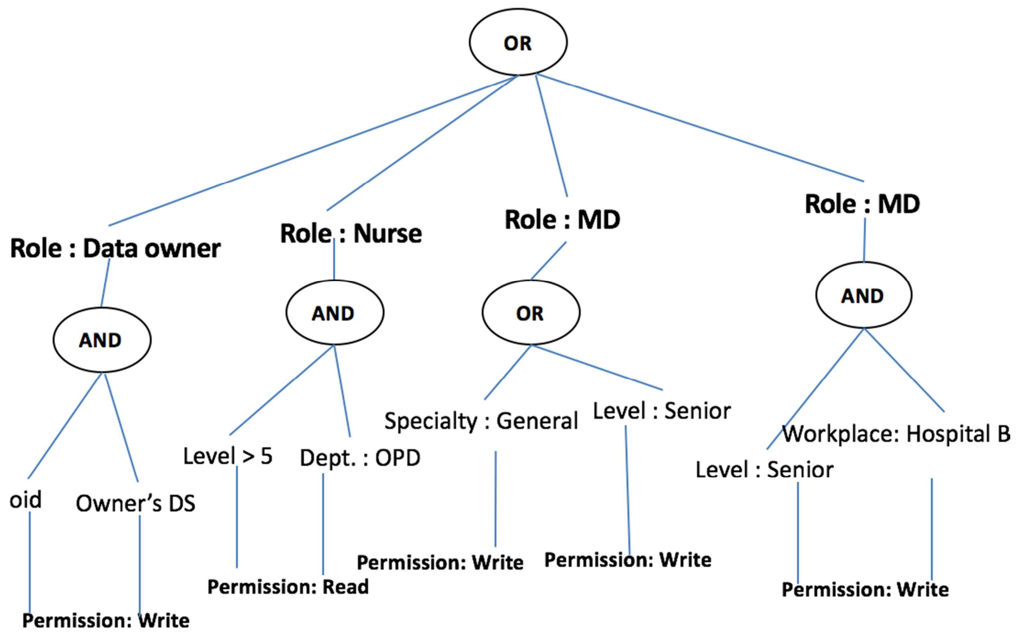


Figure 3.3: Access Tree Structure – OPD treatment File

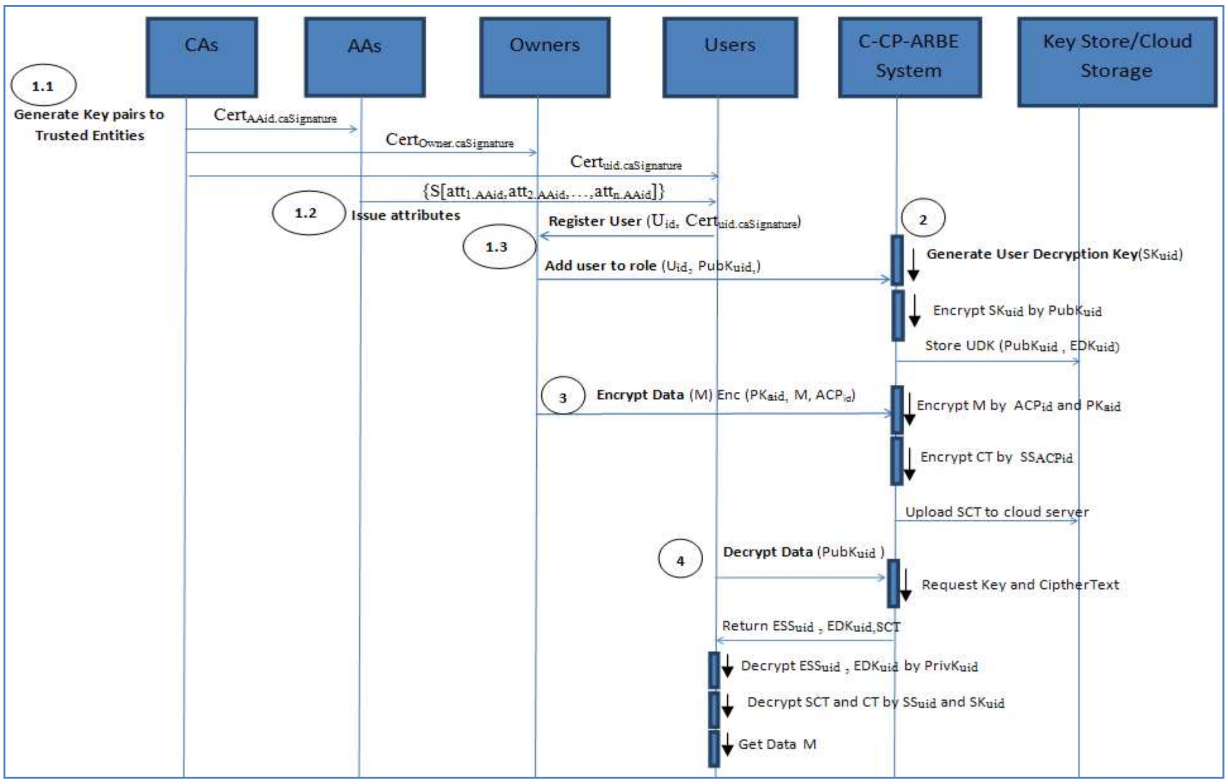
As of the example policy, the OPD file is shared to the users whose roles are data owner having his ID (oid) and digital signature (Owner's DS), nurse having level greater than 5 and belonging to the OPD department, and medical doctor (MD) whose specialty is General or level is Senior. Also, the file is allowed to be accessed by a senior MD from another partner hospital, According to this access tree, only data owner and role MD are allowed to update (write) the OPD treatment file, while the role nurse can read the file. In our model, role data owner is assigned as a default role that is included in all ACPs generated by the data owner *oid*. For the attribute Owner's DS, it is produced from using data owner's private key to sign his attribute *oid*. As the write access requires data to be re-encrypted, we will describe our mechanism in supporting write access in the Section 3.4 of this chapter.

3.2.2 System Model

3.2.2.1 Overview

In this section, we describe our system model based on our proposed cryptographic process of C-CP-ARBE scheme. C-CP-ARBE is characterized as Multi-Authority CP-ABE. We use attribute authority identifier (aid) to identify the authority who issues the attributes to users. Each user *uid* who is issued the attributes by the attribute authority *aid* is identified with *uid.aid*.

Figure 3.4 shows the protocol diagram of our proposed system model. It describes the typical cryptographic processes among the trusted entities and our proposed C-CP-ARBE system. In overall, the system environment consists of the following entities. More details of each cryptographic process will be explained in Section 3.2.2.3.



CAs
AAs
Owners
Users
C-CP-ARBE System
Key Store/Cloud Storage
1.1
Generate Key pairs to Trusted Entities
Cert_AAid.caSignature
Cert_Owner.caSignature
Cert_uid.caSignature
{S[att_1.AAid, att_2.AAid, ..., att_n.AAid]}
1.2
Issue attributes
Register User (U_id, Cert_uid.caSignature)
1.3
2
Generate User Decryption Key(SK_uid)
Add user to role (U_id, PubK_uid)
Encrypt SK_uid by PubK_uid
Store UDK (PubK_uid, EDK_uid)
3
Encrypt Data (M) Enc (PK_aid, M, ACP_id)
Encrypt M by ACP_id and PK_aid
Encrypt CT by SS_ACPid
Upload SCT to cloud server
4
Decrypt Data (PubK_uid)
Request Key and CiptherText
Return ESS_uid, EDK_uid, SCT
Decrypt ESS_uid, EDK_uid by PrivK_uid
Decrypt SCT and CT by SS_uid and SK_uid
Get Data M

Figure 3.4: System Model Describing C-CP-ARBE Cryptographic Process

1. Certificate Authority (CA) is the trusted party who signs and issues the public key certificate (X.509 certificate) to all entities including users, AAs, data owners, system agents. We assume that CA is operated by a trusted PKI system. Here, data owners, AA, and users generate key pairs and submit public key to CA for signing and issuing the certificate. The key pairs and certificate are used to sign transactions, encrypt secret key, and authenticate the individual or systems. In our framework, entities e.g. users are allowed to use certificates from different CAs as long as they are in the trusted list agreed by the collaborative data owners.

2. Attribute Authority (AA) is an entity that issues, revokes, and updates users' attributes corresponding to their roles of the particular domain. Multiple AAs are allowed. AA is responsible for generating public attribute keys for all attributes belonging to the AA and for issuing the decryption key to users. Typically, user's belonging organization works as an AA. For example, Hospital A and Hospital B may be the AA that issues the attributes to their users (doctors, nurses, staffs, etc.). In collaborative cloud storage systems with multiple AAs, the resources or data can be shared over the cloud among users in different domains. For instance, a doctor from Hospital B may need to access the diagnosis records shared by Hospital A for supporting his treatment plan for the patient having records in Hospital A.

3. Data owner creates files and initially uploads its data in the encrypted form to the cloud server. He also specifies their access control policy to regulate how users access the particular resource and what privilege they have over the resource. Data owner is usually departments or functional units in an organization such as the accounting department in Hospital A.

4. User is an entity that requests to access (read or write) data outsourced in the cloud. To each user, a set of attributes is issued by AAs and their attributes are assigned with respect to his/her role and registered to data owners.

5. C-CP-ARBE is our proposed cryptographic-based access control system stored in the cloud. It is exploited to support user authentication, data encryption/decryption, user and attribute revocation, and policy update management. The details C-CP-ARBE model are described in subsequent sections.

6. Cloud Server is a cloud storage where data files and user decryption keys are stored in the encrypted format.

3.2.2.2 Key Management

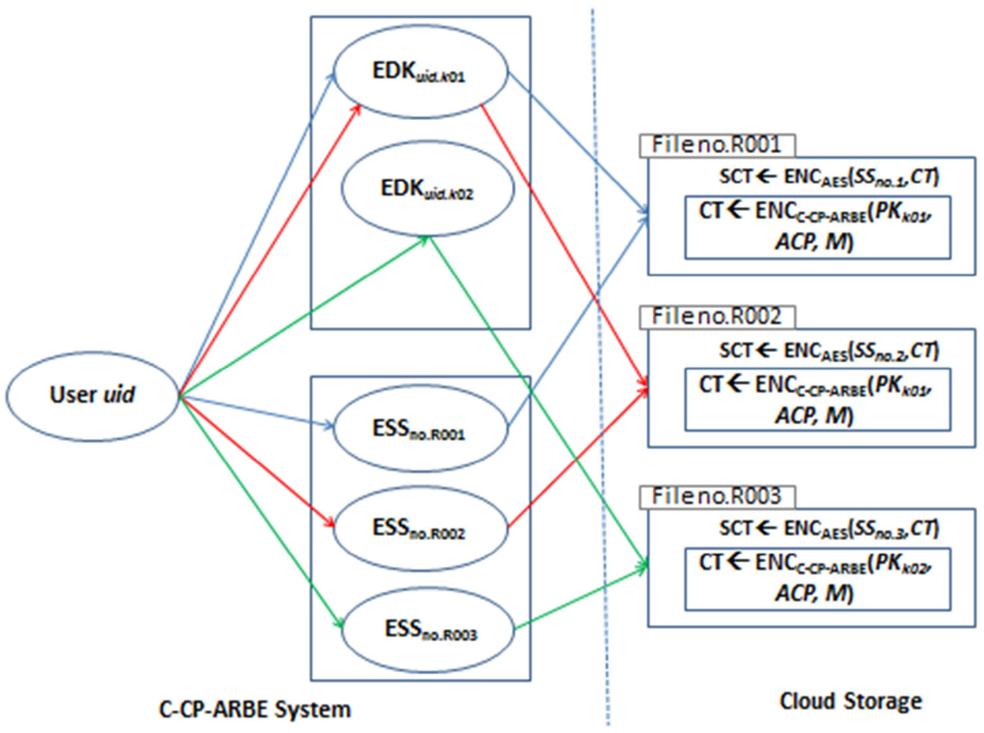
In our system, there are three major groups of user key: PKI key pair, user decryption key, and symmetric key. First, PKI key pair is a pair of public key and private key issued by the certification authority (CA). AAs, data owners, and users need to register X.509 certificate from the CAs for authentication, key encryption, and digital signing. To a given user, one's X.509 certificate is issued. We also use each user public key to encrypt the corresponding user decryption key. Then, the encrypted key is stored in the cloud. User's private key is used to sign the transaction such as data and policy update.

Second key type is user decryption key (UDK) which is generated by attribute authorities (as of CP-ABE scheme) and it is delivered to a user when she makes a request to access the resource. UDK is constructed from a set of user attributes and it is used to decrypt the ciphertext (CT) that is encrypted by the C-CP-ARBE method. The detail of C-CP-ARBE encryption is depicted in Section 3.2.2.3.

Last key type is symmetric encryption key called secret seal (SS). SS is used with AES algorithm to encrypt the ciphertext to produce the seal ciphertext (SCT) in an additional layer of encryption. Basically, each UDK and SS are encrypted by user's public key. Then, the encrypted UDK (EDK) and encrypted secret seal (ESS) are obtained and stored in the cloud. Hence, users only keep their private key used to decrypt EDK and ESS in the system and do not need to hold multiple UDKs or SSs if they have several ones.

To reduce key distribution cost in CP-ABE, we introduce user decryption key graph (UDKG). A UDKG stores ESS and EDK of each user. A user uses her private key to decrypt ESS and EDK to obtain the SS and UDK respectively. UDKG is used to manage the decryption keys so that a user can use the latest version of ESS and EDK for decrypting each corresponding ciphertext. UDKG also supports the dynamic key retrieval for accessing to multiple files of which they are encrypted by different UDK or SS keys. Each file (ciphertext) has a reference no. bound to the keys used to decrypt it. C-CP-ARBE will dynamically return the keys matching the files for each access to individual user. Because a UDKG is managed by C-CP-ARBE, user decryption keys are not needed to be distributed to users and this provides zero key distribution cost.

Figure 3.5 presents UDKG that consists of four vertices: user, EDK, ESS, and corresponding ciphertext.



EDK$_{uid.k01}$
EDK$_{uid.k02}$
User *uid*
ESS$_{no.R001}$
ESS$_{no.R002}$
ESS$_{no.R003}$
Fileno.R001
SCT← ENC$_{AES}$(*SS$_{no.1}$*, *CT*)
CT← ENC$_{C\text{-}CP\text{-}ARBE}$(*PK$_{k01}$*, *ACP, M*)
Fileno.R002
SCT← ENC$_{AES}$(*SS$_{no.2}$*, *CT*)
CT← ENC$_{C\text{-}CP\text{-}ARBE}$(*PK$_{k01}$*, *ACP, M*)
Fileno.R003
SCT← ENC$_{AES}$(*SS$_{no.3}$*, *CT*)
CT← ENC$_{C\text{-}CP\text{-}ARBE}$(*PK$_{k02}$*, *ACP, M*)
C-CP-ARBE System
Cloud Storage

Figure 3.5: UDK Graph (UDKG)

3.2.2.3 Construction of C-CP-ARBE

Our proposed cryptographic process of C-CP-ARBE scheme is based on CP-ABE where the bilinear map is a major construct of the key generation protocols. In our scheme, users who obtain the user decryption key from an AA can access the encrypted data with their permission (read or write) specified in the access policy. In the collaborative access control enforcement, the access control policies are allowed to specify the access rules that contain some attributes issued by different authorities, if necessary.

C-CP-ARBE is constructed from five major phases: System Setup, Key Generation, Encryption, Decryption, and Revocation. Table 3.1 present notations that we use in our algorithms.

Table 3.1 Notations and Its Description Used in Our Proposed Algorithms.

Notation	Description
$S_{uid.k}$	Set of all attributes issued to user *uid* and managed by authority *k*.
SK_k	A secret key which belongs to authority *k*.
PK_k	Public key which belongs to authority *k*.
$PK_{x.k}$	Public attribute key issued by authority *k*.
GSK_{uid}	A global secret key of a user *uid*. GSK is a private key issued by the certification authority CA.
$Cert_{uid}$	A public key certificate containing user's public key issued by a certification authority CA.
$(PubK_k, PrivK_k)$	A PKI Key pair consisting of public key and private key issued by CA. This key pair is issued to the attribute authority *k*.
R_{id}	A role (identified with *id*) available in the system.
UL	A database that contains a set of user list of each role ($UL_{R_{id}}$)
$UDK_{uid.k}$	User Decryption key issued by authority *k*
$EDK_{uid.k}$	EDK is an encrypted form of a UDK which is encrypted by a user public key.
GRP	Group role parameter is list of user lists for all roles.
SS	Secret seal is a symmetric key created from the AES algorithm together with the GRP.
ACP	An extended access tree (referred to as an access control policy) used to encrypt the data files.
SCT	A sealed ciphertext which is a ciphertext encrypted with the SS

Let G_1 be a bilinear group of prime order p, and let g be a generator of G_1.Also, let e: $G_1 \times G_1 \rightarrow G_2$ denote the bilinear map. In addition, we define that the lagrange coefficient $\Delta_{i,S}$ for $i \in \mathbb{Z}_p$and a set S of elements in $\mathbb{Z}_p$: $\Delta_{i,S}(x) = \prod_{j \in S, j \neq i} \frac{x-j}{i-j}$. Furthermore a hash function H:$\{0,1\}^* \rightarrow G_1$ is employed as model in random oracle. The function $att(x)$ is defined only x is a leaf node and denotes the attribute associated with the leaf node x in the tree.

Phase 1: System Setup

This phase consists of five algorithms. In our system, the authority run CreateAttributeAuthority algorithm while other algorithms in this phase are run by a data owner.

1. **CreateAttributeAuthority**(k)→PK_k, SK_k, $PK_{x.k}$. This algorithm takes an attribute authority ID(k) as input. It outputs the authority public key (public parameter) PK_k, SecretKey (SK_k), and public attribute keys ($PK_{x,k}$) for all attributes issued by the A_k.

   The public key and secret key of each AA A_k are computed as follows.
   Let S_{A_k} be a set of all attributes issued and managed by the authority A_k.

The algorithm takes a generator g_k of G_1 and two random numbers $\alpha_k, \beta_k \in \mathbb{Z}_p$. Then the AA's Public Key (PK_k) is computed as

$$PK_k = (G_1, g, h = g^{\beta}, e\,(g_k, g_k)^{\alpha_k}).$$

The corresponding AA's Secret Key SK_k is $(\beta_k, {g_k}^{\alpha_k})$.

2. **CreatRole**(SK_k, R_{id},) → UL_{Rid}. The CreateRole algorithm takes as inputs attribute authority's secret key (SK_k) , Role (R_{id}), It returns user list (UL_{Rid}) of users who qualify to the role R_{id} and stored in the database UL.
3. **UserRegister**(*uid*, $Cert_{uid}$). The UserRegister algorithm takes as input userID (*uid*) and user's certificate ($Cert_{uid}$), then it updates UL so that if *uid* has a role R_{id}, then *uid* is contained in UL_{Rid}.
4. **CreateGrouproleParameter**GRP()→ GRP. The Create GroupRole parameter algorithm first collects all user list UL_{Rid} and concatenates them.
5. **CreateUDKG**(*uid*)→UDKG. The algorithm takes *uid* authorized to use the resource in the authority domain. It outputs the UDKG for *uid*. UDKG initially consists of empty EDK and empty ESS.

Phase 2: Key Generation

This phase is run by an AA and it consists of two algorithms as follows:

6. **UserKeyGen**($S_{uid,k}$,*uid*, SK_k, $Cert_{uid}$)→ $EDK_{uid,k}$. The UserKeyGen algorithm consists of two sub-modules:
   (1) UDKGen. It takes as input a set of attributes ($S_{uid,k}$) that describes the *uid*'s user decryption key, *uid*, attribute authority's secret key (SK_k), and $Cert_{uid}$ of user *uid* issued by a CA, then it returns the set of user decryption key (UDKs).

For each user *uid*, the AA A_k chooses a random r *and* $r_j \in \mathbb{Z}_p$, for each attribute $j \in S_{A_k}$. Then the user decryption key *(*UDK$_{uid.k}$*)* is computed as:

UDK$_{uid.k}$ = $(D = g_k^{(\alpha_k+r)/\beta_k}, \ (D_{j_1} = g_k^{\ r} H(j_1)^{r_j}), \ldots, (D_{j_n} = g_k^{\ r} H(j_n)^{r_j}), \ (D'_{j_1} = g_k^{\ r_j})), \ldots, (D'_{j_n} = g_k^{\ r_j}))$.

(2) EDKGen. The algorithm takes a UDK$_{uid.k}$ as an input to be encrypted with $Cert_{uid}$ and outputs the set of encrypted decryption key $EDK_{uid,k}$. The encryption function is expressed as: EDKGen($UDK_{uid,k}$) → ENC$_{RSA}$($Cert_{uid}$, $UDK_{uid,k}$) ≡ EDK$_{uid,k}$

Phase 3: Encryption

The encryption function is either performed by data owners or users who have the write access permission. This phase performs two-layer encryption including inner encryption based on CP-ABE and outer encryption based on symmetric encryption. Table 3.2 defines the encryption function used in our system as follows:

Table 3.2: Encryption Functions Used in C-CP-ARBE

Encryption function	Meaning
ENC$_{\text{C-CP-ARBE}}$	An encryption function which uses CP-ABE method to encrypt data or message *M*. This is the inner layer.
ENC$_{\text{RSA}}$	A public key encryption function which uses RSA algorithm to encrypt the $UDK_{uid.k}$, and *SS*
ENC$_{\text{AES}}$	A symmetric encryption function which uses AES algorithm to encrypt the ciphertext. This is the outer layer.

7. **ENC**(PK_k, *SS*, *M*, *ACP*, $Cert_{uid}$)→*SCT*. The encryption algorithm performs two consecutive steps as follows:

   (1) Encrypt Message *M*

   $M \mapsto$ ENC$_{\text{C-CP-ARBE}}$(PK_k, *ACP*, *M*) ≡ *CT*

The algorithm takes as inputs authority public key (*PK_k*), access control policy (*ACP)*, and message (*M*). Then it returns a ciphertext *CT*. First, it chooses a random $s \in \mathbb{Z}_p$. The algorithm initially calculates a polynomial q_x of degree $d_x = k_x - 1$, the threshold, for each node x of *ACP*. This calculation starts from the root node R_T. We choose random polynomial of degree d_R but require $q_{R_T}(0) = s$. Let a polynomial q_x be calculated. For node x for a child y of x, it chooses a random polynomial of degree d_y but $q_x(0) = q_{parent(x)}\left(index(x)\right)$,where $index(x)$ returns number associated with the node x. By using the fact that a polynomial of degree d is determined by d+1 values, the requirement on the children determines the polynomial of the parent. We call this calculation the polynomial interpolation.

Let *Y* be a set of leaf nodes in *ACP;* Y={y_1,…, y_n}. The ciphertext is then computed as follows:

$$CT = (ACP,\ \check{C} = M\ e(g_k, g_k)^{\alpha_k s}, C = h^s,\ C_{y_1} = g_k^{q_{y1}(0)}, \dots, C_{y_n} = g_k^{q_{y_n}(0)},\ C'_{y_1} = H(att(y_1))^{q_{y_1}(0)}, \dots, C'_{y_n} = H(att(y_n))^{q_{y_n}(0)}).$$

(2) Encrypt *CT*

The algorithm used to encrypt the ciphertext is defined as:

$$CT \mapsto \mathrm{ENC}_{\mathrm{AES}}(SS, CT) \equiv SCT$$

The cipthertext is sealed (encrypted) by the symmetric encryption algorithm by using secret seal (*SS*). *SS* is calculated from the hash value of *GRP*.

Using *SS* for encryption enables the revocation of user to be done with less cost since when there is a user revoked from the system; only *GRP* is re-calculated resulting a new generation of *SS*. The data file is not required to be re-encrypted.

As the final encryption step, *SS* is encrypted with *$Cert_{uid}$* and the encrypted secret seal (ESS) is produced as follows:

$$SS \mapsto \mathrm{ENC}_{\mathrm{RSA}}\ (Cert_{uid}, SS) \equiv ESS$$

Phase 4: Decryption

The decryption phase is done by users who want to access an encrypted file. The users who need to decrypt the message are required to be authenticated by using their certificate. If their certificate is valid and not appeared in the CRLs, they will be given $EDK_{uid.k}$ and ESS for the decryption. The decryption functions used in our C-CP-ARBE is shown in Table 3.3.

Table 3.3: Decryption Functions Used in C-CP-ARBE

Decryption function	Meaning
$\text{DEC}_{\text{C-CP-ARBE}}$	A decryption function based on CP-ABE model to decrypt the cipertext *CT*.
DEC_{RSA}	A decryption function that uses a RSA private key to decrypt the $EDK_{uid.k}$ and *ESS* encrypted by the corresponding *uid*'s public key.
DEC_{AES}	A decryption function which uses AES symmetric key to decrypt the seal ciphertext *SCT*.

8. **DEC**(SCT, ESS, GSK_{uid}, $EDK_{uid\cdot k}$) $\rightarrow M$.

The decryption algorithm performs two consecutive steps as follows:

(1) Decrypt *SS* and *SCT*

$SS = \text{DEC}_{\text{RSA}}\ (GSK_{uid}, ESS)$

$CT = \text{DEC}_{\text{AES}}\ (SS, SCT)$

The algorithm takes user's global secret key GSK_{uid} and it returns the session key *SS*. Then, the algorithm takes *SS* to decrypt *SCT* and gets the *CT*.

(2) Decrypt *CT*

The user of *uid* takes $EDK_{uid,aid}$ for decrypting data in a cloud. The decryption step is conducted as follows:

$UDK_{uid,k} = \text{DEC}_{\text{RSA}}(GSK_{uid}, EDK_{uid,k})$

To decrypt *CT*, we define DEC$_{\text{C-CP-ARBE}}$ = DECNode(CT,$UDK_{uid.k}$,x) as a recursive algorithm that takes as input a ciphertext $CT = (ACP, \check{C}, C, C_y, C'_y)$, a user decryption key $UDK_{uid,k}$ which is associated with a set of attributes S_{A_k} , and a node x from ACP.

If x is a leaf node and i ($\in S_{A_k}$) is the attribute associated with x *then* DECNode(CT,$UDK_{uid.k}$,x) = DECNode(ACP, $\check{C}$,$C, C_{y_1} = g_k^{q_{y_1}(0)}, \ldots, C_{y_n} = g_k^{q_{y_n}(0)}$, $C'_{y_1} = H(att(y_1))^{q_{y_1}(0)}$, …, $C'_{y_n} = H(att(y_n))^{q_{y_n}(0)}$), $D = g^{(\alpha_k+r)/\beta_k}$, ,($D_{j_1} = g_k{}^r H(j_1)^{r_j}$,…, $D_{j_n} = g_k{}^r H(j_n)^{r_j}$) , ($D'_{j_1} = g_k{}^{r_{j_1}}$)),…, ($D'_{j_n} = g_k{}^{r_{j_n}}$)), x).

Since H is a hash that applies into G_1, we define $H(i) = g^a$ for some unknown a, and g is a generator of G_1.

$$\text{DECNode}(CT, UDK_{uid.aid}, x) = \frac{e(D_i, C_x)}{e(D'_i, C'_x)}$$

$$= \frac{e(g_k^r \cdot H(i)^{r_i}, g_k{}^{q_x(0)})}{e(g_k^{r_i}, H(i)^{q_x(0)})}$$

$$= \frac{e(g_k^r, g_k{}^{q_x(0)}) \cdot e(H(i)^{r_i}, g_k{}^{q_x(0)})}{e(g_k^{r_i}, H(i)^{q_x(0)})}$$

$$= \frac{e(g_k^r, g_k{}^{q_x(0)}) \cdot e(g_k{}^{ar_i}, g_k{}^{q_x(0)})}{e(g_k^{r_i}, g_k{}^{aq_x(0)})}$$

$$= \frac{e(g_k, g_k)^{rq_x(0)} \cdot e(g_k, g_k)^{ar_i\, q_x(0)}}{e(g_k, g_k)^{r_i a\, q_x(0)}}$$

$$= e(g_k, g_k)^{rq_x(0)}.$$

If $\notin S_{A_k}$, then DECNode(CT,$UDK_{uid.k}$,x) = $\perp$

We then consider the recursive case when x is a non leaf node. The algorithm DECNode(CT,$UDK_{uid.k}$,x_k) will proceed as follows. For all nodes z that are children of x, it calls DECNode(CT, $UDK_{uid.k}$, z) and generates the output as F_z. Let $S_x$be an arbitrary k_x-sized set of child nodes z such that $F_z. \neq \perp$. If no such set exists, the function returns $\perp$. Otherwise, F_x can be computed as follows:

$$= \prod_{z \in S_x} F_Z^{\Delta_{i,S'_x}(0)}, where\ i = index\ (x)\ and\ S'_x = \{index(z) | z \in\ S_x\}.$$

$$= \prod_{z \in S_x} (e(g_k, g_k)^{r \cdot q_x(0)})^{\Delta_{i,S'_x}(0)}$$

$$= \prod_{z \in S_x} (e(g_k, g_k)^{r \cdot q_{parent\,(z)}\,(index\,(z))})^{\Delta_{i,S'_x}(0)} \quad by\ construction$$

$$= \prod_{z \in S_x} (e(g_k, g_k)^{r q_x(i) \cdot \Delta_{i,S'_x}(0)}$$

$$=(e(g_k, g_k)^{r \cdot q_x(0)} \qquad \text{by polynomial interpolation}$$

The algorithm takes the function on the root node R_T of the *ACP*. If the tree is satisfied by a set of attributes S_{A_k} , then we set W = DECNode(*CT, UDK*$_{uid.k}$*, r*) $= e(g_k, g_k)^{r \cdot q_R(0)} = e(g_k, g_k)^{rs}$. The decryption algorithm is shown as:

$$\hat{C}/(e(C,D)/W) \ = \ M\ e(g_k, g_k)^{\alpha_k s}/(e(h^s, g_k{}^{(\alpha_k + r)/\beta_k}) \ / \ e(g_k, g_k)^{rs}) = M.$$

Phase 5: Revocation

This phase includes revoke process (done by AA) and re-encryption (done by a data owner). Our scheme supports the revocation of both user and attribute level. The details of each revocation method is described as follows:

- **User Revocation**

When a user is revoked from the system, he will be removed from the user list and his credentials including keys used for decryption (*UDK, SS*) and PKI key pairs will be also revoked. Hence he can no longer use existing keys to decrypt the files.

To revoke the users, the algorithmic procedure includes:

9. **RevokeUser**(*uid*, *SK*$_k$, *UL*) → *update UL, update GRP, and update UDKG.* The RevokeUser algorithm takes *uid* and AA's secret key (*SK*$_k$), and user list (*UL*) as inputs. The secret key of

attribute authority is used to sign the revocation request and the revoked user is removed from the corresponding $UL_{R_{uid}}$ which is contained in a UL. Once the UL is updated, the **CreateGrouproleParameter**GRP is run again. Also, a graph structure related to the user *uid* is deleted from the *UDKG*. It is assumed that when any user is revoked from the system, the public certificate $Cert_{uid}$ of the revoked user *uid* will appear in the Certificate Revocation Lists (CRLs).

As of the user revocation, there are three algorithms to be executed as follows:

(1) **UpdateSS**(*GRP′*) → *SS′*. The algorithm takes updated *GRP′* into the hash function and and it returns an updated SS (*SS′*).

(2) **ReENCCT**(*CT*, *SS′*) → *SCT′*. The algorithm takes as input ciphertext *CT*, and the updated *SS′*, then it returns an updated seal ciphertext (*SCT′*).

$$CT \mapsto \mathrm{ENC}_{\mathrm{AES}}(SS', CT) \equiv SCT'$$

(3) **ReENCSS**($Cert_{uid}$, *SS′*)→ *ESS′*. The algorithm takes user's public key ($Cert_{uid}$) of the active users who can access the ciphertext that the revoked users ever accessed and the updated SS (*SS′*), then it returns an updated version of encrypted SS (*ESS′*).

$$SS \mapsto \mathrm{ENC}_{\mathrm{RSA}}\,(Cert_{uid}, SS') \equiv ESS'$$

- **Attribute Revocation**

Attribute revocation is a finer revocation level in CP-ABE. In CP-ABE, when an attribute is revoked, the ciphertext ever encrypted with the ACP where the revoked attribute is contained needs to be re-encrypted. Here, data owner must download the affected ciphertext from the cloud storage and decrypt it. Then, the data will be re-encrypted with the updated ACP and sent back to the cloud. This introduces both computation cost at client side and communication cost for downloading and uploading the ciphertext.

Most attribute-based primitives attempt to tackle the revocation problem by either avoiding re-encryption or outsourcing re-encryption to the outsourcing server such as proxy-based re-encryption (PRE). Among the solutions employed, PRE is considered as an effective method to alleviate the re-encryption cost for data access control in cloud computing. The concept of PRE is to delegate the re-encryption function to a semi-trusted proxy and let the proxy perform this tedious and time-consuming task. Even though re-encryption is a major cost for the revocation and policy update, re-encryption key generation is also expensive overhead that data owners must account for. Furthermore, most works have tried to offload costly overhead of ciphertext re-encryption to the proxy as much as possible. This is advisable for the fixed cost the owner pay to cloud provider. However, the cloud provider may charge the service usage based on CPU time in some cases. Thus, avoiding too much overload in terms of the frequency of re-encryption operation performed by a proxy is desirable. This computation optimization aspect is mostly overlooked by the existing PRE schemes.

In this thesis, we propose an optimized PRE scheme called Very Lightweight Proxy Re-Encryption (VL-PRE) as an efficient attribute revocation mechanism where the re-encryption key generation and ciphertext re-encryption are outsourced to a semi-trusted proxy. The proxy is responsible for taking the parameters sent from the data owner to compute the re-encryption key which contains the secret key that can be used to decrypt the ciphertext. Then, the generated re-encryption key is used to re-encrypt the ciphertext. With the introduced proxy, the overheads mentioned at the client side are significantly reduced. The details of our proposed VL-PRE is presented in Chapter 4.

To revoke an attribute, the following algorithms are executed.

10. **RevokeAtt**(SK_k , $S_{uid.k}$, Att_{rev})→ $S'_{uid.k}$. The algorithm takes as input AA's secret key (SK_k), a set of all attributes issued to user *uid* and managed by authority *k* ($S_{uid.k}$), and attribute revoked (Att_{rev}), it then returns the updated $S_{uid.k}$ ($S'_{uid.k}$) for each target user *uid*. The $S_{uid.k}$ for each target user is updated as follows:

```
Set uid[] = get uid where $Att_{rev}$ is contained in $S_{uid.k}$

    foreach (uid in a uid[])
        Set $S'_{uid.k}$. = Delete $Att_{rev}$ from $S_{uid.k}$,
    end foreach
```

11. **ReGenUserKey**(SK_k , $S'_{uid.k}$, , $Cert_{uid}$)→ $EDK'_{uid,k}$, $UDKG'_{uid.k}$. The algorithm takes as inputs the AA's secret key (SK_k), updated set of attributes ($S'_{uid.k}$) of a user *uid*, and a public key

certificate of user *uid* ($Cert_{uid}$). Then, it outputs a new user decryption key $UDK'_{uid.k}$ to target users and updated $UDKG_{uid.k}$ ($UDKG'_{uid.k}$). The procedure for updating the encrypted $UDK_{uid.k}$ ($EDK'_{uid,k}$) and a $UDKG'_{uid.k}$ for each target user *uid* is conducted as follows:

```
Set uid[] = get uid where $S_{uid.k}$, is updated

    foreach (uid in a uid[])
        Set $EDK'_{uid,k}$ = UserKeyGen($S'_{uid.k}$ , $SK_k$, $Cert_{uid}$)
        Set $UDKG'_{uid,k}$ = update($UDKG_{uid,k} \rightarrow EDK_{uid,k}$ , $EDK'_{uid,k}$)

    end foreach
```

12. **UpdateACP**(*ACP*, Att_{rev}) → *ACP'* The algorithm takes as input access control policy *ACP* and attribute revoked, Att_{rev}, then it returns the updated ***ACP'*.**

    As one of the typical problem of CP-ABE, the most straightforward implementation of CP-ABE involves owner' data being re-encrypted by data owners when a policy update must be effected. In this thesis, we develop a secure policy updating algorithm and employ VL-PRE to enable the policy update to be done in the cloud in an efficient and computationally cost effective manner. The details of policy update management are discussed in Chapter 5.

13. **ReENC**(PK_k , *rk*, *CT*)→ *CT'*: This algorithm is performed by a proxy. It takes re-encryption key *rk*, an original *CT* as inputs. Actually, *rk* consists of a key used to decrypt the ciphertext and it will be used with an encryption algorithm to re-encrypt the data which outputs a re-encrypted ciphertext *CT'*. Ciphetext re-encryption is thus computed as:

We define $ENC_{rk\text{-}CT}$ as a ciphertext re-encryption function that makes use the decryption capability of the *rk* (where the data owner's secret key is resided) and then the $ENC_{\text{C-CP-ARBE}}$ method is applied to re-encrypt the ciphertext. The re-encryption function is defined as follows:

$$CT \rightarrow ENC_{rk\text{-}CT}(PK_k, rk, CT) \equiv CT'$$

The details of the protocol on how the re-encryption key is constructed and used in proxy re-encryption process are explained in Chapter 4.

To decrypt a new *CT* (*CT'*) encrypted by a proxy, users whose keys are regenerated can decrypt the *CT'* by using their new $UDK'_{uid \cdot k}$. The citphertext decryption is computed as:

$$M = \text{DEC}\ (SCT, ESS, GSK_{uid}, EDK'_{uid \cdot k})$$

3.3 Security Analysis of C-CP-ARBE

In this section, we analyze the security of C-CP-ARBE model and the security of user revocation.

3.3.1 Security Properties of C-CP-ARBE

For the security proof of our C-CP-ARBE scheme, we refer to the proof of the inner encryption layer which is based on the CP-ABE scheme. There are two possible major attacks: cryptographic key attack and collusion attack in CP-ABE setting.

Property 1: C-CP-ARBE is resistant against decryption by unauthorized users.

For the cryptographic key attack, the attacker needs to break the cryptographic construct of CP-ABE in a polynomial time. In the random oracle model, it is assumed that the attacker need to try to compute a from g^a, The discrete logarithm problem is well known to be difficult. It is not known whether there is polynomial time algorithm to solve the problem. Thus attacking the cryptographic key of CP-ABE is computationally infeasible [16].

Property 2: Our C-CP-ARBE is also collusion resistant and secure in the standard oracle model. Regarding the collusion attack, attackers may collude and collect attributes from several users in order to satisfy the access control policy. Because each attribute has its own pubic attribute key and different random number is assigned to each attribute which is not known to a user (the information on how the key is constructed is described in UserKeyGen algorithm), it is infeasible for attackers to perform collusion attack [16].

3.3.2 Security Analysis of User Revocation

Here, we first revisit the 2-layer encryption scheme to gain the understanding of the key credentials used to access the plain data. The encryption scheme is done by the following procedures:

1. Encrypt message with C-CP-ARBE encryption

   $M \mapsto \mathrm{ENC}_{\text{C-CP-ARBE}}(PK_k, ACP, M) \equiv CT$

2. Encrypt CT with the secret seal (symmetric encryption)

   $CT \mapsto \mathrm{ENC}_{\mathrm{AES}}(SS, CT) \equiv SCT$

It requires *secret seal SS* and $UDK_{uid.k}$ to decrypt the *SCT* and *CT* respectively. Because *SS* and $UDK_{uid.k}$ are encrypted with the individual's user public key $Cert_{uid}$, only the authorized users having the corresponding private key can decrypt these credentials and gain access to the plain data. Generally, when a user is revoked from the system, she cannot use her credentials to access the plaintext. In our scheme, when a user is revoked from the system, her public key is now in the CRLs. Furthermore, a corresponding *SS* is changed. Hence, the revoked users will be no longer valid in the system and they are not given the valid key (*SS*) to decrypt the sealed ciphertext. This supports backward security concept.

3.3.3 Security Analysis of Attribute Revocation

The security of attribute revocation is discussed in Chapter 4.

3.4 Write Access Enforcement

Our solution for enforcing write authorizations focuses on efficiency, availability, and flexibility, as it avoids expensive communication cost for returning the file to be re-encrypted by the data owners and the encrypted files will be loaded back to the cloud server. Especially, when there are several requests for data re-encryption, the practicality and efficiency for data owner in serving this task would be degraded. In our model, users having write privilege can update the files, re-encrypt the updated file, and save it back to the cloud storage. It is assumed that data owners encrypt their *ACPs* with a simple CP-ABE policy where data owners' ID and a set of user IDs having write permission are included.

To this end, we make use of a simple CP-ABE tree policy to encrypt the access policy. The policy is encrypted by a set of identity attributes of the data owners and authorized users. Let ACP_{PE} be the policy constructed from a set of user identity attributes. The function used to encrypt *ACPs* is described as follows:

$$\mathrm{ENC}_{\mathrm{CP\text{-}ABE}}(PK_k, ACP_{PE}, ACP) = ENC_{CP\text{-}ABE}(ACP)$$

The function returns encrypted policies $E(ACP)$.
Figure 5 shows a sample of policy encryption (PE) model.

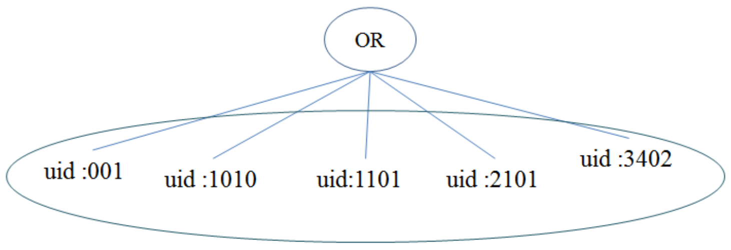


Figure 3.6: Policy Encryption (PE)

As can be seen in Figure 3.6, PE is constructed by accommodating identify attribute (*uid*) of the data owner and users who are allowed to access the policy (read only). Then, the above policy will be used to encrypt the access control policy. Hence, only data owners and authorized users listed in the PE are allowed to access the policy. Since the PE contains only role attributes and their user identity attributes, the policy encryption and decryption cost are very small. The proposed PE technique does not require additional keys for data owners or users because they can use existing decryption keys (which already contain their identity attributes, *uid*). The decryption function is defined as:

$$\mathrm{DEC}_{\mathrm{CP\text{-}ABE}}(UDK_{uid,k}, ENC_{CP\text{-}ABE}(ACP), x) = ACP$$

With our proposed policy encryption technique, it supports secure policy retrieval property that guarantees the policy access by only legitimate users.

Therefore, the permitted users with write privilege can retrieve the policy by using their current user decryption key (UDK) for re-encrypting the data. However, this method allows the authorized users (i.e. write-permitted uses) to learn the policy content. Data owner must ensure that only authorized users are allowed to access the policy. After the file is updated, it will be signed with an updater's signing key. To re-encrypt the data, write-permitted users call ENC algorithm to encrypt the updated data M' as follows:

$$\mathbf{ENC}(PK_k, SS, M', ACP, Cert_{uid}) \rightarrow SCT'$$

Then, the updated SCT' is sent to the cloud storage. SCT' can be only decrypted by data owner's and authorized users having privilege to access the file encrypted.

This proposed write access enforcement mechanism is based on secure policy retrieval where its security is described by the following Theorem.

Theorem 1: Secure policy retrieval. All access policies must be accessible only by authorized write-permitted users.

Proof: With our write enforcement mechanism, all ACPs are encrypted with the CP-ABE policy where a set of identity attributes of all authorized users (such as data owners, write-permitted users) is associated. Thus, a user having write privilege (whose key contain her identity attributes) can access the policy and use it for data encryption.

3.5 Function Analysis of C-CP-ARBE

We analyze our C-CP-ARBE in terms of the access control functions and access control efficiency as described below.

- *Expressive, flexible, scalable, and fine-grained access control*

We adopt RBAC model in the design of traditional CP-ABE policy enforcement scheme. This is to support larger number of users and better attribute management by assigning a group of attributes that belong to the specific role. In our policy tree structure, the operations AND, OR, and K out of N are supported to logically express the natural evaluation for roles and attributes as the access control rules. The policy also accommodates the privilege of user for each role distinctively. Our model supports both read and write access which reflect the practicality in a complex data sharing scenario. In our model, user attributes from multiple domains can be specified under the respective policy of any data owner. In addition, the update of the policy can be flexibly done over the access tree structure. All in all, RBAC improves CP-ABE model in the following ways.

- It provides a more expressive access policy where the attributes are grouped under the attribute "role" and privilege information (read, write) is also included.
- Existing RBAC policies implemented by the organisations can be seamlessly integrated with CP-ABE in a flexible and comprehensive manner.
- With the RBAC model, users are bound to the specific role. Hence, managing user in a role is more scalable.

As of the motivating question 1, our access control policy (e.g., a policy shown in Figure 1.1) allows the access tree to include multiple roles from multi-authority with the privilege of each role for accessing the particular file. Here, any users who belong to role (such as nurse and doctor) specified in the policy can access the shared data (e.g., OPD file, financial record of treatment).

As the integration of RBAC into CP-ABE, our C-CP-ARBE addresses the drawback of CP-ABE regarding the policy expressiveness. The proposed model renders the policy to be more expressive with the well-designed of role-attribute based structure and the inclusion of privilege (read or write) specification.

- *Key Complexity Reduction*

  Our C-CP-ARBE cryptographic model provides no key distribution cost and dynamically manages multiple user decryption keys to be stored in a structured key decryption graph (UDKG) structure. User keys will be dynamically invoked upon the user's request for access. This provides cost for key distribution which is the cumbersome for data responsible authority and enables efficient multiple keys assignment and retrieval.

Hence, a user does not need to hold multiple keys even she has several access rights to access multiple resources authorized by several parties. As of the motivating question 2, user decryption keys of users are maintained securely in the cloud. Our scheme will invoke the key to user dynamically and user does not need to hold many keys to access several files. For example, Dr.John belongs to Hospital B but he is allowed to access out patient data (OPD) file owned by Hospital A. The system will invoke the key upon his access to OPD file. We also demonstrate the performance of user key update when there is a revocation of attribute between our scheme and CP-ABE.

Using our C-CP-ARBE, a fundamental problem for managing multiple keys in CP-ABE is solved. Our key management scheme outperforms the existing CP-ABE in terms of no key distribution cost.

- *Efficient User Revocation*

  We propose two-layer encryption to support strong encryption and enable the optimization for key management and user revocation cost. Based on our proposed scheme, a ciphertext produced from the data encryption layer is encrypted by the secret seal (SS) computed from shared role parameter. Since the SS is a kind of symmetric key, its generation process is very fast. For the user revocation, if there is any user revocation request, only the SS needs to be updated and it will be used to re-encrypt the ciphertext. Since symmetric encryption delivers fast encryption, the cost for re-encrypting the ciphertext is trivial. To this end, a revoked user cannot use their existing secret seals (SS) to decrypt the cipertext as their keys and certificates are no longer valid the PKI system. This satisfies backward security. As of motivating question 3, if there is any user is revoked, the system will just update SS parameter and other non-revoked users in the hospital do not get any effects of keys change. For attribute revocation, all operations are transparent to users and they need to do nothing as all the processes are done in the cloud. The approach for addressing attribute revocation is presented in Chapter 4.

Our C-CP-ARBE provides less revocation cost and yields zero impact to non-revoked users while existing CP-ABE based schemes require re-key generation, file re-encryption, or ciphertext update.

- *Secure and Optimized Write Access*

  We propose write access enforcement mechanism to enable write-permitted users securely update data with the optimized cost of data re-encryption. In our proposed model, the access tree (access policies) are encrypted and shared among the users having write permissions. Therefore, only authorized users can retrieve the access policies for re-encrypting the data before they are loaded back to the cloud. This reduces the overheads in sending the updated file back to the data owner for data re-encryption.

In this thesis, we also propose the policy update management scheme that allows the data owner to efficiently manage their policies in a secure and cost-effective manner. With our proposed policy encryption and policy update method, the motivating question 4 is therefore satisfied. In addition, with our proposed write access enforcement mechanism which is a part of C-CP-ARBE, the problem on the dependability of data owner in serving the data re-encryption as well as the costs in doing so are eliminated.

3.6 Implementation and Evaluation

For the evaluation of our proposed scheme, we develop the prototype system called CLOUD-CAT to facilitate a flexible, secure, and efficient management of multiple user accesses and multiple access control policies in multi-owner cloud computing environment. Then, we conduct the performance evaluation and perform comparative analysis of the revocation cost between our scheme and the original CP-ABE.

3.6.1 Prototype System Development

Regarding the tool supporting access control in cloud environment, existing cloud applications services such as Google Drive, Dropbox, OneDrive, iCloud can serve functional data outsourcing in general. However, rigorous requirements for strong authentication, fine-grained access control with encryption feature, and flexibly authorization for collaborative data sharing are not comprehensively provided by these cloud services. Amazon's EC2 [36] is one of the most popular IaaS services. However, the advanced access models such as RBAC or Attribute-based Access Control (ABAC) are not supported by EC2. It simply restricts the access control by using operating system (OS) image and access control list.

We implemented CLOUD-CAT using Java and PHP. CLOUD-CAT is a web-based tool to help users and data owners can use the tool flexibly and conveniently. The tool is run on the Apache Sever. For the key management server, we use Open SSL as a core PKI service to generate key pairs to users and system entities. The service is run on Intel Xeon, E562 processor 2.40 GHz. Memory 4 GB., with Ubuntu Linux.

As a proof of concept, we developed access control policies for a hospital information system. 50 test polices were set up to validate functional policy modeling and enforcement of the tool. Over 25 roles from both internal organization and other partner organizations such as partner hospitals and insurance firm are included in the policies.

In overall, the tool can dynamically and correctly enforce access control policies to users as well as support administrative functions for data owners/administrators in an efficient manner (comprehensive, correct, and easy-to-use). Figure 3.7– 3.9 show major features of the prototype system that are designed for users and data owners/administrators.

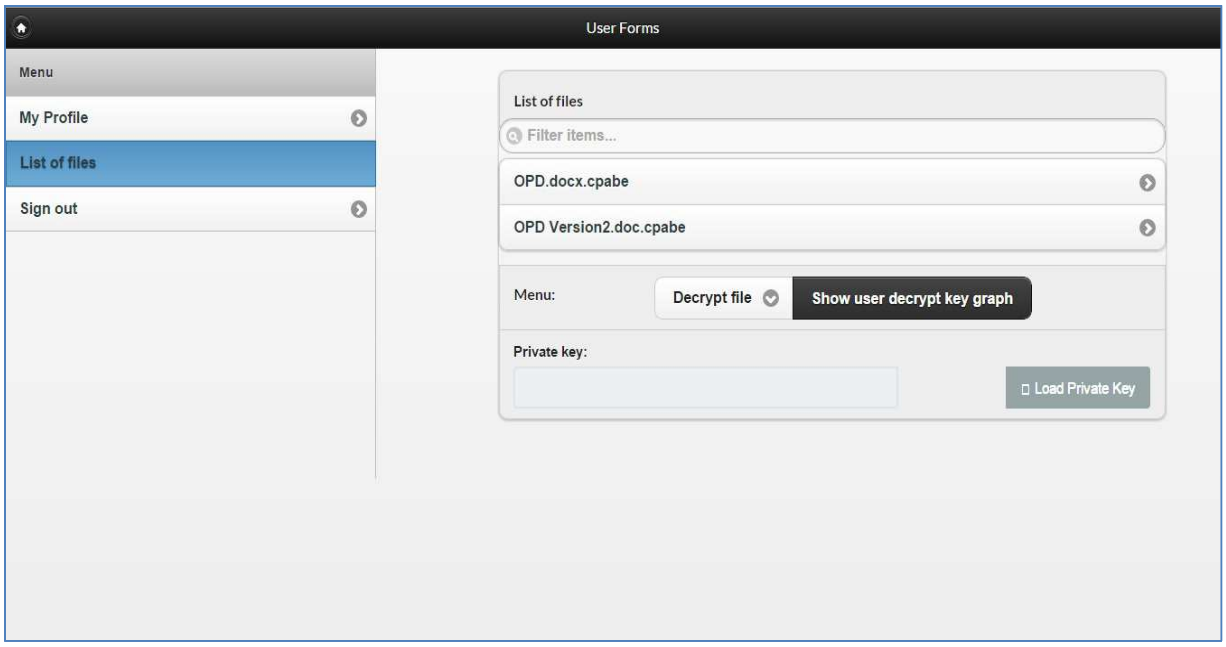


Figure 3.7: User Access Control Management

- **Data Access Management for collaborative users**

Figure 3.7 shows the screen shot of our web-based access control tool for collaborative users. From this screen, authorized users can access a list of data files shared by multiple owners with respect to their roles and privileges specified in the respective policy. To access the encrypted files via the tool, users need to be authenticated with the validity of public certificate (verify with certificate revocation lists CRLs). If the user authentication step is successful, users can login to the system and can download the encrypted files they authorized to access and use their private key to decrypt the file. The system will also check that the user can only download the files their decryption key is qualified based on the UDKG profile. Since our system supports multi-owner data sharing, a user may have several decryption keys, the tool allows the user to view the decryption key graph to realize which key is matched to the target file.

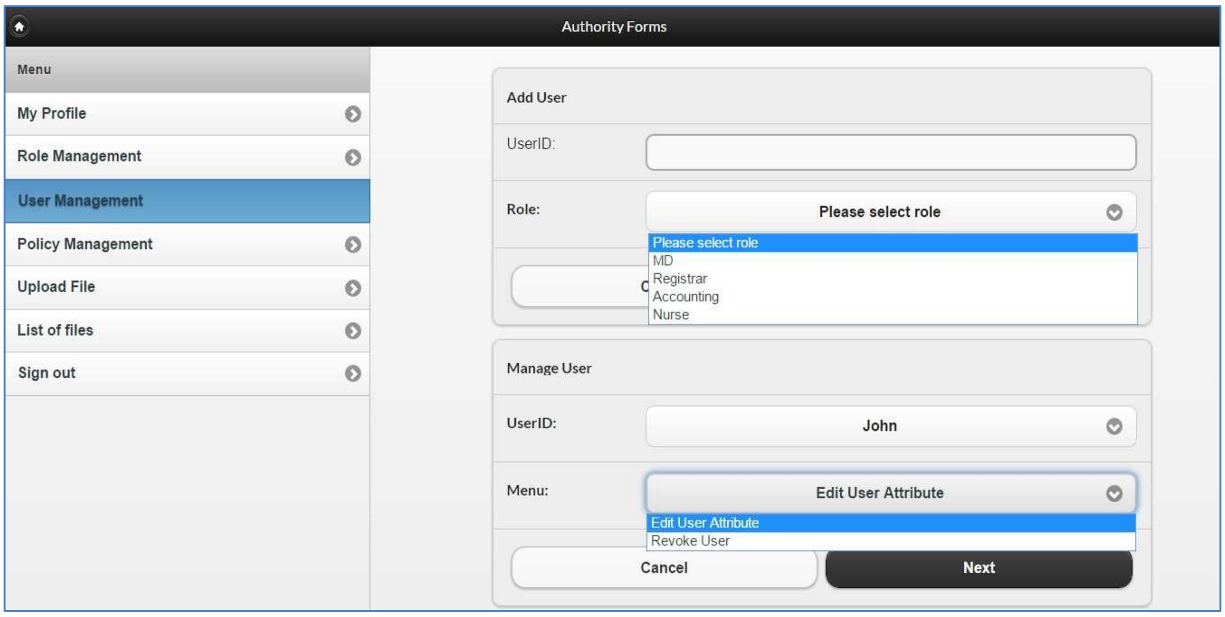


Figure 3.8: Administrative Functions: User Management

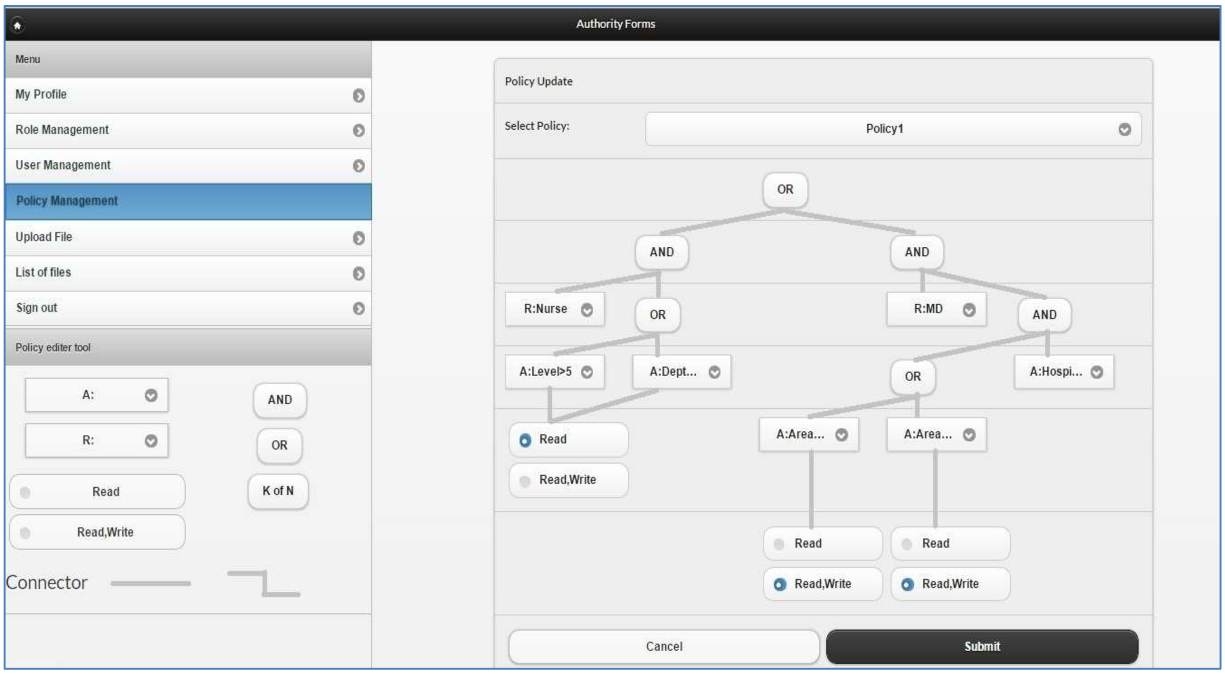


Figure 3.9: Policy update screen

- **Administrative Access Control and Policy Management for Data Owners/Administrators**

Figure 3.8 displays the screen shot for data owners and administrators. Here, administrative features are available to support user-role and role-attribute management (add, update, revoke), data upload, and policy management. For the policy updating, data owners or administrators can manipulate (add, update, delete) policy rule and/or content. Figure 3.9 displays the policy management screen where data owners or administrator can flexibly update the policy rule and/or content via the graphical interface. Administrator can efficiently update policy elements including roles, attributes, privilege, and logical operators without coding effort or naïve policy writing.

3.6.2 Performance Evaluation

For the performance measurements, we aim to evaluate the effectiveness of our cryptographic algorithms and proposed key management model to substantiate the reduction of key management cost. We conduct the experiment to evaluate the computation cost for encryption and decryption which are the major operations used in the access control. The C-CP-ARBE cryptographic service is run is run on Intel Xeon, E5620 processor 2.40 GHz. Memory 4 GB., with Ubuntu Linux. The performance evaluations for encryption and decryption are run on the same server.

- **Encryption and Decryption Performance**

We evaluate the computation time of encryption and decryption of our scheme based on the number of users and number of attributes in policy and user keys. Table 3.4 and 3.5 reveal the encryption and decryption time of our C-CP-ARBE.

Table 3.4: Encryption Time (in seconds)

	Size of data				
No. of Leaf Nodes	1MB	2MB	4MB	8MB	16MB
20	1.08	1.25	1.64	2.43	4.1
40	2.14	2.32	2.73	3.58	5.23
60	3.2	3.48	3.84	4.68	6.27
80	4.34	4.53	4.92	5.91	7.35
100	5.46	5.72	6	6.86	8.45

Table 3.5: Decryption Time (in seconds)

	Size of data				
No. of Attributes containing in the key	1.04MB	2.06MB	4.1MB	8.2MB	16.4MB
5	0.56	0.68	1.07	1.8	2.7
10	0.8	0.92	1.3	2.01	2.95
15	0.96	1.14	1.53	2.2	3.09
20	1.15	1.34	1.71	2.42	3.29
25	1.38	1.54	1.98	2.66	3.43

Figure 3.10 and Figure 3.11 show the corresponding encryption time and decryption time in graphical format that our C-CP-ARBE spends in the simulated cloud server respectively.

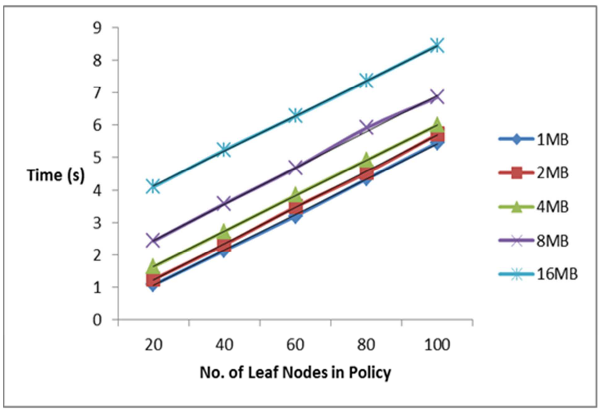


Figure 3.10: Encryption Time

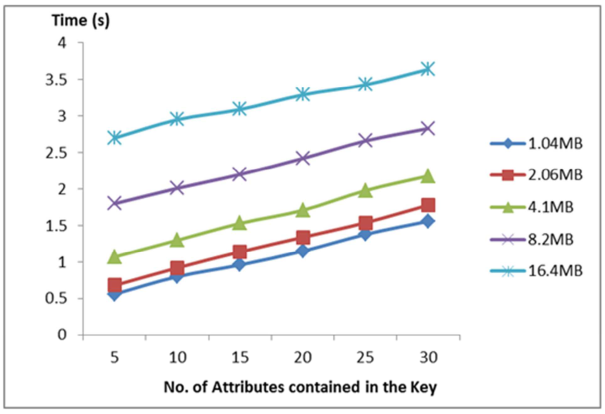


Figure 3.11: Decryption Time

For evaluating the encryption time, we use the different file sizes to be evaluated with the different number of leaf nodes containing in the ACP. As of Figure 3.9, our encryption algorithm provides competitive and acceptable time for large files. This offers practical performance because our model uses symmetric encryption and organizes the group of attributes to be under the specific role for a CP-ABE tree. Increasing the number of leaf nodes in a policy also increases the encryption time in a small constant growth. However, we notice the increase in the number of leaf nodes provides more impact to the performance.

In our experiment, the decryption time is measured from the time used to decrypt files with the different number of attributes contained in the user decryption key. Even though the ciphertext sizes and numbers of attributes in the key are large, our decryption algorithm yields efficient and comparable time for the tested files especially for the ciphertexts that are smaller than 4MB. This indicates the efficiency of our decryption algorithm and the proposed key user decryption key graph. The entire process of the encryption and the decryption is done through the multi-thread programming.

- **Performance of User Revocation**

To demonstrate the efficiency of our user revocation mechanism, we measure the performance of user revocation processing time taken by our approach and the CP-ABE scheme. We use the policy that contains 5 roles and 20 attribute-leaf nodes to encrypt a 1 MB file. We compare the revocation cost of our C-CP-ARBE and CP-ABE encrypted with the diffident no. of users accessing files. Table 3.6 displays the performance of user revocation time between the original CP-ABE and our C-CP-ARBE. Then, Figure 3.12 shows corresponding graphs of the processing time of entire revocation cost used by two approaches.

Table 3.6: Comparison of User revocation Time (in seconds)

No. of Users	C-CP-ARBE	CP-ABE
10	3.8	14.55
50	8.25	25.21
100	15.55	48.87
200	27.23	90.05
400	43.24	148.92
800	72.66	265.59

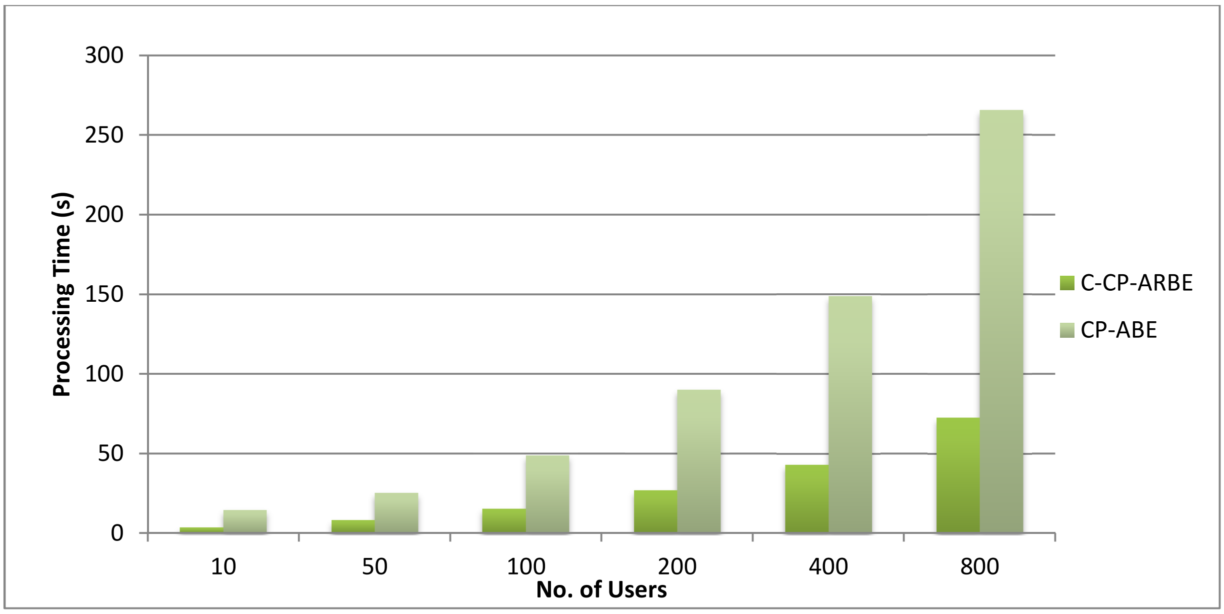


Figure 3.12: Comparison of User Revocation Time

As seen from Figure 3.12, our scheme delivers less processing time of user revocation (in (regenerating key and re-encrypting ciphertext) than the original CP-ABE scheme because our C-CP-ARBE is not linear to the number of revoked users. The scheme only needs to regenerate public role parameter to construct a new *SS* and uses it to re-encrypt the ciphertext. Hence, there is no cost for key re-generation. In contrast, CP-ABE needs to deal with file re-encryption which is the most expensive overhead for the revocation that data owner or the authority has to handle. As a consequence, we can infer that our user revocation scheme is more efficient and scalable than the original CP-ABE.

Chapter 4: Attribute Revocation Management

The chapter begins with the preliminary section describing the significance of the problem and limitations of existing attribute revocation solutions. This chapter devotes to our newly proposed revocation management scheme called Very Lightweight Proxy Re-encryption (VL-PRE) technique based on the extension of our previous works [35, 46, 81] deployed in CP-ABE based access control. The formal VL-PRE scheme is illustrated. Hereafter, the computation complexity and security analysis of VL-PRE is given. Finally, experiments and evaluation are provided.

4.1 Problem Statements

Revocation management is one of the major drawbacks of CP-ABE. With this scheme, users may hold a secret key consisting of attributes shared by other users. When an attribute is revoked, there are two subsequent processes of the attribute revocation. First, non-revoked users who share the attributes revoked are required to update their decryption keys. Second, files encrypted by policies containing attributes revoked must be re-encrypted with an updated policy. In the latter process, if the attribute revoked does not appear in the policy, file re-encryption is not needed. Key update and file re-encryption are always cumbersome, resulting in degrading the efficiency of attribute-based access control implementation.

Considered the example in Chapter 3, if the hospital issues a new attribute "professional level" to replace existing attribute "level", the attribute "level" thus needs to be revoked. If the attribute revoked appears in the ACP, the ACP needs to be updated.

Figure 4.1 presents the example of attribute revocation.

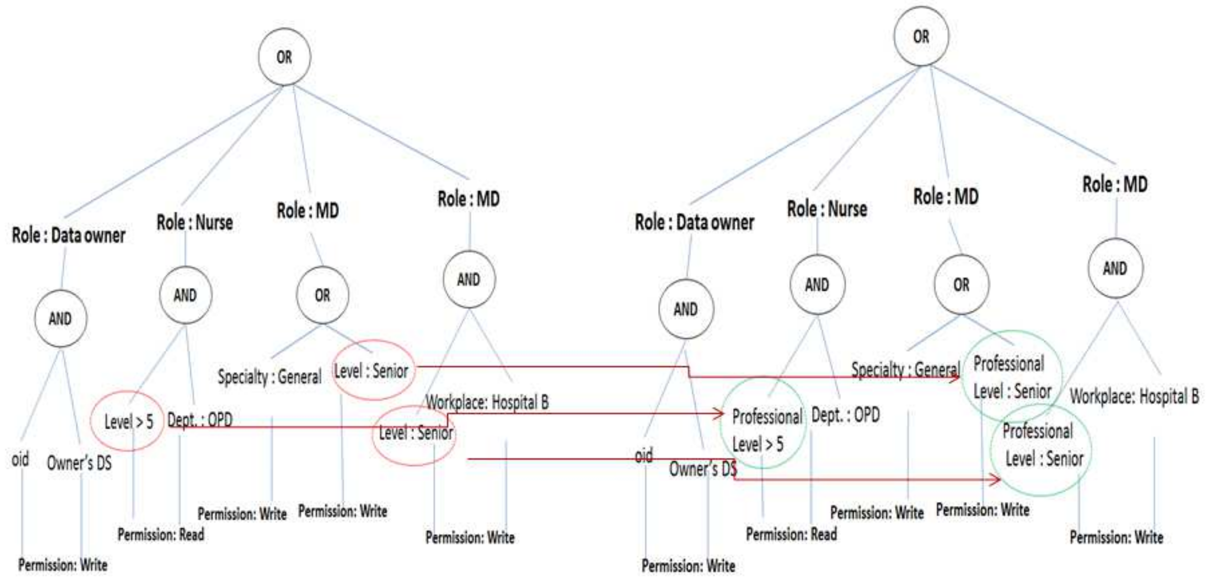


Figure 4.1: Attribute Revocation Causing a Policy Update

As of the example cases, users whose roles are doctor or nurse need to update their keys. In addition, files encrypted with the policy containing a revoked attribute “level” need to be re-encrypted. These costs for keys update and file re-encryption do not only affect the existing non-revoked users, data owner is still required to take action for re-encryption every time when the revocation occurs. These problems become more serious when there are a large number of users in the system. To conclude, the revocation problem in the CP-ABE is not so simple that it exposes sub-sequence costs (communication and computation cost) to existing users, and data owners.

Therefore, designing very lightweight components and processing of PRE, as well as secure PRE protocol will enable practical operation and more accessibility for implementing access control in mobile cloud environment. Especially, mobile cloud computing (MCC) applications have been emerged to supported flexible mobile data access. Therefore, enabling all dynamic decision making processes in the MCC environment is needed to be achieved with lightweight components [42].

In our previous work, we proposed L-PRE scheme [35] to resolve two expensive aspects of ciphertext policy - attribute based encryption: the reencryption of files that were encrypted with revoked attributes, and the update of keys containing revoked attributes. L-PRE fully outsources re-encryption task to a delegated proxy in the cloud. The pseudorandom is used to generate numbers for encrypting the re-encryption key components.

However, if there are frequent revocation cases, the re-encryption key needs to be re-generated every time. To do so, data owners have to prepare and submit data package to the proxy for computing the re-encryption key.

In this thesis, we entail a more practical PRE scheme by introducing a very lightweight PRE (VL-PRE) as an enhancement of the L-PRE to support a more flexible and scalable attribute revocation and policy update. Compared to existing lightweight PRE schemes [27, 29-31, 35], VL-PRE is proposed to improve the computation and communication performance with a more lightweight PRE protocol. Existing lightweight PRE schemes focus only the optimization of the PRE cost at a client side, while VL-PRE possesses two major aspects enabling the computation and communication cost at both data owner and cloud side are substantially reduced.

First, the size of the root decryption key (*RDK*) is significantly reduced. RDK is used to decrypt the existing ciphertext and it is a part of components sent to a proxy for computing re-encryption key. The *RDK* used in VL-PRE contains only two attributes: owner id and digital signature, while *RDK* used in L-PRE is consisted of all qualified attributes belong to the data owner.

Second, VL-PRE retains the re-encryption key generated each time in its period of validity. Accordingly, our new method relies on re-encryption key update instead of repeating re-encryption key generation every time when there is attribute revocation or policy update. Data owners only submit an updated access policy (ACP) to the proxy. Then, the proxy only updates the ACP of the existing re-encryption key before re-encrypting the ciphertext.

With a very light packet for re-encryption key generation, it requires little memory and low computation for data pre-processing. Hence, data owners are able to support secure attribute revocation in a more flexible and efficient and cost-optimized manner.

To evaluate the efficiency of the proposed VL-PRE, we will measure the initialization (set-up) cost and the re-encryption cost that must be substantially smaller than the conventional method.

The simulation for testing with mobile devices in dealing with the revocation will be performed to confirm the efficiency of our proposed scheme.

4.2 Our Proposed VL-PRE

4.2.1 System Model

Figure 4.2 presents our proposed access control system with proxy re-encryption for data outsourcing scenario.

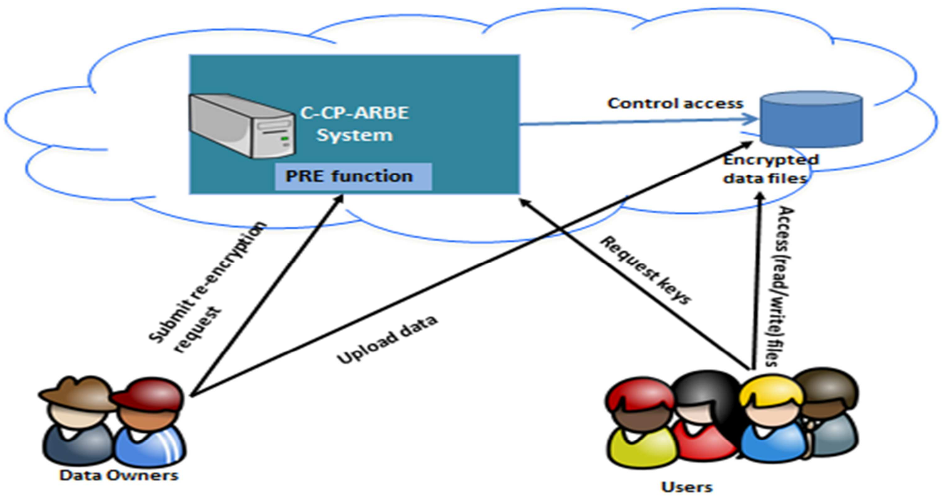


Figure 4.2: A System model of Access Control Model with PRE for Outsourced Data

In the data outsourcing environment, data owners initially upload encrypted data files to a cloud storage server. For the access control, we deploy C-CP-ARBE system accommodating access control functions to support authentication and authorization in multi-user and multi-owner cloud setting. In the original PRE, a proxy is given a re-encryption key (*rk*) for re-encrypting the ciphertext.

A re-encryption key normally contains functions and key generation components such decryption key and access control policy used for ciphertext decryption and data re-encryption respectively.

In our approach, we introduce a PRE function to a semi-trusted proxy server as an attribute revocation mechanism of the C-CP-ARBE system. A proxy server is located in the cloud environment where it can be considered as honest but curious. That is, the proxy exactly performs the functions assigned with no deviations or severe attacks, but it may try to learn as much as possible about the input or output from the view of PRE protocol it runs. To provide the trust of the proxy for performing file re-encryption, X.509 certificate and PKI key pair are issued to the proxy and users for the authentication purpose. In addition, it cannot access to the plaintext as the decryption key is not given. A proxy is generally responsible for two tasks assigned. First, it is required to compute or update the re-encryption key initiated by data owners when there is attribute revocation and the revoked attribute appear in the access policy. Second, it has to re-encrypt the ciphertext by using the generated re-encryption key. In our model, there is a repository to store the re-encryption keys (*rk*) where the freshest version of the key is used by the proxy for re-encryption process.

The proxy has no privilege in accessing the content of file as it has no knowledge of keys necessary for decryption. In a PRE in cloud environment, data owners have full privilege in holding the root key for decrypting any files and managing the policies used for encryption. Users typically request for accessing resources via C-CP-ARBE system. Any related-PRE processes are also transparent to users.

4.2.2 VL-PRE Process

We design the execution of VL-PRE into three phases: Re-encryption key Generation, Re-encryption key Update, and Re-encryption key Renewal. Basically, the proxy transforms ciphertext CT to CT' with a re-encryption key rk generated by a proxy server. The three phases of VL-PRE operation are designed to support the practical life cycle of access control implementation where the revocation and policy update can be changed any time. Phase 2 and phase 3 are added to reduce the computation costs at data owner side in preparing the re-encryption key generation package to the proxy.
In addition, we keep re-encryption key in the proxy and make it updated upon the policy change decrease the workload of proxy. To secure the re-encryption key components such as RDK and ACP, we employ a cryptographically secure random number generator (CSPRNG) [77] for encrypting these key components.
Table 4.1 depicts the notations used in VL-PRE

Table 4.1 Notations Used in VL-PRE

Notations	Meaning
rk	A re-encryption key used to transform the ciphertext CT to *CT'*.
$RDK_{oid.k}$	A root decryption key of data owner *oid* issued by attribute authority *k*. RDK contains identity attribute *oid* and digital signature of *oid*.
$Cert_{oid}$	A public key certificate containing data owner's public key issued by a certification authority CA.
DS_{oid}	Digital signature of data owner *oid*.
GSK_{proxy}	A proxy's private key
$Cert_{proxy}$	A proxy's public key
IP_{Proxy}	IP address of the proxy dedicated to perform re-encryption task.
R	Random number used to encrypt $RDK_{oid.k}$ and the ACP which are the components of re-encryption key *rk*.

VL-PRE process consists of three major phases: Generate Re-encryption key, Update Re-encryption key, and Renew Re-encryption key. Figure 4.3 illustrates the overall process of our newly proposed VL-PRE.

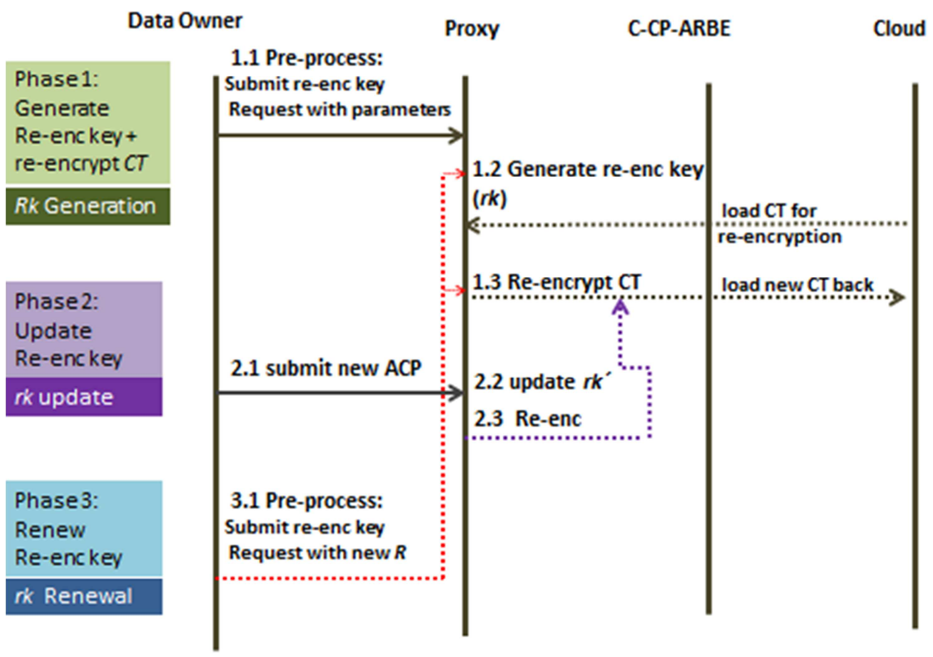

Data Owner
Proxy
C-CP-ARBE
Cloud
Phase 1: Generate Re-enc key + re-encrypt CT
Rk Generation
Phase 2: Update Re-enc key
rk update
Phase 3: Renew Re-enc key
rk Renewal
1.1 Pre-process: Submit re-enc key Request with parameters
1.2 Generate re-enc key (rk)
load CT for re-encryption
1.3 Re-encrypt CT
load new CT back
2.1 submit new ACP
2.2 update rk´
2.3 Re-enc
3.1 Pre-process: Submit re-enc key Request with new R

Figure 4.3: VL-PRE process

Phase 0: The proxy is issued a private key and certificate (GSK_{Proxy}, $Cert_{Proxy}$)

Phase 1: Re-encryption key Generation and Ciphertext Re-encryption:

In Phase 1, the initial re-encryption key, system and all related-PRE parameters are initially generated. Some of them such as re-encryption key generated, random number can be used in a specified period of time. This phase consists of Pre-process, ReKeyGen and ReEnc algorithms which are described as follows.

1. **Pre-process:** This process is a setup phase where the basic parameters and Root decryption key are constructed. These components are used to produce re-encryption key (*rk*). First, AA issues root decryption key ($RDK_{oid.k}$) to the data owner *oid* by running the algorithm **RDKGen** as follows:

(1) **RDKGen**(SK_k, *oid*, DS_{oid})$\rightarrow RDK_{oid.k}$.

RDKGen takes input attribute authority's secret key SK_k, identity attribute of data owner *oid*, and data owner's digital signature (DS_{oid}), then it produces the root decryption key $RDK_{oid.k}$ for further use in re-encryption key generation process. In the design of our extended access tree, all ACPs developed by the data owner *oid* are added with a pair of (*oid*, DS_{oid}). Thus, $RDK_{oid.k}$ can be used to decrypt all files which belong to their owner *oid*.

Data owner initially runs the following algorithms.

(2) **GenPREConfig**(IP_{Proxy}, $Cert_{oid}$)$\rightarrow$PREConF

The algorithm takes the input as the network address of proxy server, $\mathrm{IP_{Proxy}}$ and owner's public key certificate, $Cert_{oid}$ which is used to authenticate with the proxy before executing the ReKeyGen algorithm. It then outputs the PRE configuration file (PREConF). In general, the PREConF is generated once and can be used until there is a change of its input parameter. The file will be included in the *param* which is a part of Re-KeyGen algorithm.

(3) **Gen*R***($\{r_1, r_2, \ldots r_n\}$)$\rightarrow R$

The algorithm randomly chooses a set of random seeds *rs*, as input and generates random number *R*. The size of *R* is 256 bits.

(4) **Enc*R***(R, $RDK_{oid.k}$, ACP)$\rightarrow$ $\mathrm{ENC_{AES}}(R, RDK_{oid.k})$, $\mathrm{ENC_{AES}}(R, ACP'(Y))$

We use $\mathrm{ENC_{AES}}$ with *R* as the encryption key to encrypt the root decryption key $RDK_{oid.k}$ and *Y* a set of leaf nodes. Then, $\mathrm{ENC_{AES}}(R, RDK_{oid.k})$ and $\mathrm{ENC_{AES}}(R, ACP'(Y))$ are computed. Since *Y* consists of a set of attributes whose data type come in two forms: non-numerical and numerical, non−numerical attributes are specified as any string of letters(attr), and numerical attributes are specified as "attr = N" where N is a non-negative integer, it can also contain comparison operators('<', '>', '<=', '>=', and '='). We take $\mathrm{ENC_{AES}}$ to encrypt individual attribute x ($x \in Y$). For the implementation in Java, the encryption procedure is conducted as follows:

```
foreach ( $Y$ as $x$ )
        $\mathrm{ENC_{AES}}$($x$.getBytes());

end foreach
```

Then, a data owner takes the proxy's public key to encrypt the *SS* and *R* to produce *ERSS.*

$$R, SS \mapsto \mathrm{ENC_{RSA}}\ (Cert_{proxy}, R, SS) \equiv ERSS$$

Then, data owner submits PREConF, *ERSS*, $\mathrm{ENC_{AES}}(R, RDK_{oid.k})$ and $\mathrm{ENC_{AES}}(R, ACP'(Y))$, as parts of re-encryption key to the cloud proxy.

2. ReKeyGen (PREConF, *ERSS*, $\mathrm{ENC_{AES}}(R, RDK_{oid.k})$, $\mathrm{ENC_{AES}}(R, ACP'(Y))$$\rightarrow rk$

This algorithm is run by a proxy. It verifies PREConF, and takes input *encrypted secret seal SS* (*ESS*), $\mathrm{ENC_{AES}}(R, RDK_{oid.k})$ and $\mathrm{ENC_{AES}}(R, ACP'(Y)\,)$*).* The proxy then uses two elements: $\text{ENC}_{\text{AES}}(R, RDK_{oid.k})$ and $\text{ENC}_{\text{AES}}(R, ACP'(Y))$ to formulate a re-encryption key *rk*. In our model, we associate the Expire_time to specify the validity of *rk*. Then, it outputs a re-encryption key *rk* that can be used to transform a ciphertext *CT* to another ciphertext *CT′*. Therefore, *rk* is structured as follows:

$rk = (\text{ENC}_{\text{AES}}(R, RDK_{oid.k}), \text{ENC}_{\text{AES}}(R, ACP'(Y)), Expire\_time)$

3. **ReENC**$(PK_k, rk, CT) \rightarrow CT'$: This algorithm is performed by a proxy. It takes input a AA's public key (PK_k), re-encryption key *rk*, an original *CT*. It outputs a re-encrypted ciphertext *CT′*.

   There are four steps for re-encryption including:

   (1) Decrypt *ERSS*

   (2) Decrypt CT with $RDK_{oid.k}$

   (3) Encrypt *CT* with a new ACP'

   (4) Encrypt *CT′* with *SS*

Details of each step are described as follows:

3.1 Decrypt *ERSS*

To re-encrypt the *CT*, *R* is obtained from decrypting ERSS by using the following function.

$R, SS = \text{DEC}_{\text{RSA}}(GSK_{Proxy}, ERSS)$

Then, the proxy runs the functions called CMR(CallMatchRemove) which is a local function used to trigger the decryption functions: $\text{ENC}_{\text{AES}}(R, RDK_{oid.k})$and $\text{ENC}_{\text{AES}}(R, ACP'(Y))$.

The decryption step is conducted as follows:

We use DEC_{AES} as a decryption function to decrypt $\text{ENC}_{\text{AES}}(R, RDK_{oid.k})$ and $\text{ENC}_{\text{AES}}(R, ACP'(Y))$. The CMR procedure is shown below.

```
Set $RDK_{oid.k} = \text{DEC}_{\text{AES}}(R, \text{ENC}_{\text{AES}}(R, RDK_{oid.k}))$
Set $ACP' = \text{DEC}_{\text{AES}}(R, \text{ENC}_{\text{AES}}(R, ACP'(Y))$
```

For each encrypted leaf node (Y'), it is decrypted as the following procedure.

```
foreach ($Y'$ as $x$ )
    $\text{DEC}_{\text{AES}}(x)$
end foreach
```

When leaf node (*Y*) is decrypted, the ACP is obtained.

3.2 Decrypt *CT* with $RDK_{oid.k}$

In this step, the ciphetext decryption is conducted as follows:

(1) Decrypt *SCT*

$CT = \mathrm{DEC}_{\mathrm{AES}}\ (SS, SCT)$

(2) Decrypt *CT*

$RDK_{oid.k}$ is automatically used to decrypt the current ciphertext as follows:

$$\mathrm{DEC}_{\mathrm{C\text{-}CP\text{-}ARBE}}\ (CT, RDK_{oid.k}, x) = M$$

where $x \in Y$.

3.3 Encrypt *M* with a new ACP'

In this step, the algorithm instantly applies *rk* to re-encrypt the data.

We define $\mathrm{ENC}_{rk\text{-}CT}$ as a ciphertext re-encryption function that makes use the decryption capability from the *rk*. $\mathrm{ENC}_{\mathrm{C\text{-}CP\text{-}ARBE}}$ is applied to re-encrypt the ciphertext.

$$\mathrm{ENC}_{rk\text{-}CT}\ (PK_k\ , rk, M) = \mathrm{ENC}_{\mathrm{C\text{-}CP\text{-}ARBE}}(PK_k, CMR(rk), M)$$

In the re-encryption step, new policy is computed as *CMR(rk)* and is used in the re-encryption. The re-encryption function is defined as follows:

$$CT \rightarrow \mathrm{ENC}_{rk\text{-}CT}(PK_k\ , rk, CT) \equiv CT'$$

3.4 Encrypt *CT'* with *SS*

Finally, the proxy takes *SS* to encrypt a new Ciphertext (*CT'*).

$$CT' \mapsto \mathrm{ENC}_{\mathrm{AES}}(SS, CT') \equiv SCT$$

Then, *SCT* is sent back to be stored in the cloud server.

Since *rk* is independent to the original message *M*, the re-encryption does not harm any privacy of the message.

Phase 2: Re-encryption key Update:

In this phase, it is assumed that the existing *rk* is still valid but there is a policy update. Thus, the existing *rk* needs to be updated. First, data owner applies a current random number *R* to encrypt the updated ACP and a new $\mathrm{ENC}_{\mathrm{AES}}\ (R, ACP'(Y))$ is produced.

Then, a new ENC$_{\text{AES}}$(R, $ACP'(Y)$) is used to update the rk with respect to the following procedure.

Set rk' =update(rk→ (ENC$_{\text{AES}}$(R, $RDK_{oid.k}$), ENC$_{\text{AES}}$ (R, $ACP'(Y)$, $Expire\_time$))

The new rk' is used to re-encrypt the existing ciphertext.

Technically, the algorithms help to reduce both computation and communication overhead at both data owner side and proxy since the $RDK_{oid.k}$ needs not to be encrypted every time and the information (only the updated ACP) sent out to the proxy is small, while the proxy does not need to fully compute a new re-encryption key upon every revocation cases, it only updates the key instead. With the key update strategy, it provides less computation cost for re-encryption key generation compared to the existing PRE schemes.

Phase 3: Re-encryption key Renewal

In this phase, if the current re-encryption key rk expires, the **Gen*R*, Enc*R*, ReKeyGen** and **ReEnc** algorithms in phase 1 will be run. Then, re-encryption key generation and ciphertext re-encryption are performed by the proxy. In the renewal phase, *PREConF* file and *CMR* function are not required to be re-generated.

Thus, the renewal phase usually consumes less computation time than the initialization phase. In addition, re-encryption key renewal is not required to perform instantly when the key expires, it will be executed when there is the next policy update or attribute revocation.

4.3 Security Analysis of Attribute Revocation

This subsection analyzes the security of attribute revocation mechanism in C-CP-ARBE. In our scheme, we identify two major requirements: backward security and confidentiality of re-encryption key for guaranteeing the security of our attribute revocation mechanism as follows:

(1) Backward security: In our scheme, an attribute revocation deals with two possible cases.

Revoking attribute of individual user. In this case, the user (whose attributes are revoked) cannot decrypt ciphertexts that require the revoked attributes for decryption. In our scheme, when an attribute is revoked, a user whose decryption key ($UDK_{uid,k}$) contains the revoked attributes needs to be issued with an updated key if he/she is still an active user. Technically, the $UDK_{uid,k}$

which contains the revoked attributes is automatically invalid in the system and a new $UDK_{uid,k}$ will be generated and encrypted with the user's public key before it is sent to update the UDKG in the cloud. The system will also check that the user can only download the files their decryption key is qualified based on the UDKG profile. Decryption key re-generation can prevent the user whose attributes are revoked from decrypting any existing or new ciphertexts which are encrypted by the policies contained revoked attribute. With this case, the re-encryption process is not required.

Revoking attribute from the access control policy. In this case, users whose decryption key contains the revoked attribute cannot use their keys to decrypt existing and new ciphertext (re)encrypted with an updated policy as their keys are no longer valid in the system.
Their keys need to get updated (or re-generated) with a valid set of attributes before it is considered as a valid decryption key. In addition to the re-generated decryption key that can prevent the use of invalid key (key containing revoked attribute), an updated ACP is encrypted with the AES key before it is used to re-encrypt the ciphertexts. Therefore, the users or attackers cannot compromise (add, modify, delete) the policy content in order to make use the decryption key that contains revoked attribute or may be generated from a corrupted AA.

(2) Confidentiality of re-encryption key (*rk*)

We assume that ACP which is used to encrypt the message is maintained by the data owner and not disclosed to any parties. Because the re-encryption key generation components generated and sent from the data owner to a proxy through the network, it is possible that the attack such as network eavesdropping or network sniffing may occur to steal or modify the re-encryption key components. In our model, the communication between a data owner and a proxy is secure with the SSL. Therefore, the communication is secure based on SSL encryption. To protect the *rk* components, a random number is generated to encrypt the *rk* with AES encryption. With the brute force attack for the decryption key, we define the following theorem and prove that VL-PRE is secure against this attack.

Theorem 2: An attacker cannot recover a plaintext by using the given re-encryption key components.

Proof: Since the proxy is a semi-trusted server located in the cloud, it is assumed that it performs the functions as it is delegated and it will not harm any security to the systems such as collusion attack with the attackers. For the security attack in this case, if the attacker can get the re-encryption key components by anyways, he needs to try decrypting both *SS* encrypted by AES algorithm, and $RDK_{oid.k}$ encrypted with AES encryption where its key is constructed from *R,* a cryptographic pseudo random number generator (CPRNG) which is proven for achieving the recovering security and preserving security [82].

□

4.4 Experiments and Evaluation

We evaluate the efficiency of our VL-PRE by focusing on the revocation case that causes the re-encryption of data outsourced in the cloud. We perform a comparative analysis of our proposed VL-PRE and the recent optimized PRE schemes proposed in [27] and [30]. We also conduct the experimental scenarios to measure the performance and throughput between these three schemes.

4.4.1 Comparative Analysis of Existing PRE Schemes

In this section, we discuss the comparative analysis of the attribute-based access control revocation schemes including Tysowki and Hasan scheme [27], Liang et al. scheme [30], and VL-PRE. Here, [27] and [30] are chosen to compare with our scheme since they share a similar goal in optimizing the PRE cost but using different methods.

From Table 4.2, we can see that both [27] and [30] still need a client to have a computation task for computing the re-encryption key generation or key update in the PRE process. VL-PRE fully offloads re-encryption key generation to the proxy. Thus, it eliminates the computation cost at client side. Besides, the size of re-encryption key of both [27] and [30] is subject to the number of attributes contained in the ciphertext they use to compute or update the re-encryption key, while VL-PRE contains only fixed two attributes accommodated in the RDK. Therefore, VL-PRE delivers less decryption cost compared to those two schemes.

Table 4.2: Comparison of PRE Schemes

Characteristics	Tysowki and Hasan Scheme [27]	Liang et al. scheme [30]	VL-PRE
Offloading Re-encryption key generation to cloud	Partially support	Partially support	Fully support
Re-encryption key Size	No. of attributes contained in two ciphertexts	No. of identity attributes used to reencrypt ciphertext	Fixed 2 identity attributes
Cryptographic Operations for constructing re-encryption key	Attribute-based encryption	Identity-based encryption	Symmetric encryption
Computation workload in updating re-encryption key	Fully update	Fully update	Partially update

We can see that both [27] and [30] still need a client to have a computation task for computing the re-encryption key generation or key update in the PRE process. To complete re-encryption key generation used for ciphertext re-encryption, [27] suffers from high computation cost that needs two computation steps from both client in computing secret group key (GSK) and proxy in computing re-encryption key $RK_{0\rightarrow x}$. Also, the GSK will be delivered to all active users which introduce communication cost. In [30], the PRE scheme is based IBE where a set of identity attributes are used to construct the re-encryption key for re-encryption the ciphertext. In our scheme, the data owner needs to prepare re-encryption key (*rk*) components and send them to the proxy for computing *rk.* Some parameter such as RDK is pre-computed. Thus, the computation cost at client side in fully computing the *rk* is delegated to a proxy. Besides, the size of re-encryption key of both [27] and [30] is subject to the number of attributes or identity attributes contained in the ciphertext they use to compute or update the re-encryption key, while ours contains only fixed two attributes accommodated in the RDK. Therefore, our scheme delivers less communication and decryption cost compared to those two schemes.

In our scheme, the computation cost of re-encryption key is based on symmetric encryption which provides less computation cost than the ABE and IBE cryptography which are a kind of pairing-based operation. Furthermore, both [27] and [30] require workload for computing the key whenever the user or attribution is revoked or the policy is updated. In contrast, our scheme uses key update strategy instead of key re-generation. The re-encryption key will be re-generated periodically based on the key changeover policy specified by the data owners. Therefore, it provides less computation cost compared to those schemes using naïve PRE key generation.

4.4.2 Performance Evaluation

We conduct the implementation by setting up the simulation to proof the efficiency of our VL-PRE by measuring the processing time of individual cost of three VL-PRE phases and measure the entire proxy re-encryption process that occurs at client and the delegated proxy. Also, we perform the throughput test to demonstrate the scalability of our proposed model. Hence, our evaluation criteria are the less processing time and more scalability of VL-PRE compared to existing lightweight PRE schemes.

- *Performance of individual VL- PRE process*

We first measure the processing time of three phases of VL-PRE including re-encryption key generation, re-encryption key update, and re-encryption key renewal. In the experiment, we use ACP containing 20 attributes and random number generator library [17], which are parts of a re-encryption key. Table 4.3 presents the processing time used to generate, update, and renew re-encryption key in VL-PRE. This processing time excludes the ciphertext re-encryption process.

Table 4.3: VL-PRE Algorithm Performance

VL-PRE Operations	Processing Time (ms.)
Generate Re-encryption key	95
Update Re-encryption key	52
Renew Re-encryption key	75

From Table 4.3, we can see that key update algorithm takes least time in updating the re-encryption key and it significantly reduces the computation and communication cost if there is an update of policy that requires the re-generation of re-encryption key. The re-encryption key renewal enjoys the less computation cost compared to the initial re-encryption key generation. This is because it avoids computing some system parameters as the re-encryption key generation does. Therefore, the latter two algorithms are proven to optimize the PRE cost when there is an update of policy or frequent attribute revocations.

- *Performance of entire PRE cost occurring at a client side and a proxy side.*

In this section, we perform the comparison of performance evaluation of our scheme and [27] and [30].We developed a custom program and implemented the algorithms of both schemes by using Java pairing-based cryptography (jPBC) [78] to measure the performance of re-encryption task. We measure the processing time for the client setup and re-encryption process of all schemes. In our simulation, we also measure the PRE setup cost by using mobile phone. We use Android 6.0 SDK and employ secureRandom Java class in [77] to build a custom simulation program of cryptographic secure random number generator in Android OS. Samsung Galaxy Note II smartphone with a quad-core 1.6 GHz ARM with 2 GB RAM is used to test the performance of PRE setup cost initiated by the client (data owner).

In our simulation, the ciphertext size is 100 KB and the policy size contains 20 attributes. The group secret key (GSK) used in [27] and a user secret key (sk_{id}) used in [30] contain 15 attributes. The following graph shows the average processing time (ms.) of entire PRE cost obtained from the implementation in both client and proxy. From Figure 4.4, all schemes require client (data owner or manager) to deal with re-encryption key generation for the setup cost. In VL-PRE, the time used at both the client and the proxy is less than [27] and [30]. As for the setup cost, VL-PRE requires the client to prepare the key construction parameters and then fully offload re-encryption key generation task to the proxy. In our simulation, all parameters are initialized by the data owners and the time used for computing the parameters is counted. The proxy is generally responsible for constructing the re-encryption key and using the re-encryption key to re-encrypt the ciphertext.

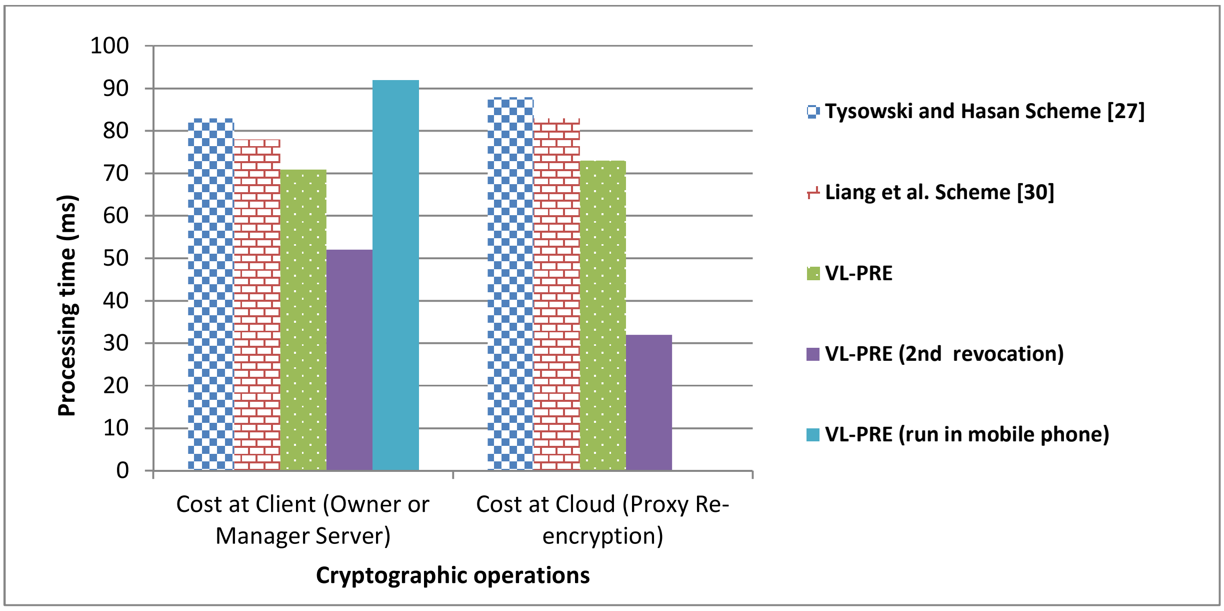


Figure 4.4: The Comparative Performance Result Showing the Processing Time of Entire PRE Process

From Figure 4.4, in [27] and [30], the re-encryption key is partially computed at client before it is sent to the proxy for ciphertext re-encryption. The scheme proposed in [27] takes more processing time than [30] as it requires two computation tasks in computing group secret key and re-encryption key. To demonstrate the efficiency of VL-PRE in supporting the revocation through client mobile device, a mobile phone is used to run the setup algorithm of PRE. The time used by mobile phone is a bit greater than other schemes run in the Mac book. As around 92 ms for the set up time run by the mobile phone, it demonstrates that our VL-PRE is proven to be feasible and efficient in supporting mobile revocation management through less-constraint mobile device.

Regarding the ciphertext re-encryption cost performed by the proxy, our VL-PRE uses a smaller key size. Hence, it provides better performance for ciphertext re-encryption than [27] and [30]. With a single ciphertext to be re-encryption, [30] takes a bit less PRE processing time than [27] as the decryption based on the identity-based encryption using fewer identity attributes is faster than attribute-based encryption.

For the 2nd round of attribute revocation, it is assumed that the same PRE cost will occur for both [27] and [30], while ours offers substantial improvement of PRE cost as there is no re-encryption key generation cost in this round. Here, the valid re-encryption key can be used to re-encrypt the ciphertext.

- *Performance of PRE cost with varying number of re-encrypted ciphertexts*

We also compare the PRE performance of three schemes to evaluate the performance impact when there are multiple ciphertexts to be re-encrypted. For the simulation, the average size of all ciphertexts used in the test is 20 KB. Figure 4.6 shows the results of PRE processing time (ms.) when there is an increased number of ciphertexts to be re-encrypted.

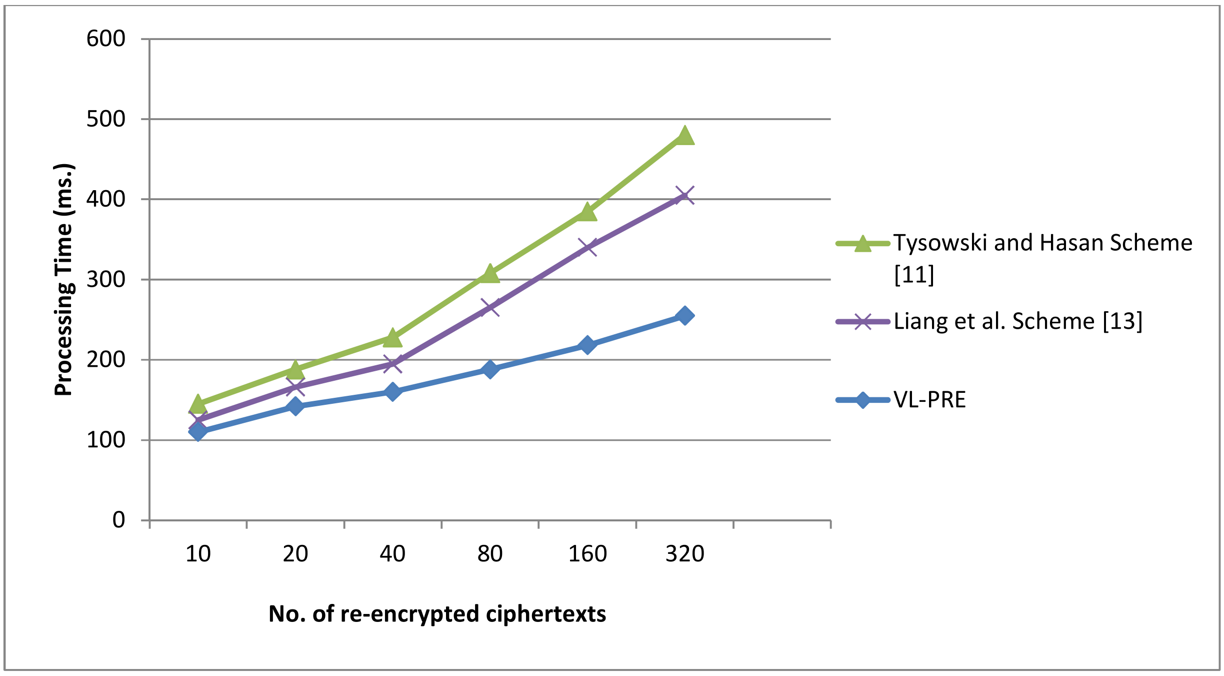


Figure 4.5: PRE Performance for Multiple Ciphertexts

As seen from Figure 4.5, both [27] and [30] suffer from the high computation cost of re-encryption key when there is an increased number of ciphertexts to be re-encrypted. This is because the computation of re-encryption keys is subject to the number of ciphertexts to be re-encrypted. In [27], the computation of re-encryption key is equal to the number of chiphertexts to be re-encryped, while [30] requires all re-encryption keys are computed once in each revocation. Hence, [27] delivers more rounds of computation than [30]. In contrast, our VL-PRE applies only one re-encryption key for all affected ciphertexts since the RDK contains two major attributes (owner id and owner's digital signature) that are sufficient to decrypt all ciphertext stored in the cloud. Therefore, VL-PRE outperforms those methods as it takes least processing time when it processes a high number of re-encrypted ciphertexts.

- *Measuring Server Throughput*

In addition to the evaluation of re-encryption time of the experimental schemes, we also evaluate the scalability of the proxy in terms of the throughput of the proxy server in serving multiple re-encryption requests in a simulated period of time. The policy used for the test contains 20 attributes and it is used for encrypting 1 MB file. Figure 4.6 presents the comparison of the PRE throughput of the two schemes.

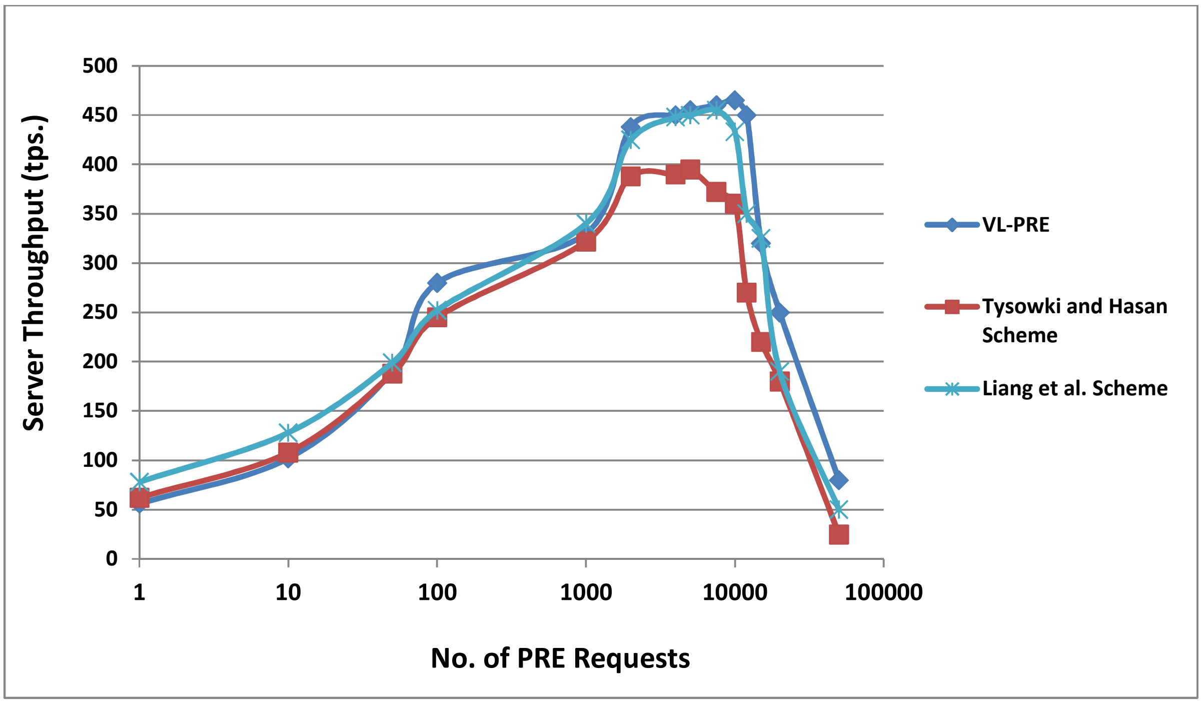


Figure 4.6: Server Throughput for Simultaneous Re-Encryption Requests

The graph shows that VL-PRE yields higher throughput in transaction per second (tps) of PRE operation than [11] and [13]. According to the result, the maximum throughput of VL-PRE was about 465 tps with 10,000 request threads, while [11] and [13] can best support approximately 5,000 and 7,500 request threads respectively. VL-PRE can tolerate around upto 12,000 requests before it declines. The result confirms that our new scheme provides a more scalability in processing concurrent workloads at the cloud side compared to recent PRE schemes.

Chapter 5: Policy Update Management

This chapter presents our proposed secure and flexible policy update in data outsourcing environment. Our proposed policy update solution which is a part of this chapter is appeared in [61]. The chapter begins with the introduction of policy update problem in the CP-ABE based access control. Then, section 5.2 presents our proposed policy update method. Hereafter, section 5.3 discusses the evaluation of our policy update scheme based on the pre-defined policy update requirements. Section 5.4 gives the details of our evaluation and implementation. Finally, the summary of our proposed solution is provided in section 5.5.

5.1 Problem Statements

In the CP-ABE based access control environment, handling evolution of access control elements such as change of users or access policy is paramount of importance. The most straightforward implementation of CP-ABE involves clients' data being re-encrypted by clients when a policy update must be affected. The problem is that the policy is tightly coupled with the encryption itself. Specifically, if there are any changes and they cause the update of policy, the data owner must deal with the operational costs including downloading the encrypted files encrypted with the before-updated policy and re-encrypting them with a new policy. Then, the updated ciphertext files are uploaded back to the cloud. These communication and computation overheads are the burdens at data owner occurring throughout the lifecycle of access control.

Unfortunately, policy updates in CP-ABE scheme has got less attention by existing research works. Most research works assume that if there is any change of policy, the straightforward update can be applied. However, the overheads caused by the policy update will be more substantial high and will degrade the quality of service if there are frequent changes. Figure 5.1 presents the access control policy specifying the access rules for accessing analyzed healthcare data.

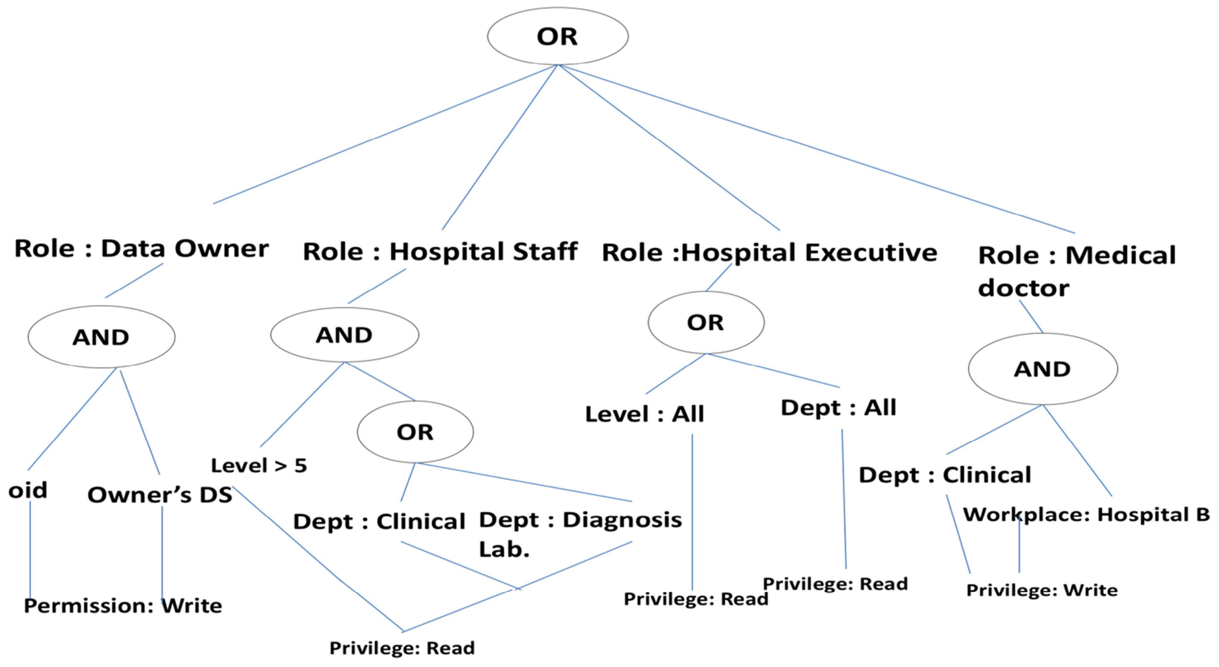


Figure 5.1: Access Control Policy of Disease Diagnosis File

As seen from the above figure, the policy allows data owner, hospital staffs, hospital executives, and a specific group of medical doctor from another hospital to access the disease diagnostic data file. Therefore, users who belong to these roles and satisfy the conditions in the policy can user their key to decrypt the file.

Typically, the policy is administered by the host hospital and it is able to be updated by authorized administrator or data owner only. Generally, such a policy can be changed anytime. For example, the senior nurse may be allowed to access the diagnosis file for preparing the summarized report. In this case, the data owner (host hospital or a responsible department) has to add role “nurse” and its attributes with the logical rules specifying the authorized access to the diagnosis file. In addition to updating the policy, the related disease diagnosis files encrypted by the before-updated policy are required to be re-encrypted with a new policy. Then, it will be uploaded back to the cloud. Thus, the effect of policy update is non-trivial and it needs an efficient management to guarantee the flexibility and security of policy update process with the optimization of subsequent communication and computation costs.

For the operational point of view, the issues including correctness, security, and accountability of the subsequent update of policy are the requirements to be provided by CP-ABE policy updating scheme. These requirements are described as follows.

- Correctness: An updated policy must be syntactically correct and the policy updating must support any types of CP-ABE policy boolean. In addition, users who hold the keys containing a set of attributes satisfying the policy are able to decrypt the data encrypted by an updated policy.
- Security: A policy must be updated by the data owner or authorised administrator only in the secure manner and a new policy should not introduce problems for the existing access control.
- Accountability: All policy update events must be traceable for auditing.

In our research, we aim to enable the policy updating to be done in the cloud in an efficient and computationally effective manner, in order to solve the problems, including the limits of CP-ABE efficiency and the computation, communication, and maintenance cost at data owner side in the process of policy update. To this end, we propose the policy updating algorithm and employ VL-PRE technique to be efficiently support policy update in C-CP-ARBE system. Our method guarantees the requirements above and VL-PRE is applied to optimize the cost at both the data owner and the proxy.

5.2 Policy update method

To complete the policy updating process, two major tasks: updating policy structure and executing PRE process are required. To this end, we propose the policy updating algorithm and employ our proposed VL-PRE to efficiently support the required tasks respectively. Figure 5.2 presents the overall process of policy update.

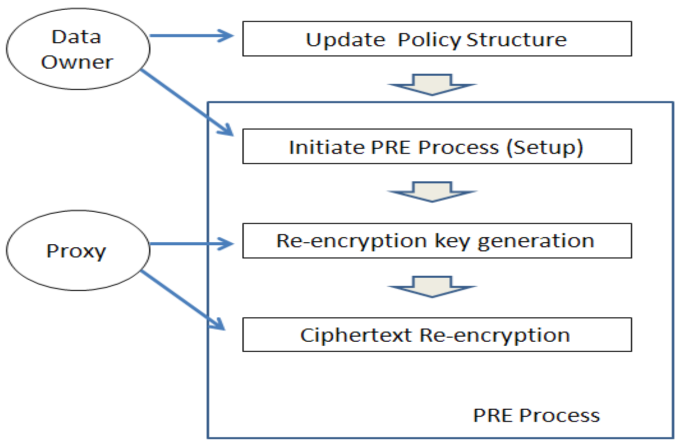


Figure 5.2: Policy Update Process

The first task of the process is to update the policy structure with respect to the policy updating algorithm and policy syntax validation algorithm. The data owner updates the access policy elements by adding/deleting attributes, logical operators AND, OR, K out of N, privilege information with respect to a new policy specification. Then, the PRE process is initiated by the data owner in the way that the data owner prepares system parameters for re-encryption key generation. Then, the parameters and re-encryption key generation components are sent to the proxy. The proxy then computes re-encryption key and performs ciphertext re-encryption with the policy updated. Finally, the updated ciphertext is uploaded to the cloud server.

5.2.1 Update Policy Structure

To update an ACP structure, we define the operators used to update threshold access trees, where the non-leaf nodes are AND, OR, and *KofN* gates, and the leaf nodes correspond to attributes.

There are 6 operations for updating the policy structure: *AddAttr2OR, AddAttr2AND, DelAttr_from_OR, DelAttr_from_AND, AddAtt2KofN,DelAtt_from_KofN.*

1) *Add an attribute to an OR gate (AddAttr2OR):* This operation updates an existing attribute $x_j (j \in [1, n]$) to an OR gate $(x_j \vee x_{n+1})$ by adding a new attribute x_{n+1}. With this operation, a new attribute x_{n+1} plays the same role as existing attribute x_j in the new policy. The procedure is shown as follows:

```
IF Add an attribute($x_{n+1}$) to an OR gate
        newpolicy$ACP'$(associated to OR gate)=$x_{1\ to\ n} \vee x_{n+1}$
END IF
```

2) *Add an attribute to an AND gate.* This operation updates an existing attribute $x_j(j \in [1, n])$ to an AND gate $(x_j \wedge x_{n+1})$ by adding a new attribute x_{n+1}. With this operation, the combination of a new attribute $x_{n+1}$and the attribute x_j in the new access control policy plays the same role as the attribute x_j in the previous policy. The procedure is shown as follows:

```
IF Add an attribute($x_{n+1}$) to an AND gate
        newpolicy$ACP'$(associated to AND gate)=$x_{1\ to\ n} \wedge x_{n+1}$
END IF
```

3) *Delete an attribute from OR gate (DelAttr_from_OR):* To delete an attribute x_j from an OR gate, a dummyAttr, a null-value attribute generated by the system, is introduced in place of the attribute deleted. In this case, the data owner needs to decide to remove the OR gate associated to dummyAttr,if the OR gate is associated with only single attribute member and a *dummyAttr*. The data owner needs to decide where the left attribute is relocated. After the deletion, if the number of attributes associated to OR gate is equal or greater than 2, the *dummyAttr* should be deleted. The procedure is shown as follows:

```
IF Delete an attribute($x_j$) from an OR gate
        IF ($n$>=3 in ORgate of $x_{1\ to\ n}$), $x_j$ = null($where\ j \in [1, n]$)→newpolicy$ACP'$(associated to OR gate) = null ∨ $x_{1\ to\ n-1}$= $x_{1\ to\ n-1}$
        ELSE $x_j$ = $dummyAttr$ → newpolicy$ACP'$(associated to OR gate)=$dummyAttr \vee x_{1\ to\ n-1}$
        END IF
END IF
```

4) *Delete an attribute from AND gate (DelAttr_from_AND):* To delete an attribute x_j from an AND gate, *dummyAttr* is introduced in place of the attribute deleted. In this case, the data owner can decide to remove the AND gate associated to *dummyAttr*, if the AND gate is associated with only single attribute member and a *dummyAttr*. The data owner needs to decide where the left attribute is relocated. After the deletion, if the number of attributes associated to AND gate is equal or greater than 2, the *dummyAttr* should be deleted. The procedure is shown as follows:

```
IF Delete an attribute($x_j$) from an AND gate
        IF ($n$>=3 in ANDgate of $x_{1 to n}$), $x_j$ = null($where\ j \in [1,n]$)→newpolicy$ACP'$(associated to AND gate) = null $\wedge\ x_{1\ to\ n-1}$=$x_{1\ to\ n-1}$
        ELSE $x_j$ = $dummyAttr$ → newpolicy$ACP'$(associated to AND gate) =$dummyAttr \wedge x_{1\ to\ n-1}$
        END IF
END IF
```

5) *Add an attribute to a KofN gate*(*AddAtt2KofN*).This operation updates an existing attribute $x_j (j \in [1,n])$ by adding a new attribute $x_{n+1}$to a *KofN*gate. With this operation, a *KofN*value is updated to *K*of(*N*+1) while a new attribute x_{n+1} plays the same role as existing attribute x_j in the new policy.

```
IF Add an attribute($x_{n+1}$) to a KofN gate
        newpolicy$ACP'$(associated to KofN gate)=$x_{1\ to\ n+1}$
        Update KofN= Kof(N+1)
END IF
```

6) *Delete an attribute from a KofN gate*(*DelAtt_from_KofN*)*:* To delete an attribute x_j from an *KofN*gate, *dummyAttr* is introduced in place of the attribute deleted. Then, the data owner needs to adjust *KofN* value as shown in the procedure below.

```
IF Delete an attribute $x_j$ from a KofN gate
        IF N-1 >= K in KofN gate
            $x_j$ = null($where\ j \in [1,n]$)→newpolicy$ACP'$(associated to KofN gate) = $x_{1\ to\ n-1}$
            Update KofN =Kof(N-1)
        ELSE
            $x_j$ = dummyAttr → newpolicy$ACP'$(associated to KofN gate)=$x_{1\ to\ n}$ (when $dummyAttr \in [1,n]$)
            KofNvalue = KofN
        END IF
END IF
```

In addition to the procedures used to update the structure of ACP, there is a procedure to check syntax of the policy update. For the syntax checking, the algorithm checks the possible operands taken on the attribute type and attribute value. This guarantees that the updated policy is syntactically correct. In our scheme, after the policy updating is complete, the proxy will automatically re-encrypt all files with the updated policy.

```
// veridatePolicySyntax

Import java.lang.*;

Input (String)ACP

if (ACP.contain(comparison between an attribute name and a negative integer OR non-integer) )
Output false

else if (ACP.contain(attribute name is threshold gate) ) Output false

else if (ACP.contain(comparison between an attribute name and an attribute name) ) Output false

else if (ACP.contain(attribute name start with integer) ) Output false

else if ( ACP.contain(threshold gate operator as 'K of (P1,P2,… PN) ' AND ( K is negative integer
OR K is non-integer OR N > K ) ) Output false

else  Output true
```

Figure 5.3: Policy Syntax Validation

5.2.2 Execute Proxy Re-encryption Function

After the policy structure is updated and validated, the PRE function is then executed to complete the policy update process. In this process, we employ three-phases VL-PRE described in previous chapter for supporting file re-encryption task when there is a policy update. The overall process of VL-PRE consists of three phases: Generate re-encryption key, Update re-encryption key, and Renew re-encryption key. Basically, the VL-PRE process is triggered when there is a case of attribute revocation or policy update and the delegated proxy will perform the re-encryption task. Basically, the proxy transforms ciphertext with a re-encryption key *rk* which is generated by a proxy server.

Since the VL-PRE is based on the key updates strategy instead of key re-generation, it outperforms existing PRE schemes in supporting policy evolution with an optimization of computation and communication cost in CP-ABE setting.

5.3 An Analysis of Our Proposed Policy Update Scheme

We analyse and evaluate our policy update scheme based on the correctness, accountability, and security requirement.

Correctness: *An updated policy must be syntactically correct and users who hold the keys containing a set of attributes satisfying the policy are able to decrypt the data encrypted by an updated policy.*

Proof: The syntax of the updating is validated through the access tree structure. Hence, attribute updated to AND, OR, *K*outof*N* is done at the policy structure. The policy checking for the update is controlled by our policy updating algorithm. The algorithm verifies the syntax of the threshold gates to ensure the correctness of the access tree. Also, if the policy is updated with valid attributes (issued trusted AA) the users who hold sufficient attributes satisfying a new policy are able to decrypt the file encrypted by a new policy.

Security: *A policy must be updated by the data owner or authorized administrator only in the secure manner and a new policy should not introduce problems for the existing access control.*

Proof: To enable the policies to be securely managed in the re-encryption process, data owners encrypt all ACPs with AES algorithm before they are sent to the proxy for re-encryption process. Hence, only data owners or authorized administrators are allowed to access the policy and update the policy structure. It is assumed that the policy updating mechanism requires the presence of digital signature of the data owner or the administrator after the policy is updated. With the restrict policy update capability, any new access rules are thus reflected from the data owners.

Accountability: *All policy updating events must be traceable.*

Proof: When the policy is updated, event log keeps the details of update including login users, update time, and update operations. In addition, the system requires digital signing of the data owner or the administrator to commit the update.

5.4 Experiments and Evaluation

In this section, we present our evaluation and experiments to demonstrate the efficiency of our proposed policy updating algorithm and VL-PRE. We choose Kan Yang et al. [60] scheme based on the ciphertext update method and J. Lai et al. [28] scheme based on the recent PRE method to be compared with our proposed VL-PRE. For the evaluation, Yang scheme [14] and Lai scheme [20] are chosen to compare with our approach because they represent the ciphertext update method and PRE method that share similar goal to our work in addressing the policy update in CP-ABE based access control. We first analyze the cost of policy update of two related works. Then, the comparative performance evaluation is performed to measure the policy update cost of our scheme and the mentioned schemes.

5.4.1 Evaluation of Policy Update Cost

We compare features and computation cost of policy update of the three schemes based on four major aspects including the party performing key generation, the availability of policy outsourcing, policy update method, and the computation cost (computational complexity). Let t_c be a total number of attributes contained in the ciphertext.

Table 5.1: Comparison of Policy Update Features and Computation Cost

Operation	K. Yang et al. [60]	J. Lai et al. [28]	Our C-CP-ARBE
Update Key Generation	At owner side	At owner/authority side	At Cloud Server
Policy Encryption	No	No	Yes
Policy Update Method	Ciphtertext update	PRE	VL-PRE
Computation Cost	$O(t_c)$	$(\#S_{aid}(U))$ + $\text{ENC}_{\text{CP-ABE}}$	$\text{ENC}_{\text{CP-ABE}}$

From Table 5.1, In Yang scheme, data owner has to update key generation and to update the ciphertext to complete the policy updating process. For the ciphertext update, the data owner needs to compute ciphertext components for new attributes. The entire computation cost is subject to the number of attributes $O(t_c)$ in the ciphertext and the type of update operations (i.e. OR, AND) over the access structure. In Lai scheme, PRE concept is used to convert the existing ciphertext according to the updated policy. In each policy update, this scheme requires the computation cost of trapdoor generation of which all attributes ($\#S_{aid}(U)$) in the systems are used, and data re-encryption cost of CP-ABE encryption ($ENC_{CP\text{-}ABE}$). However, the trapdoor needs to be generated at the authority or data owner side. This limits the operation with the availability of the authority or data owner. In contrast, in our model, we delegate the major cost of re-encryption key generation and file re-encryption to a proxy in the cloud.

The cost of policy update of our scheme is not dependent on the update operations and number of attributes contained in the policy. The computation cost is totally computed by the proxy for ciphertext re-encryption based on CP-ABE encryption ($\mathrm{ENC}_{\mathrm{CP\text{-}ABE}}$). In addition, all updated policies must be done by the data owners and they are encrypted with AES before they are sent to the proxy for executing the PRE process. This guarantees security and privacy of access control policy used to enforce the access control in multi-authority cloud systems.

5.4.2 Performance Evaluation

In our experiments, we implement the application service using PHP and Java language which are run on the Apache Sever. The service is run on Intel Xeon, E562 processor 2.40 GHz., Memory 4 GB. with Ubuntu Linux. We use the Pairing-Based Cryptography library version 0.5.12 to simulate the cryptographic constructs of those two compared schemes. Our core cryptographic library is extended and developed from the CP-ABE programming library provided in [18]. For the client's (data owner) environment, we use MacBook Pro Intel Core i5 Dual-core, 2.7GHz, memory LPDDR3 1866MHz, 8 GB.

In the experiment setting, we simulate KeyUpdate and CiphertextUpdate algorithms for Yang scheme, while Trapdoor generation and policy update based on PRE are simulated for Lai scheme. For our C-CP-ARBE scheme, policy updating algorithm and VL-PRE are used to measure the cost of policy update.

To demonstrate the performance improvement, we compare total time used for policy update and re-encryption between these three approaches. We simulate the policy update protocols of Yang scheme by simulating the key update generation and ciphertext update while Lai scheme and our C-CP-ARBE use the PRE strategy. To measure the performance, we vary the number of attributes updated (added) in the given access policy. The access policy contains up to 160 attributes with the mix of an "OR" and an "AND" gate. The policy is used to encrypt 2-MB file. Then, we measure the total time for the policy update and file re-encryption or ciphertext update used by these three schemes.

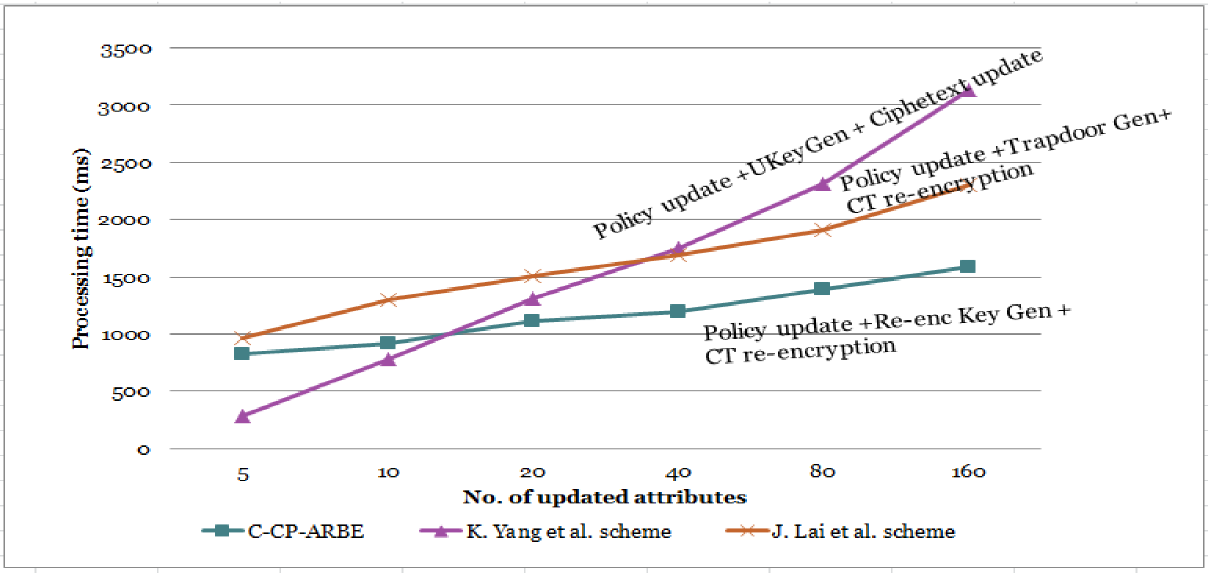


Figure 5.4: Comparison of policy update cost

As of the Figure 5.4, compared with Yang scheme, and Lai scheme, our C-CP-ARBE fully outsources the PRE process to the proxy. Thus, the computation at data owner side is significantly reduced. With our scheme, the data owner only updates a policy at her own machine, while the policy and the subsequent costs (re-encryption key generation and ciphertext re-encryption) are fully outsourced to the delegated proxy. With a small re-encryption key size and key update strategy of VL-PRE, the processing workload performed by the proxy at cloud is also substantially reduced. In Lai scheme, even though the authors exploit PRE to transform the ciphertext in the cloud server, the data owner still has to compute the trapdoor and update the policy before the proxy does the re-encryption process. Noticeably, the performance of our scheme and Lai scheme are not subject to the number of attributes changed in the policy or the operations used in the policy. In contrast, with the ciphertext update strategy of Yang scheme, it is very practical to support a small number of updated attributes. However, when the number of updated attributes increases, the processing time sharply increases. Furthermore, in Yang scheme, the type of operations where the attributes are added (especially AND gate) also introduces more cost for ciphertext update.

Chapter 6: Conclusion and Future Work

6.1 Conclusion

In this dissertation, we propose the access control scheme called Collaborative Ciphertext-Policy Attribute Role Based Encryption (C-CP-ARBE) providing expressive, scalable, and effective access control solution for collaborative data sharing in multi-owner and multi-authority cloud computing. C-CP-ARBE encompasses four major technical contributions including a core access control supporting read and write access enforcement, reduction of key management cost, efficient user and attribute revocation, and dynamic and secure policy update management. The details of these contributions are summarized as follows.

In Chapter 3, we introduced the core construction of the proposed access control scheme, C-CP-ARBE based on the integration of RBAC model and CP-ABE. Accordingly, an attribute-based encryption feature concerting with benefits of RBAC that simplifies user management, enables fine-grained privileges (read, write) is achieved. We also introduce the user decryption key graph model and two-layer encryption to minimize key complexity and revocation process.

Significantly, the proposed key management and 2LE method eliminate the key distribution cost and offer drastic reduction of user revocation cost enabling C-CP-ARBE is practical and efficient in multi-user access control. Furthermore, we presented a write access enforcement scheme to enable the flexible and secure data update to the users having write privilege. This does not only address the limitation of the original CP-ABE scheme that lacks the write access enforcement, but it also reduces the computation and administrative burdens in supporting encryption service and local policy maintenance at data owner side. Finally, the experiments and performance evaluation is conducted to demonstrate the efficiency of our cryptographic algorithms (encryption and decryption) and our user revocation scheme.

In Chapter 4, we investigated the problem of attribute revocation in CP-ABE implementations. We introduced a new attribute-based proxy re-encryption scheme called VL-PRE (Very Lightweight Proxy Re-encryption) as an optimized PRE scheme. VL-PRE aims to reduce communication cost as well as computation cost at both client and cloud side. VL-PRE scheme advance the naïve PRE with new two processes based on key update strategy. Instead of re-generating re-encryption key for every revocation cases, the valid re-encryption key is updated with minimal cost. In addition, VL-PRE requires a smaller key size compared to the traditional PRE for re-encryption key computation. To evaluate the efficiency of PRE, we did the comparative analysis of computation cost between VL-PRE and related works and conducted the simulations to assess the performance of VL-PRE. The results confirm that VL-PRE delivers more performance and scalability for supporting attribute revocation than the existing PRE schemes.

In Chapter 5, we described and evaluated new algorithms for the updating of access policies in C-CP-ARBE system. Technically, we employ VL-PRE to support the optimization of data re-encryption cost. With the policy updating algorithms developed and VL-PRE, the update of attribute-based access policies and the execution of data re-encryption process are done with efficient and cost-optimized way. This also reduces the client-side overhead of communication, computation, and overall bookkeeping. The policy update feature advances the CP-ABE based access control with the criteria including correctness, security, and accountability. The performance evaluation is also given to substantiate the optimization of our proposed method.

6.2 Future Work

For future work, we identify challenges on extending the research for enhancing the security and usability of outsourced data as follows.

6.2.1 Secrecy and Privacy of Access Control Policy

Our access control scheme supports write privilege enforcement in the way that users having write privilege can access the policy shared on the cloud in the encrypted format. However, the policy content can be accessed by these users. In addition, in CP-ABE, access policy is usually applied to encrypt the plain data and is carried with the ciphertext. In a real-world system, policies may contain sensitive information that must be hidden from untrusted parties or even the users of the system. Therefore, access control policy privacy becomes more crucial for the attribute-based encryption model including our C-CP-ARBE. When the hidden policy is applied, the efficiency of encryption and decryption should also not be degraded. Therefore, the cryptographic techniques for supporting both policy privacy preservation and encryption and decryption efficiency are worth to be explored.

6.2.2 Secure and Efficient Search over Encrypted Data

Due to the popularity and the increased adoption of cloud computing, the capability for serving data management including search and query is essential. However, the data stored at the cloud server are usually encrypted; search capability over ciphertext is thus limited. Existing searchable encryption based on keyword search are not suitable for outsourced data since some keywords may expose the privacy and security problem. In addition, the performance for searching the data outsourced to support realtime query is another important factor usually demanded by the applications or users. Therefore, applying searchable symmetric or asymmetric encryption techniques that are both resistant to keywords attack and efficient in terms of search performance is another open issue in the cloud security research.

6.2.3 Privacy-Preserving Public Auditing for Outsourced Data with the Capability of Anomalies Detection

In addition to the secure and efficient access control, the integrity auditing of outsourced data is also very challenging and important. Enabling public auditability for cloud data storage is crucial since it guarantees data privacy and availability of healthy data. Furthermore, batch and online auditing of outsourced data is currently required by many organizations for their international security standard compliance. Trusted-third auditing party can be delegated to check the integrity of outsourced data when needed. Any errors (such as data duplication, false data injection) or incidents related to data integrity violation should be automatically detected and the data owners or users are notified for the errors. However, the auditing process must not generate overheads to degrade the data access operations. The auditor is also not allowed to breach the privacy of data while its integrity is checked. To do so, lightweight cryptographic protocols such as proxy signature, homomorphic encryption, or any advanced integrity checking techniques in data mining and intrusion detection system (IDS) are possible to be applied.

Bibilography

[1] S. De Capitani di Vimercati, S. Foresti, S. Jajodia, S. Paraboschi, and P. Samarati, "Encryption policies for regulating access to outsourced data," In ACM Transactions on Database Systems (TODS), vol. 35, no. 2, 2010, pp.12:1-12:46.

[2] M. Raykova, H. Zhao, and S. Bellovin, "Privacy enhanced access control for outsourced data sharing," In Proceedings of Financial Crypto 2012 (FC 2012), pp.223-238, Springer, 2012.

[3] Shucheng Yu, Cong Wang, Kui Ren, and Wenjing Lou, "Achieving secure, scalable, and fine-grained data access control in cloud computing," In Proceedings of IEEE INFOCOM 2010, IEEE, 2010, pp.1-9.

[4] R. Bobba, H. Khurana, and M. Prabhakaran, "Attribute-sets: A practically motivated enhancement to attribute-based encryption," In Proceedings of ESORICS 2009, Springer, 2009, pp.587-604.

[5] H. Xiong, X. Zhang, D. Yao, X. Wu, and Y. Wen, "Towards end-to-end secure content storage and delivery with public cloud," In Proceedings of ACM CODASPY 2012, IEEE, 2012, pp.257-266.

[6] S. De Capitani di Vimercati, S. Foresti, S. Jajodia, S. Paraboschi, and P. Samarati, " Support for write privileges on outsourced data,". In Proceedings of SEC 2012, Springer, 2012, pp.199-210.

[7] Z. Wan, J. Liu, R. H. Deng, "HASBE: A hierarchical attribute-based solution for Flexible and scalable access control in cloud computing," In IEEE Transactions on Information Forensics and Security, vol.7, no.2, IEEE, 2012, pp.743-754.

[8] M. Nabeel and E. Bertino, "Privacy-preserving fine-grained access control in public clouds," In IEEE Data Eng. Bull., vol. 35, no.4, IEEE, 2012, pp.21-30.

[9] M. Li, S. Yu, Y. Zheng, K. Ren, and W. Lou, "Scalable and secure sharing of personal health records in cloud computing using attribute-based encryption," In IEEE Transactions on Parallel and Distributed Systems, vol.24, no.1, IEEE, 2012, pp.131-143.

[10] K. Yang, X. Jia, K. Ren, B. Zhang, and R. Xie, "Expressive, efficient, and revocable data access control for multi-authority cloud storage," In IEEE Transactions Parallel Distributed System, IEEE, 2014, vol. 25, no.7, pp.1735-1744.

[11] V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-based encryption for fine-grained access control of encrypted data," In Proceedings of ACM Conference on Computer and Communications Security (CCS 2006), ACM, 2006, pp.89-98.

[12] M. J. Atallah, K. B. Frikken, and M. Blanton, “Dynamic and efficient key management for access hierarchies,” In Proceedings of ACM Conference on Computer and Communications Security (CCS 2015), ACM, 2005, pp.190–202.

[13] M. Chase, “Multi-authority attribute based encryption,” In Proceedings of the 4th Theory of Cryptography Conference on Theory of Cryptography (TCC 2007), Springer, 2007, pp. 515-534.

[14] M. Chase and S. S. M. Chow, “Improving privacy and security in multi-authority attribute-based encryption,” In Proceedings of the 16th ACM Conference on Computer and Communications Security (CCS 2009), ACM, 2009, pp.121-130.

[15] A. B. Lewko and B. Waters, “Decentralizing attribute-based encryption,” In Proceedings of the 30th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Advances in Cryptology (EUROCRYPT 2011), Springer 2011, pp.568-588.

[16] J. Bethencourt, A. Sahai, and B. Waters, “Ciphertext-policy attribute-based encryption,” In IEEE Symposium of Security and privacy (S&P 2007), IEEE, 2007, pp.321-334.

[17] N. Meghanathan, “Review of access control models for cloud computing,” In Proceedings of The Third International Conference on Computer Science, Engineering & Applications (ICCSEA 2013), Delhi, India, 2013, pp.77-85.

[18] A. Sahai and B. Waters, “Fuzzy identity based encryption,” In Advances in Cryptology (Eurocrypt 2005), Springer, 2005,pp.457-473.

[19] L. Zhou, V. Varadharajan, and M. Hitchens, “Enforcing role-based access control for secure data storage in the cloud,” In The Computer Journal, vol. 54, no.10, pp. 1675-1687.

[20] D. Boneh and M. Hamburg, “Generalized identity based and broadcast encryption schemes,” In Proceedings of International Conference on Advances in Cryptology (Asiacrypt 2008), Springer, 2008, pp.455-470.

[21] L. Zhou, V. Varadharajan, and M. Hitchens, “Achieving secure role-based access Control on encrypted data in cloud storage,” In IEEE Transactions on Information Forensics and Security, IEEE, 2013, vol. 8, no.12, pp.1947-1960.

[22] J. Hur and D. K. Noh, “Attribute-based access control with efficient revocation in data outsourcing systems,” In IEEE Transactions on Parallel Distributed System, IEEE, 2011, vol. 22, no. 7, pp.1214–1221.

[23] V. Goyal, A. Jain, O. Pandey, and A. Sahai, “Bounded ciphertext policy attribute based encryption,” In Proceedings of ICALP, Springer, 2008, pp.579-591.

[24] L. Cheung, C. Newport, "Provably secure ciphertext policy ABE," In Proceedings of ACM Conference on Computer and Communications Security (CCS 2007), ACM, 2007, pp. 456-465.

[25] R. S. Sandhu, D. F. Ferraiolo, D. R. Kuhn, "The NIST model for role-based access control: Towards a unified standard," In Proceedings of ACM Workshop on Role-Based Access Control (RBAC 2000), ACM, 2000, pp.47-63.

[26] M. Mambo and E. Okamoto, "Proxy cryptosystems, delegation of the power to decrypt ciphertexts," In IEICE Transactionsactions, vol. E80-A(1) pp.54-63, 1997.

[27] P. K. Tysowski and M. A. Hasan, "Hybrid attribute- and re-encryption-based key management for secure and scalable mobile applications in clouds," In IEEE Transactions On Cloud Computing, IEEE, 2013, vol. 1, no. 2, pp.172-186.

[28] J. Lai, R.H. Deng, Y. Yang, and J. Weng, "Adaptable ciphertext-policy attribute-based encryption," In Proceedings of Paring Cryptography (Paring 2013). Springer, 2013, pp.199-214.

[29] Y. Kawai, "Outsourcing the Re-encryption Key Generation: Flexible ciphertext-policy attribute-based proxy re-encryption," In Proceedings of International Conference on Information Security Practice and Experience (ISPEC 2015), Springer, 2015, pp.301-315.

[30] K. Liang, J K. Liu, D. S. Wong, and W. Susilo, "An efficient cloud-based revocable identity-based proxy re-encryption scheme for public clouds data sharing," In Proceedings of European Symposium Research in Computer Security (ESORICS 2014), Springer, 2014, pp. 257-272.

[31] J. Son, D. Kim, R. Hussain, and H. Oh, "Conditional proxy re-encryption for secure big data group sharing in cloud environment," In Proceedings of IEEE INFOCOM Workshop on Security and Privacy in Big Data, IEEE, 2015, pp. 541-546

[32] S. Yu, C. Wang, K. Ren, and W. Lou, "Attribute based data sharing with attribute revocation," In Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security (ASIACCS 2010), ACM, 2010, pp.261-270.

[33] X. Liang, Z. Cao, H. Lin, and Jun Shao, "Attribute based proxy re-encryption with delegating capabilities," In Proceedings of the 4th ACM Symposium on Information, Computer and Communications Security (ASIACCS 2009), ACM,2009, pp. 276-286.

[34] K. Yang, X. Jia, K. Ren, R. Xie, and L. Huang, "Enabling efficient access control with dynamic policy updating for big data in the cloud," In Proceedings of the 33nd IEEE Conference on Computer Communications (INFOCOM 2014), IEEE, 2014 pp.2013-2021.

[35] S. Fugkeaw and H. Sato, “Embedding lightweight proxy re-encryption for efficient attribute revocation in cloud computing,” In International Journal on High Performance Computing and Networking, Inderscience, 2016, vol. 9, no.4, pp.299-309.

[36] Amazon Elastic Compute Cloud http://aws.amazon.com.

[37] S. De Capitani di Vimercati, S. Foresti, G. Livraga, and P. Samarati, “Practical techniques building on encryption for protecting and managing data in the cloud,” In The Newcode Breakers, Springer, 2016, pp.205-239.

[38] R. Jhawar, V. Piuri, and M. Santambrogio, “A comprehensive conceptual system level approach to fault tolerance in cloud computing,” In Proceedings of SysCon 2012, IEEE, 2012, pp. 1-5.

[39] S. Ruj, M. Stojmenovic, and A. Nayak, “Privacy preserving access control with authentication for securing data in clouds,” In Proceedings of CCGrid 2012, IEEE, 2012, pp. 556-563.

[40] Z. Fangming, N. Takashi, S. Kouichi, “Realizing fine-grained and flexible access control to outsourced data with attribute-based cryptosystems,” In Proceedings of ISPEC 2011, Springer, 2011, pp.83-97.

[41] Secure random number generator Java library, https://docs.oracle.com/javase/7/docs/api/java/security/ SecureRandom.html

[42] P. Angin, B. Bhargava, Z. Jin, “A self-cloning agents based model for high performance mobile cloud computing,” In Proceedings of IEEE International Conference on Cloud Computing (CLOUD 2015), IEEE, pp.301-308.

[43] S. Fugkeaw and H. Sato, “An extended CP-ABE based Access control model for data outsourced in the cloud,” In Proceedings of IEEE International Workshop on Middleware for Cyber Security, Cloud Computing and Internetworking (MidCCI 2015), IEEE, 2015, pp.73-78.

[44] S. Fugkeaw and H. Sato, “Enabling constraints and dynamic preventive access control policy enforcement in the cloud,” In Proceedings of IEEE International Workshop on Cloud Security and Forensics (WCSF 2015), IEEE, 2015, pp.576-583.

[45] S. Fugkeaw and H. Sato, “CLOUD-CAT: A collaborative access control tool for data outsourced in cloud computing,” In Proceedings of IEEE International conference of Digital Information and Management (ICDIM 2015), IEEE, 2015, pp.243-248.

[46] S. Fugkeaw and H. Sato, “Improved lightweight proxy re-encryption for flexible and scalable mobile revocation management in cloud computing,” In Proceedings of IEEE International Conference on Cloud Computing (CLOUD 2016), IEEE, 2016, pp.894-899.

[47] S. Fugkeaw and H. Sato, “Key Update As A Service (KAAS): An agent modeling for cloud-based access control,” In Proceedings of IEEE International Congress on Big Data (IEEE BigData), IEEE, 2016, pp.430-437.

[48] N.Doshi and D. Jinwala, “Updating attribute in CP-ABE: A new approach,” In IACR Cryptology ePrint Archive, Volume 2012 (2012).

[49] N. Kaaniche, M. Laurent, M. El Barbori, “CloudaSec: A novel public-key based framework to handle data storage security in cloud,” In Proceedings of 11th International Conference on Security and Cryptography (SECRYPT 2014), IEEE, 2014, pp.1-14.

[50] M. Peter and G. Tim. “The NIST Definition of Cloud Computing. National Institute of Standards and Technology,” September 2011.

[51] N. Kaaniche, E. El Moustaine, and M. Laurent, “A novel zero-knowledge scheme for proof of data possession in cloud storage applications,” In Proceedings of 14th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid 2014), IEEE, 2014, pp. 522–531.

[52] L. Krzywiecki and M. Kutylowski, “Proof of possession for cloud storage via lagrangian interpolation techniques,” In Proceedings of the 6th international conference on Network and System Security(NSS 2012), Springer. 2012, pp.305-319.

[53] J. Yu, K. Ren, and C. Wang, “Enabling cloud storage auditing with verifiable outsourcing of key updates,” In IEEE Transactions on Information Forensics and Security, IEEE, 2016, vol.11, no.6, pp.1362-1375.

[54] C. Wang, S. S. M. Chow, Q. Wang, K. Ren, and W. Lou, “Privacy-preserving public auditing for secure cloud storage,” In IEEE Transactions on Computers, IEEE, 2013, vol. 62, no.2, pp.362-375.

[55] N. Cao, Z. Yang, C. Wang, K. Ren, and W. Lou, “Privacy-preserving query over encrypted graph-structured data in cloud computing”, In Proceedings of International Conference on Distributed Computing Systems (ICDCS 2011), IEEE, 2011, pp.393-402.

[56] C. Wang, Q. Wang, and K. Ren, “Towards secure and effective utilization over encrypted cloud data,” In Proceedings of ICDCS Workshops 2011, IEEE, 2011, pp.282-286.

[57] Z. Mo, Q. Xiao, Y. Zhou, and S. Chen, “On deletion of outsourced data in cloud computing,” In Proceedings of IEEE International Conference on Cloud Computing (CLOUD 2014), IEEE, 2014, pp.344-351.

[58] B. Wang, H. Li, X. Liu, F. Li, and X. Li, “Efficient public verification on the integrity of multi-owner data in the cloud,” In Journal of Communications and Networks, vol. 16, no.6, pp. 592-599, 2014.

[59] A. Sahai, H. Seyalioglu, B. Waters, “Dynamic credentials and ciphertext delegation for attribute-based encryption,” In Proceedings of International Cryptology Conference, Springer, 2012, pp.199–217.

[60] K. Yang, X. Jia and K. Ren, “Secure and verifiable policy update outsourcing for big data access control in the cloud,” In IEEE Transions on Parallel and Distributed Systems (TPDS), IEEE, 2015, vol. 26, no.12, pp.3461-3470.

[61] S. Fugkeaw, and H. Sato, “Updating policies in CP-ABE based access control : an optimized and secure service,” In Proceedings of the European Conference on Service-Oriented and Cloud Computing (ESOCC 2016), Springer, 2016, pp.3-17.

[62] R. Ostrovsky, A. Sahai, B. Waters, “Attribute-based encryption with nonmonotonic access structures,” In Proceedings of ACM CCS 2007, ACM, 2007, pp.195–203.

[63] D. Naor, M. Naor, and J. Lotspiech, “Revocation and tracing schemes for stateless receivers,” In Proc. of CRYPTO 2001. Springer, 2001, pp.41–62.

[64] N. Attrapadung and H. Imai, H. “Attribute-based encryption supporting direct/indirect revocation modes,” In Proceedings of Cryptography and Coding 2009. Springer, 2009, pp. 278–300.

[65] https://cloudsecurityalliance.org/media/news/cloud-security-alliance-releases-the-treacherous-twelve-cloud-computing-top-threats-in-2016/

[66] National Institute of Standards and Technology, Security Requirements of Cryptographic Modules, FIPS 140-2, May 25,2001. Available at URL: http://www.nist.gov/cmvp.

[67] RFC 1750: Randomness Recommendations for Security, https://www.ietf.org/rfc/rfc1750.txt

[68] Y. Wang, F. Li, J. Xiong, B. Niu, and F. Shan, “Achieving lightweight and secure access control in multi-authority cloud,” In Proceedings of the 14th IEEE International Conference on Trust, Security and Privacy in Computing and Communications (IEEE TrustCom 2015), IEEE, 2015, pp.459-166.

[69] F. Guo, Y. Mu, W. Susilo, D. S. Wong, and V. Varadharajan, “CP-ABE with constant-size keys for lightweight devices,” IEEE Transactions on Information Forensics and Security, vol. 9, no. 5, pp.763–771, 2014.

[70] B. Wang, B. Li, and H. Li, “Public auditing for shared data with efficient user revocation in the cloud,” In Proceedings of the 32nd IEEE Conference on Computer Communications (INFOCOM 2013), IEEE, 2013, pp.2904–2912.

[71] J. Kelsey, B. Schneier, D. Wagner, and C. Hall, “Cryptanalytic attacks on pseudorandom number generators,” In Proceedings of 5th International Workshop of Fast Software Encryption (FSE 1998), Springer, 1998, pp.168-188.

[72] H. Lin, Z. Cao, X. Liang, and J. Shao, “Secure threshold multi authority attribute based encryption without a central authority,” In Information Science, vol. 180, no. 13, pp. 2618–2632, 2010.

[73] S. Jahid, P. Mittal, and N. Borisov, “Easier: encryption-based access control in social networks with efficient revocation,” In Proceedings of ASIACCS’11.ACM, 2011, pp.411–415.

[74] K. He, C. Huang, K. Yang, and J. Shi, “Identity-preserving public auditing for shared cloud data,” In Proceedings of the 23rd IEEE International Symposium on Quality of Service (IWQoS 2015), IEEE, 2015, pp.159-164.

[75] W. Li, H. Wan, X. Ren, and S. Li, “A refined RBAC model for cloud computing,” In Proceedings of IEEE/ACIS Conference on Computer and Information Science (ICIS 2012), IEEE, 2012, pp.43-48.

[76] S. Wang, J. Zhou, J. Liu, J. Yu, J. Chen, and W. Xie, An efficient file hierarchy attribute-based encryption scheme in cloud computing, In IEEE Transactions on Information Forensics and Security, vol.11, pp.1265-1277, 2016.

[77] Secure random number generator Java library, https://docs.oracle.com/javase/7/docs/api/java/security/ SecureRandom.html

[78] Java Paring -Based Cryptography Library (JPBC) http://gas.dia.unisa.it/projects/jpbc/download.html

[79] W. C. Garrison III, A. Shull, S. Myers, and A. J. Lee, “On the practicality of cryptographically enforcing dynamic access control policies in the cloud,” In Proc. IEEE Symposium on Security and Privacy (S&P 2016), San Jose, CA, USA, May 23-25, 2016.

[80] W. Lueks, G.Alpár, J. Hoepman, and P. Vullers, “Fast revocation of attribute-based credentials for both users and verifiers,” In Proceeings of the IFIP International Information Security and Privacy Conference (IFIP SEC 2015), Hamburg, 2015, pp.463-478.

[81] S. Fugkeaw and H. Sato, “Achieving scalable and optimized attribute revocation in cloud computing,” In IEICE Transactions on Information and Systems, Vol.E100-D,No.5, pp.973-983,May. 2017.

[82] S. P. Vadhan, Pseudorandomness, Foundations and Trends® in Theoretical Computer Science: Vol. 7: No. 1–3, pp.1-336, December, 2012.

Authors' Publications

International Journals

1. **Somchart Fugkeaw** and Hiroyuki Sato, Scalable and secure access control policy update for outsourced big data, to appear in Future Generation Computer Systems, ELESVIER, 2017.
2. **Somchart Fugkeaw** and Hiroyuki Sato, Achieving scalable and optimized attribute revocation in cloud computing, IEICE Transactions on Information and Systems, Vol.E100-D, No. 5, pp.973-983, 2017.
3. **Somchart Fugkeaw** and Hiroyuki Sato, 'Embedding lightweight proxy re-encryption for efficient attribute revocation in cloud computing', Int. J. High Performance Computing and Networking, Vol. 9, Issue no.4, pp.299-309, 2016.
4. **Somchart Fugkeaw** and Hiroyuki Sato, Design and Implementation of collaborative ciphertext-policy attribute-role based encryption for data access control in cloud, Journal of Information Security Research, Vol. 6, Issue no. 3, pp.71-84, 2015.

International Conferences

5. **Somchart Fugkeaw** and Hiroyuki Sato, An extended CP-ABE based access control model for data outsourced in the cloud, In Proceedings of IEEE International Workshop on Middleware for Cyber Security, Cloud Computing and Internetworking (MidCCI 2015), IEEE, 2015, pp.73-78.
6. **Somchart Fugkeaw** and Hiroyuki Sato, Enabling constraints and dynamic preventive access control policy enforcement in the cloud, In Proceedings of IEEE International Workshop on Cloud Security and Forensics (WCSF 2015), IEEE, 2015, pp.576-583.
7. **Somchart Fugkeaw** and Hiroyuki Sato, CLOUD-CAT: A collaborative access control tool for data outsourced in cloud computing, In Proceedings of IEEE International conference of Digital Information and Management (ICDIM 2015), IEEE, 2015, pp.243-248.
8. **Somchart Fugkeaw** and Hiroyuki Sato, A privacy-preserving access control model for big data cloud, In Proceedings of IEEE International Computer Science Engineering Conference (ICSEC 2015), IEEE, 2015, pp.1-6.
9. **Somchart Fugkeaw** and Hiroyki Sato, Improved lightweight proxy re-encryption for flexible and scalable mobile revocation management in cloud computing, In Proceedings of IEEE

International Conference on Cloud Computing (CLOUD 2016), IEEE, 2016, pp.894-899.

10. **Somchart Fugkeaw** and Hiroyuki Sato, "Key Update As A Service (KAAS): An agent modeling for cloud-based access control," In Proceedings of IEEE International Congress on Big Data (IEEE BigData), IEEE, 2016, pp.430-437.
11. **Somchart Fugkeaw** and Hiroyuki Sato, "Updating policies in CP-ABE based access control : an optimized and secure service," In Proceedings of the European Conference on Service-Oriented and Cloud Computing (ESOCC 2016), Springer, 2016, pp.3-17.
12. **Somchart Fugkeaw** and Hiroyuki Sato, "Enforcing hidden access control policy for supporting write access in cloud storage systems, In Proceedings of International Conference on Cloud Computing and Service Sciences (CLOSER 2017), 2017, pp.558-564.