Generic Access Control of Cloud Storage Using Attribute-Based Cryptography

Zhiqian Xu

Thesis submitted to the University of London
for the degree of Doctor of Philosophy

Information Security Group
School of Mathematics and Information Security
Royal Holloway, University of London

2018

Declaration of Authorship

I, Zhiqian Xu, hereby declare that this thesis and the work presented in it is entirely my own. Where I have consulted the work of others, this is always clearly stated.

Signed:

Date: March, 2018

Acknowledgments

First and formost I would like to express my deepest appreciation to my supervisor, Professor Keith Martin; for agreeing to supervise me although I have to study as a remote part-time student, for motivating me and reminding me to stay focused, for teaching me how to do research and write research papers, for correcting my mistakes, and for being patient throughout these years. Without him I would not have reached this far.

Throughout my study at RHUL, I have been advised by many professors, in particular, I would like to thank Professor Jason Crampton, Professor Stephen Wolthusen, Professor Kenny Paterson, and Professor Carlos Cid for coming to my annual reviews; for inspiring discussions, wise guidance, and unconditional support. I would also like to thank my thesis committee members, Professor Liqun Chen and Professor Geraint Price, for their tremendous efforts as well as valuable reviews and feedbacks.

A special thank to Po-Wah Yau for introducing RHUL to me, for encouraging me and explaining that PhD actual stands for permanent head damage; for offering me help when I visited the campus. Another special thank to Jenny Lee, for answering my questions, particularly helping me to book the conference travels I needed.

Lastly I would like to thank my wonderful family, my husband and daughter, for understanding and knowing my limits, and for always being there when I need them.

Thank you all. You make my life over those years unforgettable.

Abstract

Cloud storage provides cost-efficient storage services in an era of increased demands on data generation and reliance. However, since cloud storage providers might not be trusted by end users, security and privacy are a major concern to data owners. Although cryptography is traditionally used to support data confidentiality, integrity and availability, data access control is yet another potential field. Cryptography can help achieve access control and authentication effectively.

Data access control can be achieved by two major approaches: server-mediated access control and cryptographically-enforced access control. In server-mediated access control, servers receive access attempts and grant permissions based on pre-defined policies. This requires storage servers to be fully trusted, which is not the case in many storage environments. Cryptographically-enforced access control does not require the storage servers to be fully trusted since the data stored on storage servers is encrypted. Data access control is enforced through the management of decryption keys. If the key distribution process is not managed by cloud service providers directly, data secrecy and user privacy will be protected in untrusted cloud storage environments. However, if data owners are directly involved in key distribution, access control could become cumbersome to operate in a cloud storage environment. Therefore, scalable and flexible access control schemes are necessary for securing storage systems in untrusted clouds.

Attribute-based Encryption (ABE) provides both cryptographically-enforced confidentiality and access control in cloud storage environments. This allows data to be protected with automatically enforced access policies. Key distribution can be delegated to attribute authorities

that do not require the direct involvement of data owners. However, current access control schemes associated with existing ABE schemes are inflexible and have limitations concerning dynamic user and attribute revocation, key refreshing and revocation, and key escrow.

Attribute-based signature (ABS) schemes can also be used to facilitate anonymous access control and allow users to sign messages without disclosing their identities. However, access control schemes associated with existing ABS schemes also have practical limitations concerning dynamic user and attribute revocation, in particular anonymous user revocation.

In this thesis, we improve the practicality of access control mechanism for securing data stored on untrusted cloud storage. We adopt existing attributed-based cryptography and provide them with more flexible user and attribute management capability. We propose two deployment models and three systems that can be used with existing ABE and ABS schemes with no, or minor, modifications. Those adaptations reduce the management overheads and improve the scalability of attributed-based cryptography deployment to support security in cloud storage environments.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Contents

The need for distributed data storage has been driven by distributed computing environments and data sharing demands. Distributed data storage is the result of advances in network infrastructures and technologies of the past decades [78]. Over the years storage systems have rapidly evolved. They have become one of the most important components in every business, as well as something of significance to individual users. As the amount of electronically stored data increases rapidly every day, tremendous pressure has been placed on storage infrastructure to scale well with the growth of data while continuing to provide high performance.

The amount of electronically stored sensitive data, such as healthcare records, customer records or financial data, has also increased rapidly. Since data tends to be shared, replicated, and retained online in order to satisfy various information system requirements, such as performance, availability, and recovery, storage systems are becoming much more complicated and vulnerable to security breaches. The goals of storage security are to protect data from unauthorized access and changes and to prevent data secrecy from being disclosed.

Cloud-based storage is a newly emerged service model to both businesses and individual users for either paid or free storage resources. Due to its friendly accessibility, unlimited capacity, and economic benefits, users find it attractive. The advantage of cloud storage over distributed storage systems (see Section 2.1) is its ability to leverage virtualization techniques to provide a storage service composed of thousands of different types of networked storage devices, distributed storage systems, and storage middleware. This enables on-demand service, capacity, and management to users anywhere via the Internet.

Since most cloud storage is provided by third-party service providers, the trust required for the providers (CSPs) and the shared multi-tenant environment, resulting in storage virtualization, presents special challenges for data protection and access control [78]. These unique challenges also raise the concerns of user privacy protection for data access. Although user privacy protection is not the most obvious goal of storage security, we will explain why it becomes important in untrusted storage environments in Section 2.5.1.

Attribute-Based Encryption (ABE), a type of public key encryption, has a one-to-many relationship between a public encryption key and a set of decryption keys. ABE not only protects data secrecy, but also has ciphertexts or decryption keys associated with fine-grain access policies that are automatically enforced during the decryption process. This enforcement puts data access under control at each data item level. Descriptive access policy is not directly tied in to an individual user's identity, but rather to a set of common attributes among users who need the access. This attribute indirectly provides a kind of anonymous data access for user privacy protection. ABE schemes can be classified into two categories: *key-policy ABE* (KP-ABE) (see Section 4.1.1.2) and *ciphertext-policy ABE* (CP-ABE) (see Section 4.1.1.3).

Although ABE seems to be an ideal choice for meeting data protection goals in an untrusted storage environment, existing ABE schemes have practical limitations when they are directly used in cloud storage systems. These limitations include the lack of support for dynamic user and attribute revocation, key refreshing, and an inbuilt and potentially undesirable key escrow.

Inspired by ABE, *Attribute-Based Signature* (ABS) schemes enable anonymous authentication and signature generation. Since CSPs can be untrusted, we do not want CSPs to manage user identities which can contain or be associated with users' personal identifiable information (PII), or user accounts with passwords for authentication. Using ABS schemes, CSPs only need a public key and its associated access policy (referred to as predicate) to authenticate users. There is no need for CSPs to know or store user identities or login information. Although ABS seems to be an ideal anonymous authentication mechanism, existing schemes also have limitations concerning the lack of dynamic user revocation, especially anonymous user revocation.

1.1 Motivation and Research Statement

Motivated by the practical limitations of ABE and ABS, in this thesis we pose the following central research question:

How can practical ABE and ABS schemes be constructed to enforce distributed data and user privacy protection efficiently and flexibly in untrusted cloud storage environments?

Throughout this thesis we assume that CSPs are not fully trusted, meaning that they are trusted to do what they are supposed to do, but are also curious to learn the secrecy of data or user identities (see Section 2.5.1). Using ABE and ABS schemes to protect data and user privacy in these kinds of cloud environments requires:

- ABE schemes to be capable of preventing revoked users or attributes from decrypting data;
- ABS schemes to be capable of revoking user access rights;
- User access control to be anonymous with respect to untrusted cloud storage or CSPs and not require any administration of user management from CSPs.

The solutions to the above requirements require enabling of the following features:

- ABE schemes: dynamic user and attribute revocation. These features need to avoid expensive re-issuing of users' private keys, and the revocation processes need to have built-in user privacy protection.

- ABS schemes: dynamic user revocation. The revocation process avoids expensive re-issuing of users' private (signing) keys and is anonymous in that the authenticating party should be able to distinguish the signatures from the revoked users without being able to link any signature to any particular user.

Although issues concerning user and attribute revocation in ABE schemes have been studied in the literature, most of the proposed schemes work for only one type of ABE scheme. User privacy protection has been left out from those revocation systems. Since each ABE scheme has its own strengths, it is more desirable that the developed mechanism is as generic as possible so that it can be directly adapted by any ABE scheme to address user and attribute revocation. In addition, a revocation process should have built-in user privacy protection in untrusted cloud storage systems.

The existing ABS user revocation schemes often need to uncover a user's identity during the signature verification process in order to distinguish if the signatures are generated by revoked users or not. Many existing schemes employ a trusted entity to maintain and identify user identities in order to distinguish the signatures from revoked users. We argue that this setting conflicts with the anonymity property of ABS. The user revocation process should also be anonymous in that the signatures of revoked or non-revoked users should be distinguishable with neither knowledge of an individual's identity nor signature verification by a third party.

1.2 Our Contributions

In this thesis we propose practical systems and adaptable models for using ABE and ABS schemes in untrusted cloud storage. Those models and systems enable ABE and ABS to overcome the possible limitations and make them more suitable for untrusted cloud storage environments. Before we describe our proposed models and systems, we want to clarify two things:

1. *Model* versus *System*: Although all the proposed mechanisms are schemes, we use *model* to indicate that a proposed scheme does not treat an ABE or ABS scheme as a black box. A *model* needs to be adapted to fit an ABE or ABS scheme, ideally with only minor modification to the scheme. A *system* is a scheme that treats an ABE or ABS scheme as a black box. It builds a wrapper outside an ABE or ABS scheme, which can then be directly applied without modification.

2. Characteristics of proposed models and systems:

    - *Flexible*: the proposed model can work directly for a scheme with at most minor modifications to the scheme.
    - *Generic*: the proposed system can be directly applied to a scheme with no modification to the scheme.
    - *Dynamic*: the proposed model or system can avoid user private or signing keys needing to be re-issued as well as data needing to be re-encrypted.

- *User privacy protection*: the proposed model or system has built-in user privacy protection that allows a user to access data anonymously or to be verified anonymously by cloud storage service providers.

We highlight our main contributions as follows:

1. A survey of distribute storage systems. This survey investigates the security issues and challenges concerning distributed and cloud storage systems. It then identifies the security requirements for data and user privacy protection in untrusted cloud storage environments.

2. Two *generic* user revocation systems for ABE with *user privacy protection.* The two systems, UR-CRE and UR-CSP, work with any type of ABE scheme to dynamically revoke users.

3. *A generic* attribute revocation system for ABE with *user privacy protection* (AR-ABE). AR-ABE works with any ABE scheme to dynamically revoke attributes. Any attribute revoked from a user does not impact the same attribute used by other users.

4. An *Anonymous* user revocation model (AUR-ABS) for ABS with *user privacy protection.* An ABS scheme with AUR-ABS model enables CSPs to anonymously identify whether a signature is generated by a revoked user or not during the signature verification process.

1.3 Outline of the Thesis

The remainder of this thesis is organized as follows:

- *Chapter 2: Security Services and Techniques Suitable for Distributed and Cloud Storage Systems*

We survey the existing distributed and cloud storage systems and identify the security services and techniques suitable for data protection and access control in these environments. We then define the special requirements of data and user privacy protection in untrusted cloud storage environments. Based on these requirements, we compare existing data and user privacy protection mechanisms and their possible shortcomings in cloud storage environments. Most of the content of this chapter was published in [144].

- *Chapter 3: Cryptographic Preliminaries and Background*

We present relevant background material and notation necessary to understand the remainder of the thesis. We start with a brief introduction to relevant notions from mathematics. We then review the concepts of modern cryptography and complexity theory that will be used later.

- *Chapter 4: Attribute-Based Cryptography*

We introduce attribute-based encryption (ABE) and attribute-based signature (ABS) schemes.

We begin with a description of the concept of ABE followed by related work. We then introduce the concept of an ABS scheme and describe the construction of a threshold ABS scheme [83], which will be used in Chapter 7.

- *Chapter 5: Generic User Revocation Systems for ABE in Cloud Storage*

In this chapter we propose two dynamic user revocation systems that are generic for any ABE scheme. These systems have user privacy protections and are suitable to be used in untrusted cloud storage environments.

- *Chapter 6: A Generic Attribute Revocation System for ABE in Cloud Storage*

We propose a user attribute revocation system that meets the requirements of being generic, dynamic, and having built-in user privacy.

- *Chapter 7: An Anonymous User Revocation Model for ABS in Cloud Storage*

We propose an anonymous user revocation model (AUR-ABS) for ABS schemes. Our user revocation model is dynamic and open for adaptation by ABS schemes. The model not only removes the involvement of a third party in the path of signature generation or verification, but also makes the user revocation completely anonymous. The content of this chapter was published in [143].

- *Chapter 8: Concluding Remarks*

In this chapter we briefly talk about our research journey, conclude the thesis and suggest future directions for research.

Chapter 2

Security Services and Techniques Suitable for Distributed and Cloud Storage Systems

Contents

The rapid growth of data and data sharing has driven an evolution in distributed storage systems. The need for data protection and greater capacity for handling massive data volume has encouraged research to develop secure and scalable storage. In this chapter we identify major security issues and requirements of data protection related to distributed data storage systems, in particular to cloud storage systems.

2.1 Distributed Data Storage Systems

Distributed data storage can be built from many small-scale unreliable, or low availability, distributed hosts. Users access distributed data storage though their local networks or the Internet.

Generally speaking, there are four types of common storage infrastructures [66]: *Direct Attached Storage* (DAS) [4], *Network Attached Storage* (NAS) [54], *Storage Area Networks* (SAN) [54], and *Object Storage* [96]. Most current distributed storage systems are built on these common storage infrastructures.

From a data formatting point of view, we classify different types of storage as file, block, and object (also referred to as object-based) storage. File storage stores data in individual files organized by folders and file hierarchy [119]. Block storage splits data or files into evenly sized blocks, with each block associated with its address within disk sectors and track [2]. Object storage splits data or files into data objects. Each object is self-contained with its metadata and a globally unique identifier [50]. The data sizes and formats of objects can vary with no hierachy structure. This makes object storage be more efficient on data searching, mining, and analytics. Amazon S3 [3] is an example of cloud-based object storage systems.

From the user and application standpoints, we classify distributed data storage systems into centralized, decentralized or hybrid storage systems [144]. In a centralized storage system, data and files are managed by a central component. A uniform interface provides a single point of view to the underlying storage system. In a distributed management system, data and file management are distributed to individual storage servers or devices. Users have to access each individual storage server to locate or access data. A hybrid storage system separates data access and metadata management from the data I/O path. Data access is through a central component, while data retrieval can be conducted directly from individual storage servers or devices simultaneously. This hybrid approach allows data to be split into data blocks. These data blocks are stored on different network servers, which allows unlimited sizes of data files. In addition, simultaneously retrieving data blocks from different network servers achieves maximum throughput.

2.2 Data Protection Challenges in Distributed Data Storage Systems

Although distributed data storage provides many benefits, protecting distributed data storage introduces many challenges:

1. Data is highly distributed across a network, thus increasing management complexity and introducing more vulnerability points with respect to data integrity and privacy.

2. A decentralized system can be built on different sub-systems. Those sub-systems may use different storage infrastructures, data or file formats, and storage technologies. This complexity creates more vulnerability to security breaches, as different storage infrastructures may be vulnerable to different security threats. The security of the whole system depends on the security of the weakest sub-system.

3. Highly distributed storage systems can consist of storage systems residing anywhere, possibly even across different security domains or regions. Therefore, sensitive data protection in distributed storage systems not only serves to control who can access the data, but also needs to secure data in transit and at rest in different domains and regions. Some legislation and regulatory requirements for data access and protection may be different.

4. Legislation has placed more security demands for the preservation and retrieval of long-term stored data. Extended retention time provides wider time windows for attackers to discover or damage data. Dramatically increased computing power and resources can

result in the strength of encryption algorithm used for encrypting the archived data (when the data was stored) being regarded as weaker over time. Backward compatibility after data migration, data format translation, re-encryption of data without any violation of secrecy policy due to the weaker strength of the used encryption algorithm, authorized integrity correction due to the corruption or malicious attacks, and data recovery due to delete of, are examples of long-term data protection issues.

5. Although there are many data protection mechanisms available today, no single method can protect a distributed data storage system and address all vulnerabilities. For example, encryption can protect data secrecy but cannot address data recovery and availability. Authentication and authorization can prevent unauthorized data access, but cannot replace integrity and confidentiality protection. The use of data replication may provide availability and fault tolerance but may also increase storage complexity and exposure to confidentiality and integrity attacks.

Different distributed data storage systems are vulnerable to different threats. When selecting and designing a distributed data storage system, we have to weigh trade-offs of security countermeasures, system performance, and network overheads within specific environments. Therefore it is very important to understand the threats to a specific distributed data storage system before implementing any security features.

2.3 Threats to Distributed Data Storage Systems

The goals of storage security are to protect data from unauthorized access and alteration as well as prevent sensitive data from being disclosed to non-legitimate users or applications at rest or in transit. Understanding threats at each entry point and layer is essential to selecting the right protection strategies. Distributed data storage protection includes trade-offs. For example, encryption can impact performance, usability, and capability for data recovery. Data replication provides availability, but can open up more entry points for attacks. Storage standards which define interoperability for various storage systems also need to be considered when selecting the right countermeasures.

Confidentiality, integrity, and availability (*CIA triangle*) are defined by the Committee on National Security Systems (CNSS) as the three core characteristics of protecting critical data [139]. The CIA triangle has long been an industry “model” for computer and data security. Although the CIA triangle model is still mentioned as the industry standards for data protection, it does not adequately address today’s environments. While CIA deals with data protection, data also needs to be accessed by users. Data security also needs consideration of access control, which restricts un-authorized users or entities from accessing data. Data access control should consist of authentication (a process of identifying users who want to access data) and authorization (a process to prevent the data from unauthorized access or modification). In untrusted storage environments, an authentication mechanism also needs to protect user privacy for the reasons explained in Section 2.5.1.

In this thesis, we consider data security in four dimensions: confidentiality, integrity, availability, and access control (CIAA). Hasan et al. [66] classifies storage threats based on the four dimensions as follows.

2.3.1 Threats to Confidentiality

General threats to confidentiality include sniffing storage traffic, snooping on buffer caches, deleting storage blocks, de-allocating memory, and file system profiling [110] (an attack that uses system metadata to gain knowledge of system's operational information, such as Input/Output performance, file accessing protocols, CPU power, etc. The metadata information can assist attacking on accessing decrypted data, or encryption keys). Storage and backup media can be stolen in order to access the data or brute force keys.

2.3.2 Threats to Integrity

General threats to integrity include storage jamming (a malicious but surreptitious modification of stored data) to modify or replace the original data, metadata modification to disrupt a storage system, subversion attacks to gain unauthorized OS level access to modify critical system data, man-in-the-middle attacks to change data contents in transit, exploitation of storage backup procedures or applications, hardware failures, and long-time data preservation providing a large time window for adversaries to find collisions of hash values.

2.3.3 Threats to Availability

General threats to availability include denial-of-service (DoS) or distributed DoS (DDoS), disk fragmentation, network disruption, and file deletion. Buffer overflows in applications have allowed attackers to take control of systems, execute malicious code, and launch DoS attacks. Centralized data location management or indexing of servers can be points of failure or victims of DoS attacks. For long-term data archive systems, data retention introduces additional challenges such as long-term key management for encrypted data, constant protection against new threats, backward system compatibility, new technology adoption, and data recovery [125].

2.3.4 Threats to Access Control

General threats to access control include privilege escalation to access unauthorized data, session hijacking, man-in-the-middle attacks, authentication token theft, and replay attacks that trick the system into performing unauthorized operations.

2.4 Data Protection Mechanisms

Common mechanisms for preventing or minimizing the risks posed by those threats listed in section 2.3 are access control, confidentiality protection, integrity protection, data availability and recovery mechanisms, and auditing.

2.4.1 Access Control

Access control typically includes authentication and authorization. The access control type can be either centralized or decentralized. In both types, the control validates legitimate entities (authentication) for accessing data according to pre-defined policies (authorization). The access privileges of an entity need to be periodically reviewed or re-granted to prevent replay and impersonation attacks, as well as to remove excessively granted privileges from an entity if those privileges are no longer needed.

2.4.2 Confidentiality Protection

Encryption and secret sharing are commonly used methods for confidentiality protection. Encryption uses keys to encrypt data. Key management is critical, in addition to key or encryption algorithm strength. A widely distributed environment requires the key management to be flexible and adaptable for key distribution, refreshing and revocation. Secret sharing splits secrets into multiple shares. It protects data without the use of keyed encryption. Thus, it replaces key management with secret share management. However, those secret shares also require periodical secure transfer, access, and renewal. Since a secret sharing mechanism can generate multiple data shares, with each share potentially being the same size as or larger than the original data size, more storage space and additional network communication are required. These overheads may impact system performance.

2.4.3 Integrity Protection

Integrity mechanisms protect data from unauthorized modification, while non-repudiation mechanisms prevent an entity from denying a modification of the data. Storage integrity violation has two different aspects: hardware or software malfunctions and malicious intentions [61]. Integrity protection against hardware or software malfunction is typically provided by technologies, such as mirroring, parity, erasure codes, or hash functions. Integrity despite malicious attacks requires digital signatures or message authentication codes (MAC). Both mechanisms can detect unauthorized modification of data, while digital signatures additionally provide non-repudiation verification.

2.4.4 Data Availability and Recovery

Data availability and recovery functionality are very important to critical data for users and applications. Availability is particularly important to accessibility to critical long retention data. Recovery mechanisms repair damaged data or re-encrypt the data when a key gets lost, a key is compromised, or selected encryption algorithms become vulnerable to known attacks. Replication or data redundancy provides both data availability and recovery capability. Intrusion detection or prevention mechanisms detect or prevent malicious activity aimed at stealing or damaging data.

2.4.5 Auditing

Maintaining audit logs is critical for storage systems. Audit logs not only provide evidence of compliance but also form the basis for system recovery, intrusion detection, and computer forensics.

2.5 Cloud Storage

Cloud storage is a form of distributed storage that provides massive storage resources and services to meet on-demand data center growth for corporations as well as remote storage for small businesses and individuals. Cloud storage is a service model provided by cloud computing [67, 94, 137], which is a computing paradigm that uses Internet-based services to support business needs. It allows consumers to pay for IT services on a utility-like basis, using a shared pool of configurable computing resources (e.g. networks, servers, storage, applications, and services). The main attributes of cloud computing include rapid deployment, low start-up costs and capital investment, costs based on usage and subscription, and multi-tenant sharing of services and resources. Figure 2.1 shows the NIST cloud computing model [94].

As a type of infrastructure as a service (IaaS) offered by cloud computing, cloud storage pools and consolidates storage resources that can be administered on demand. It is a storage model that may run on different storage infrastructures, multiple devices, in many application domains, or with the inclusion of many services [149]. Cloud storage provides consumers with a

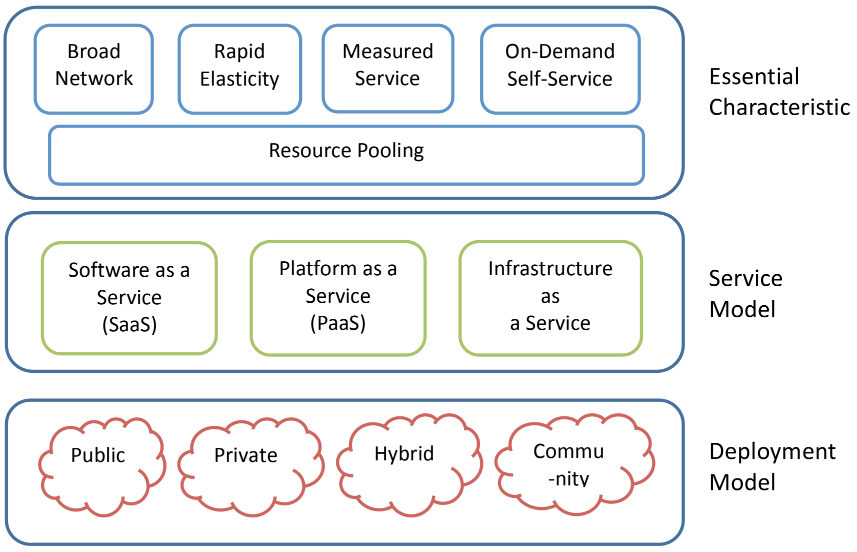


Figure 2.1: NIST Cloud Computing Model

unified interface that abstracts the different storage infrastructures and physical locations of the storage, making whether the storage is local or remote (or hybrid) irrelevant. This means that cloud storage is by default shared by many users on many layers of the storage stack. This shared environment is referred to as multi-tenancy. Figure 2.2 shows a typical infrastructure diagram for cloud storage. Examples of cloud storage providers include Dropbox [5], Amazon S3 [3], and Google Drive [6].

As a type of cloud service model, cloud storage inherits the benefits of cloud computing [33, 55, 98] and is argued by many people to offer a range of advantages over traditional owned storage systems. Those potential benefits include:

- Cost savings. Using cloud storage, a company does not need to undertake the capital

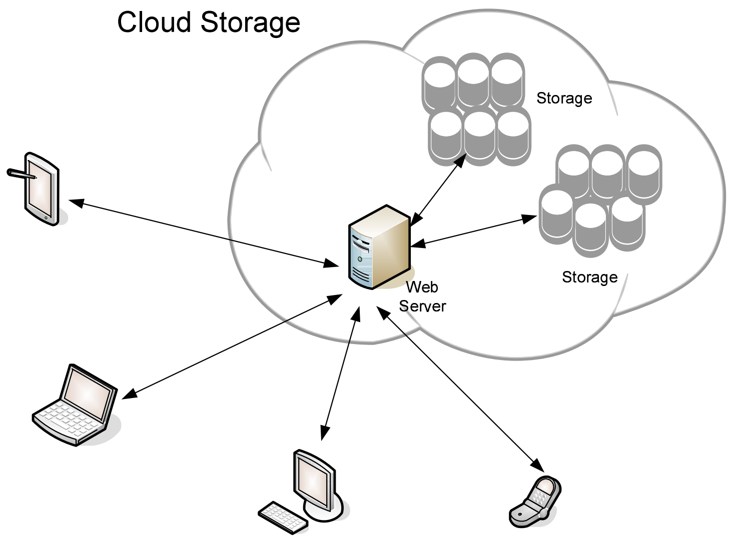


Figure 2.2: Cloud Storage

expense of purchasing or maintaining storage equipment or physical servers.

- Mobility and Accessibility. Cloud storage allows consumers to access data via the Internet, using any device, independent of current location.

- Flexibility. Cloud storage is capable of adapting to on-demand storage capacity changes by provisioning and deprovisioning resources quickly. Consumers only pay for the server space they use, rather than the space they might eventually need.

- Reliability and Availability. Virtualization technologies enable cloud storage to potentially supply unlimited storage space to users. Since data can be duplicated and backed up to cloud storage, when one server goes down, other servers can immediately be put into service. In the event of catastrophic data loss, data can be rapidly restored.

- Agility. Cloud storage enables rapid development, deployment, and change management.

- Collaboration and Sharing. Cloud storage services facilitate simpler data sharing between companies and organizations.

While the benefits of cloud storage are compelling, cloud storage does have its potential downsides and risks, as follows [1]:

- Cost. Cloud storage services may have a specific bandwidth allowance. If an organization surpasses the given allowance, the additional charges could be significant.
- Mobility and Accessibility. Reliance on Internet connectivity at the customer's end might create a single point of failure.
- Insecure or incomplete data deletion. Data duplication and redundancy provide reliability and availability, but can also result in insecure or incomplete data deletion.
- Lock-in. Cloud storage standards are not yet fully established, particularly with regard to portability of data, applications, and services. Consumers thus face difficulties if they migrate from one storage provider to another.
- Data protection and security. As data is outsourced to a third party storage system, data owners lose physical control of their data. Data security is thus a big concern when using cloud storage.

In this thesis, we focus on the last of these issues by extending data protection mechanisms suitable for cloud storage. In particular we consider data protection and access control in untrusted cloud storage environments.

2.5.1 Cloud Storage Security Challenges

In addition to the discussed security issues of distributed storage systems, cloud storage has unique challenges [41, 111, 131, 55] that arise from two major aspects: trust in the cloud storage providers (CSPs) and the shared storage environments (multi-tenancy). Those unique security challenges have special requirements for data protection and access control.

2.5.1.1 Trust in CSPs.

In traditional storage systems, storage servers are trusted to protect the data stored on the servers. This kind of trust does not necessarily exist in a cloud storage environment. After data is outsourced to cloud storage, data owners lose physical control of it. Data confidentiality and integrity are thus at increased risk.

Although sensitive data can be encrypted before it is sent to cloud storage, key distribution and management become more complicated. The mobility and accessibility of cloud storage enable users to access data at any time from anywhere in the cloud. Any adopted encryption mechanism must be suitable for dynamic environments and users. As the CSPs may be untrusted, data owners need to take responsibility for key distribution and management. This approach presents a number of performance and scalability issues, especially when a potentially large number of users desire fine-grained access to the data.

As encryption has key distribution and management overhead, it is undesirable for non-sensitive data. Data protection for non-sensitive data should be more focused on access control, and unauthorized modification or corruption. Although data integrity can be achieved by the mechanisms mentioned in Section 2.4.3, verifying correctness of data in the cloud may be challenging because the original copy of the data may no longer be held by the data owners.

User privacy or personal information is a further issue of concern [12]. Traditionally storage servers are trusted to mediate user access to stored data. User access management is typically via user account management, which can store or link to user personal information. This personal information can include a range of data items, such as user or account names, email addresses, phone numbers, dates of birth, social security numbers, etc., which are combined to form *Personally Identifiable Information* (PII) data [92]. PII data is treated as sensitive data, which can be used or linked to an individual's identity. As CSPs may not be fully trusted, relying on a CSP's administration of user accounts, or sharing/storing user PII data at CSPs, is problematic.

2.5.1.2 Multi-Tenant Environments

In traditional storage systems, the data stored on the servers is typically from the same security domain or shared by users in the same organization. In cloud storage, virtualization (enabling multi-tenancy) allows data from different security domains to be stored on the same physical server, or a single data owner's data to be stored on multiple servers, which can reside in different security domains. The risk of unauthorized data access to other user's data in such an environment is a big concern [39].

In multi-tenant storage environments, the hypervisor of the virtualization may have privileged access to the hardware's physical resources. In the event that a hypervisor is compromised, it may be possible to intercept the contents of memory, virtual network traffic, and other forms of communication that occur from other virtualization environments hosting different applications on the same physical server [55]. This can lead to user privacy disclosure of other applications, as well as information about their data access patterns. Data access patterns can be used by malicious insiders or external attackers in order to gain knowledge of the possible importance of files, which can then be either damaged or manipulated in order to achieve malicious goals, such as denial-of-service attacks (DoS).

2.5.2 Requirements for Protecting Data Secrecy and User Privacy in Cloud Storage

Motivated by the special needs of data protection and access control in a cloud storage environment, in this thesis we study effective and efficient access control mechanisms to protect data secrecy and user privacy. We assume that the cloud storage trust model consists of at least three parties in general:

- Cloud Storage Provider (CSP). These CSPs provide cloud storage services to users. They are not fully trusted. They are assumed to faithfully perform the required tasks, but are curious to learn the secrecy of data or user identities.

- Users. Users access the data stored on cloud storage. They are untrusted.
- Data owners (DO). Data owners store data in cloud storage. They are responsible for protecting the data.

Due to the trustworthiness of cloud storage systems, data needs to be capable of "self-protection". We refer to this as *data-centric* access control. In data-centric access control, sensitive data is encrypted before it is sent to cloud storage. Thus data-centric access control requires the following characteristics:

- *Fine-grained protection.* The data access policy is defined at a data-item level. The access policy should be enforced at each access attempt, with or without the data owner's involvement.
- *Dynamic access rights management.* The granting or revoking of user access rights to a particular data item should be straightforward to conduct and should, ideally, be performed almost instantaneously.
- *Efficient key management.* Critical key management operations such as key distribution, key revocation, and key refreshing should be conducted in an efficient manner that scales well and is appropriate for the highly dynamic and heterogeneous nature of a cloud storage environment.

User access control and management in untrusted cloud storage should consider user privacy protection. We refer to this as the *user-centric* access control. We argue that it requires the following characteristics:

- *Anonymous access control.* Access control mechanisms should authenticate users anonymously.
- *Dynamic user management.* User access granting or revocation should be anonymous, dynamic, and instantaneous to CSPs. CSPs should not need to know any user's actual identification information (PII) or administrate any user information that can be directly or indirectly linked to the user's true identity.
- *Fine-grained policy.* Access policies should be fine-grained and expressive. They should not contain or be able to hint at any user's identification information and should be enforced by CSPs.

As data is accessed by users, a data access control mechanism in an untrusted cloud storage should be concerned with both the requirements of data and user-centric controls.

2.6 Existing Access Control and Data Protection Techniques

Existing access control mechanisms consist of a large set of technologies. Depending on the trustworthiness of a server, there are two general approaches: *server-mediated* and *cryptographically-enforced* access control.

To understand how those approaches work in an untrusted cloud storage environment, let us first take a look at the following scenario:

Suppose a small global company XYZ wants to use cloud storage for storing their HR data. Their HR personnel from different offices worldwide can access it through the Internet. Depending on the responsibility of the HR personnel, the access to the data may need to be controlled. Company *XYZ is concerned about whether its data will be properly protected or not. It is also concerned with the protection of employee identities and PII data during the account creation and authentication process.*

2.6.1 Server-Mediated Access Control

Server-mediated access control is the traditional approach [123], which is mainly based on the reference monitor concept [11]. It requires the servers who mediate access to be fully trusted. Server-mediated access control typically includes traditional authentication and authorization mechanisms, in which users establish a trusted server to store data in the clear and then delegates that server to perform authentication and authorization checking on every access request. The most commonly used authentication methods include username and password as well as certificate-based techniques that normally use *Public key Infrastructure* (PKI) [8]. Username and password authentication relies on identity management systems to maintain and validate user credentials and their associated data, which can include PII data. Although certificate-based authentication does not depend on an identity management system, both user certificates and certificates of trusted authorities need to be managed and distributed. In addition, user certificates can contain user identification information.

Traditional authorization models utilize user capabilities, roles, or resource access control lists (ACLs) for granting access rights. The access policies define the rules by which access control must be regulated. Policies can be grouped into classes, such as [118]:

- *Discretionary* (DAC): the policies enforce the rules that define who is, or is not, authorized to carry out which actions on which resources.
- *Mandatory* (MAC): the policies control access based on mandated regulations determined by a central authority.
- *Role-based* (RBAC): the policies control access depending on the roles that users have within the system, and on rules stating what access is allowed to users in given roles.

While server-mediated access control is an acceptable way to protect data access, it requires company XYZ to fully trust CSPs. For untrusted CSPs, company XYZ may have the following concerns:

1. Data privacy. If the data is not encrypted, XYZ's HR data can be accessed by anyone who can access the server. While this may not be an issue if the stored data is public information, it is a concern if the data contains sensitive information, such as PII information.
2. Access policy enforcement. Server-mediated access control is only implemented at the

server level instead of the data item level. The data access policies may not be enforced when XYZ's HR data is transferred across the network or is temporarily cached on an intermediate server.

3. User privacy. The server needs to manage user accounts and access, which could contain or use PII data. If XYZ does not fully trusted the server, the company may want to protect their employees' privacy with anonymous access control.

Due to the above concerns, sever-mediated access control does not naturally fit in an untrusted cloud storage environment.

2.6.2 Cryptographically-Enforced Access Control

An alternative approach is to use cryptographically-enforced access control [56, 65, 97], which is access control provided through cryptographic "sealing". A data owner encrypts data using an encryption technology. The decryption key is controlled by an access policy or a trusted party to ensure that only appropriate parties can obtain it. Cryptographically-enforced access control allows users to store their data on an untrusted server.

Using cryptographically-enforced access control, company XYZ can first encrypt HR data with a key and map an access policy to the decryption key. After that, the encrypted HR data is stored on cloud storage. Since the data is encrypted, only users who are given the decryption key can decrypt and access the data. The data is self-protected in the sense that the access policy is also bound to the data. This is important since data could be stored and replicated anywhere on the servers within the cloud storage environment. In some situations, this could mean that the same data resides on servers in different countries.

There are two types of data encryption approach: symmetric encryption and asymmetric encryption. We now discuss how these encryption approaches can be used by company XYZ to enforce its access policies. We will also analyze the limitations that motivate our research objectives.

2.6.2.1 Access Control with Symmetric-Key Encryption

Symmetric encryption, such as [10, 43], uses the same key to encrypt and decrypt data. It is also known as *secret-key* encryption since the encryption key must be kept secret and only shared amongst the legitimate users. Company XYZ can use symmetric encryption to protect their data and enforce access control policies through key distribution. Two possible, but flawed, approaches are:

1. The company can generate one secret key and encrypt all their HR records with the same key. It only needs to distribute the key privately to the employees who need to access the data. The limitation of this approach is that the data sharing is "all-or-nothing." The company does not have the flexibility to choose a fine-grained access policy. For example, a salary record can only be viewed by managers and above. Another drawback is that the company has to distribute the same secret key to all intended users. If only one user

is compromised with the distributed key, all the HR records are potentially exposed.

2. Alternatively the company can generate a different set of keys, with each key encrypting a set of records. Different keys are thus shared among different groups of users. Although this approach overcomes the all-or-nothing problem, the key management becomes complicated. For example, if the company wants to allow n record sets with different access rights, then it has to create and securely distribute n keys. In addition, if a data item is shared among all of them, it has to be encrypted n times.

In both approaches, key distribution and access control need to be performed by data owners. This means that data owners have to be online all the time to mediate access to the data. This is not feasible and scalable in a highly distributed environment. Thus symmetric encryption does not exactly meet the requirements of data-centric protection.

Encryption has performance and key distribution overhead. For some internal data such as company XYZ's general announcements that are not sensitive and only used by internal employees, it is more efficient to store these unencrypted. Authentication can be used to control who can access the data. Symmetric encryption can also be used for user authentication. Using *Message Authentication Code* (MAC) [21], a unique secret key is shared between users and an authentication server. A user is authenticated by sending to the authentication server a randomly generated message (the challenge) encrypted by the secret key. If the server can match the received encrypted message (the response) using its shared secret key, the user is authenticated.

Although company XYZ can use symmetric encryption authentication for access control of insensitive data, it is unfeasible to share a secret with untrusted CSPs.

2.6.2.2 Access Control with Asymmetric Cryptography

Asymmetric encryption [9, 46], also referred to as *public-key encryption* (PKE), uses a pair of public and private keys. The public key is public, and mathematically related to its private key that is kept secretly. Data is encrypted under the public key and decrypted by the private key.

Public-key encryption can also be used to support user authentication using digital signatures [112]. In a digital signature scheme, a user is required to sign a piece of information using a private signing key. Only the public verification key associated with the private signing key is able to validate the signature. This implies that the signature is genuinely generated by the user. Digital signatures can also validate the integrity of data, but they do not protect the secrecy of data.

Using PKE, company XYZ can protect its sensitive HR data as follows:

- A manager first generates a pair of keys. The manager uses the public key to encrypt all the HR data and stores it on the storage server. When someone needs to access the data, the manager will download and decrypt the data, then re-encrypt the records using the employee's public key based on the HR access policy. The issue with this approach is that the manager has to remain on-line to mediate the requests.

- Another way to do this is for the manager to encrypt the records using the employees' public keys based on their HR access control policies. However, the manager needs to obtain all the employees' public keys ahead of time in order to encrypt the data. This is difficult to manage because employees can move on to other groups or new employees can join. Also, PKE tends to be a one-to-one relationship, meaning that one key pair is owned by one user. If a record is shared by n users, the record will need to be encrypted n times.

For protecting its internal data, company XYZ can implement user authentication as follows:

- Every employee is issued with a digital signature key pair. The company sends all the public verification keys to the CSP. At the time when the CSP authenticates an employee, the CSP randomly generates a piece of information for the employee to sign. The CSP validates the signature with the employee's public key.

As we can see, there are two main benefits of using asymmetric cryptography over symmetric cryptography. Firstly the public key is readily available to the public. In fact, public keys are often published in public directories on the Internet so that they can be easily retrieved. This simplifies key distribution efforts. Secondly it enables user authentication with digital signatures, which does not require a secret to be shared with CSPs. Moreover, the verification keys are also public keys and publicly available. However, the issues with using public-key cryptography are as follows:

- *Encryption.* Data has to be encrypted multiple times using all the public keys of each user who needs to share the data. The data owner has to know who will require access to the data ahead of time. Multiple encryption also introduces a large overhead if the data needs to be shared among large group of users. Furthermore, since the access control policy (regarding who can decrypt data) is separated from the ciphertext, it cannot be directly enforced by CSPs.

- *Authentication.* Although digital signatures are a viable authentication method for cloud storage, they still require CSPs to keep all users' public keys. As each public key can be linked to an employee's identity, this may still be an issue when it comes to protecting employees' privacy.

2.6.2.3 Attribute-Based Encryption and Signatures

The notion of *Attribute-Based Encryption* (ABE) was introduced by Sahai and Waters [116] in 2005. It is a type of public-key encryption, which has a one-to-many relationship between a public key and a set of decryption keys. ABE is a generalization of *Identity-Based Encryption* (IBE) [25], in which data is encrypted by a public key and a set of system parameters. The decryption keys or decryption process is associated with an access policy. A user can encrypt data addressed to a group of users that fit a specific set of requirements without knowing each individual user in the group ahead of time. Users who have the attributes satisfying the access policy can decrypt the data. All the ciphertexts or decryption keys are associated with fine-grained access policies that are automatically enforced in the decryption process.

When using ABE to protect sensitive data, company XYZ can use one public key to encrypt the data just once. It then defines different access policies so that different users are granted access to different sets of HR records. Since the access policy can be automatically enforced during the decryption process, it does not need data owners or a third party, such as a CSP, to mediate the data access. Furthermore, when data is backed up on different servers in the cloud, the access policy is also transferred and enforced. ABE seems to be the ideal choice for data-centric protection in a cloud environment. However, it does come with the following practical limitations on user management:

- *Dynamic user revocation.* An access policy in ABE defines a group of users who share the same set of attributes. When an employee leaves the company or switches to another department, the employee may no longer be entitled to access the data. But the employee may still hold a valid private key to decrypt the data. In order to revoke the employee's rights, company XYZ will have to re-generate the public key and re-issue new private keys to other employees in the same group (or groups). Also the data will need to be decrypted and re-encrypted under the new key.
- *Dynamic user attribute revocation.* A user can have their status changed over time. Those changes can invalidate one or multiple attributes of the user. Attribute revocation can be more granular than user revocation, because a user may still hold other attributes that can satisfy other access policies. Any revoked attribute from a user could be valid for other users. The revocation should not impact the attribute usage of other users. Like user revocation, a user's attribute revocation requires key re-generation and data

re-encryption can significant impact the usability and scalability of ABE.

Inspired by ABE, *Attribute-Based Signatures* (ABS) schemes [90] enable anonymous authentication and signature generation. In ABS a set of system attributes are used to define predicates in a similar way to the access policy in ABE. A public key is used to validate signers who have the attributes complying to a predicate. To authenticate a user, a user uses his private key to sign a piece of data according to the predicate. The signature can only be validated if the user has attributes complying with the predicate. CSPs only need the public key and associated predicate to authenticate users with no need of knowing individual user identity or certificate information.

Using ABS, company XYZ first generates a public key and system parameters. Every employee is issued with a signing key based on their attributes (or roles). Company XYZ then defines a predicate that is expressed by attributes, such as $\{\text{employee} \wedge \text{HR Department}\}$. The CSP is then given the public key and predicate. At the time when a user requests data access, the CSP randomly generates a message (challenge) and asks the user to sign it (response). The user generates the signature based on the predicate using his signing key. The CSP validates the signature only if the user has the attributes conforming to the predicate. In this way the CSP only knows that the user is one of the employees from the HR Department. The CSP cannot link the signature to any individual user; even if the same user generates the signatures on the same message multiple times. ABS seems to meet the requirements of user-centric protection.

However, like ABE, ABS also suffers from the issue of dynamic user revocation, especially anonymous user revocation. In ABS, a signature attests that a user belongs to a group who share the same subset of attributes according to a predicate. Although each user is issued with his own signing key, those individual signing keys are all derived from the same master secret. This kind of setting makes a user anonymous in the process of signature verification, but leads to issues with revoking a user from the group since the revoked user can still generate valid signatures that can be used for authentication. Therefore the signing keys of non-revoked users need to be re-generated.

2.7 Conclusions

ABE and ABS meet the requirements of data and user centric protection in untrusted cloud storage systems. However, they have practical limitations when directly applied. In this chapter we have identified these limitations. In the following chapters, we will propose deployment models and systems designed to overcome the user and attribute revocation limitations of ABE and ABS schemes. In doing so, we make them more practical for deployment in untrusted cloud storage environments.

Chapter 3

Cryptographic Preliminaries and Background

Contents

In this chapter we provide a brief background of the cryptography necessary for the the remainder of the thesis. We start with a brief introduction to the necessary abstract algebra. We then touch on the necessary concepts of modern cryptography and related complexity assumptions upon which our schemes are analyzed. We then describe provable security and relevant security models.

3.1 Abstract Algebra

A *group* $\mathbb{G}$, sometimes denoted by $\{\mathbb{G}, \star\}$, is a set of elements with a binary operation denoted by $\star$. A group $\mathbb{G}$ satisfies the four group axioms: closure, associativity, the identity property, and the inverse [124]. When the group $\mathbb{G}$ has a finite set of elements, $\mathbb{G}$ is called a *finite group.* The number of elements in $\mathbb{G}$ is called the *order* of the group. A *cyclic group* is a group that can be generated from a single element of the group, $g \in \mathbb{G}$. Assuming the binary operation is multiplication, then $\mathbb{G} = \langle g \rangle = \{g^i \mid i \in \mathbb{Z}\}$, which implies that for any $y \in \mathbb{G}$ there exists an integer i such that $g^i = y$. Given a non-empty subset H of the group $\mathbb{G}$ defined under binary operation $(\star)$, H is a *subgroup* of $\mathbb{G}$, if H is also a group under $(\star)$.

Let $\mathbb{Z}_n$ be the set of integers $\{0, 1, \ldots, n - 1\}$, where n is a positive integer. If the operation $+$ in $\mathbb{Z}_n$ is addition modulo n, then $\{\mathbb{Z}_n, +\}$ is a group and n is the *order* of the group. If the operation $\times$ in $\mathbb{Z}_n$ is multiplication modulo n, then $\{\mathbb{Z}_n, \times\}$ is not a group, since not every element in $\mathbb{Z}_n$ has a multiplicative inverse. But if n is a prime and $\mathbb{Z}_n^\star = \mathbb{Z}_n \setminus \{0\}$, then $\{\mathbb{Z}_n^\star, \times\}$ is a group [124].

A *field* $\mathbb{F}$, sometimes denoted $\{\mathbb{F}, +, \times\}$, is a set of elements with two binary operations, *addition* and *multiplication.* A field satisfies the following axioms: associativity, commutativity, distributivity, the identity property, and the inverse property [124]. The number of elements in a field $\mathbb{F}$ is called the *order* of the field. If the order is finite, the field is called a *finite field*. Examples of fields include the real numbers $\mathbb{R}$ and the complex numbers $\mathbb{C}$.

3.2 Modern Cryptography

The goal of modern cryptography is to design cryptographic schemes (e.g. encryption, message authentication, etc.) with security properties that can be proven based on the robustness of computational assumptions, such as the assumption of factorization, or discrete logarithm [77].

Modern cryptography evolved from *classical* cryptography [57], which is viewed as an *art* with the focus of designing and using *codes* (also called ciphers) to enable two parties to communicate secretly in the presence of an eavesdropper who can monitor all communication between them. The security of classical encryption schemes relies on a secret (or a secret key) that is shared by the communicating parties in advance and unknown to the eavesdropper. There was no systematic way of defining what constituted a good code or proving it to be secure. In the late 1970s to early 1980s, a rich set of theories began to emerge enabling a more rigorous study of cryptography as a *science* and a mathematical discipline. This perspective led to modern cryptography.

Modern cryptography is concerned with the rigorous analysis of a cryptographic system that should resist malicious attempts to abuse it. It is the scientific study of techniques for securing digital information, transactions, and distributed computations. This scientific approach distinguishes modern cryptography from classical cryptography. It made cryptography go from a heuristic set of tools to ensure secret communication for the military to a science that helps secure systems for ordinary people. In addition, modern cryptography has a much broader scope. It has been extended from symmetric key encryption and data secrecy protection to also include public-key cryptography, data integrity protection, and non-repudiation verification.

3.2.1 Principles of Modern Cryptography

Modern cryptography follows three principles: formal definitions, precise assumptions, and proofs of security [77].

3.2.1.1 Formal Definitions

A security definition defines what a scheme can possibly achieve or what the security goal of the scheme is. The importance of security definitions is to define security guarantees for a scheme; provide guidance on design, construction, and usage of cryptographic primitives; and enable a way to evaluate and analyze the security of the scheme. A security definition consists of two components: a threat model and security guarantees.

The threat model describes the power of an adversary, or the possible attacks from the adversary, but does not place any restriction on the adversary's strategy. In other words, the threat model only assumes an adversary's abilities but does not assume anything as to how it uses those abilities. For example, a threat model can assume the following attacks in the context of encryption [77]:

- *Ciphertext-only attack*: refers to a scenario where the adversary just observes one or multiple ciphertexts in an attempt to determine information about the underlying plaintext (or plaintexts).
- *Known-plaintext attack*: refers to an adversary being able to learn one or more plaintext/ciphertext pairs generated by an encryption key. The adversary attempts to deduce information about the underlying plaintext of some other ciphertexts encrypted by the same key.
- *Chosen-plaintext attack*: is a similar attack to a known-plaintext, except that an adversary can obtain plaintext/ciphertext pairs for the plaintexts of the adversary's choice.
- *Chosen-ciphertext attack*: refers to an adversary having the additional ability to obtain information about the decryption of ciphertexts of the adversary's choice, e.g. whether the decryption of some ciphertexts chosen by the adversary yields a valid message or not.

Choosing the right threat model to use depends on the environment in which an encryption scheme is deployed.

The security guarantee defines what the scheme intends to prevent an attacker from doing or what constitutes a successful attack on the scheme. Based on Kerckhoffs's principle, a cryptographic system should be secure even if everything about the system, except the key, is public knowledge. For example, when we define the security guarantee of an encryption scheme, the goal of the scheme might be the following:

- An attacker should not be able to recover the decryption key;
- The ciphertext should make it impossible for an attacker to recover the plaintext entirely or partially, regardless of any information the attacker already has.

Semantic security [58] and ciphertext indistinguishability [99] are commonly used definitions for secure encryption. These are essentially equivalent, and formal definitions will be given in Section 3.2.2.

3.2.1.2 Assumptions

Modern cryptography relies on computation theory, mathematical modeling, and proofs to argue that a cryptosystem or construction is secure. Certain assumptions need to be introduced to establish our security arguments. Those assumptions must be made explicit and mathematically precise. Examples of the assumptions include how powerful the adversary is in terms of run time or computer resources and what the mathematical theory or known assumptions are.

3.2.1.3 Security Proofs

Security proofs provide evidence as to whether proposed constructions are secure, based on their security definitions and assumptions. For example, either no attacker will succeed or one will have a certain probability of breaking the system. Although security proofs in modern cryptography are much more formal than the heuristic approach in classical cryptography, proofs are still relative to their definitions and assumptions. If a security guarantee does not match the security definition, or the threat model does not capture the adversary's true abilities, then the proof may be irrelevant. Similarly, if an assumption turns out to be false, then the proof of security is meaningless.

3.2.2 Computational Security

Computational security is used by modern cryptography to prove the security of a cryptographic scheme. It uses computation theory to argue how feasibly a given adversary can break the scheme. Computational security allows cryptographic schemes to limit the computational power of attackers and achieve *a small* probability of failure.

Computational security is a weaker notion than the information-theoretic security introduced by Shannon [122]. Information-theoretic security requires that no information about an encrypted message be leaked, even to an eavesdropper with unlimited computational power. Information-theoretic security is considered to be the ideal security for a cryptosystem. However, this kind of security is also considered to be too difficulty to achieve in practice [77].

There are two types of encryption systems. In *symmetric encryption*, the encryption and decryption keys are the same. *Public-key encryption* has distinct encryption and decryption keys, with the encryption keys (public keys) being publicly known, and the decryption keys being kept secret. We will proceed by explaining symmetric encryption schemes, and will define public-key encryption in Section 3.4.

3.2.2.1 Perfect Indistinguishability

To better understand computational security, we will explain what we mean by *a small* probability of failure with bounded computational power of an attacker. The security of *perfect indistinguishability* [77] is defined as follows:

Definition 3.2.1. A symmetric key encryption scheme is defined by three algorithms *Gen*, *Enc*, and *Dec*, with a specification of a (finite) message space M where the length of M is larger than 1: $|M| > 1$:

- *Gen:* A probabilistic key generation algorithm. We denote K as the key space (typically a finite set), the set of all possible keys k is chosen uniformly from K, and output by *Gen* $\rightarrow k \in K$.
- *Enc:* Taking a key k and a message $m \in M$ as inputs, it outputs a ciphertext c. We denote $Enc_k(m)$ the encryption of the plaintext m using the key k. If *Enc* is probabilistic algorithm, $Enc_k(m)$ might output a different ciphertext c each time it runs. We denote $Enc_k(m) \rightarrow c$ to indicate the possibly probabilistic process by which message m is encrypted using k to output c. Let $\mathbf{C}$ denote the set of all possible ciphertexts that can be output by $Enc_k(m)$, for all possible choices of $k \in K$ and $m \in M$. Here K and M are assumed to be independent. The selection of k is completely independent from message selection m. Note: in case *Enc* is deterministic, we emphasize this by writing $c := Enc_k(m)$.

- *Dec:* Taking a key k and a ciphertext c as inputs, it outputs a plaintext m. We denote the decryption of the ciphertext c using the key k by $Dec_k(c)$. We assume perfect correctness, meaning that for every key k output by Gen and every message $m \in M$, it holds that $Dec_k(Enc_k(m)) := m$ with probability 1.

Let $\Pi = (Gen, Enc, Dec)$ be an encryption scheme, and A be an adversary (eavesdropper) with message space M and $|M| > 1$. We define the randomized experiment $PrivK^{eva}_{A,\Pi}$ as follows:

1. The adversary A outputs a pair of messages $m_0, m_1 \in M$.
2. A key k is generated using Gen, and a uniform bit $b \in \{1, 0\}$ is chosen. The ciphertext $Enc_k(m_b) \rightarrow c$ is computed and given to A. We refer to c as the challenge ciphertext.
3. A outputs a bit $b^{'}$.
4. The output of the experiment is defined to be 1 if $b^{'} = b$, and 0 otherwise. We write $PrivK_{A,\Pi} = 1$ if the output of the experiment is 1 and we say that A succeeds.

Definition 3.2.2. (Perfect Indistinguishability) Encryption scheme $\Pi = (Gen, Enc, Dec)$ with message space M is perfectly indistinguishable if for every A it holds that:

$$\Pr\left[PrivK^{eva}_{A,\Pi} = 1\right] = \frac{1}{2} .$$

3.2.2.2 Computational Indistinguishability

As perfect indistinguishability is not considered to be practical, computational indistinguishability relaxes it in two aspects: allowing *a small* probability of security failure, and only considering *efficient* adversaries (attackers). Two general approaches are used to define those relaxations: the *concrete* approach and *asymptotic* approach.

The concrete approach [17] takes the following form of definition:

Definition 3.2.3. *A scheme is (t, ε)-secure if any adversary running for time at most t succeeds in breaking the scheme with probability at most ε.*

Although a concrete approach can provide concrete guarantees of a cryptographic scheme in terms of exact running time t and possible breaking probability ε, it is arguably difficult to achieve those guarantees [77, 114]. For example, it may be necessary to precisely specify what computational power an adversary has (i.e. the type of computer, or the type of operating system, and etc.); and whether or not to consider future advances in computing power and technology.

Instead of binding any particular computational mode to an adversary, an asymptotic approach introduces an integer-valued security parameter (denoted by n) that parameterizes both cryptographic schemes as well as all involved parties, including both honest parties and adversaries. One of the important aspects of using the security parameter n is that the security level of a scheme can be adjusted, i.e., n is set to different values. Also, n can be viewed as the key length shared by parties, including adversaries. With this security parameter n, the security failure is defined as the probability in n, and an adversary's running time is in a polynomial form of n. As efficient algorithms run in polynomial time in complexity theory, an adversary is referred as an efficient one if its execution time is bounded by polynomial time.

Definition 3.2.4. *A polynomial-time algorithm* is an algorithm whose worst-case running time function is of the form $O(n^k)$, where n is the input size and k is a constant.

Definition 3.2.5. A probabilistic algorithm A is a *probabilistic polynomial time* (PPT*)* algorithm if the running time of $A(x)$ is bounded by $p(|x|)$ where p is a polynomial. The running time is measured by the number of steps in the model algorithm (i.e. the number of steps in a probabilistic Turing machine). Tossing a coin is one step in this model.

Definition 3.2.6. A function f is *negligible*, if for every polynomial p, there is an N such that for all integers $n > N$ it holds that f$(n) < \frac{1}{p(n)}$.

Definition 3.2.7. (Redefining Definition 3.2.1 using Asymptotic Approach) A symmetric key encryption scheme is a tuple of PPT algorithms (*Gen*, *Enc*, *Dec*) such that:

- $Gen(1^n)$: takes as input 1^n (i.e. the security parameter written in unary), outputs a random key $Gen(1^n) \rightarrow k$. Without loss of generality, we assume the length of k is $|k| > n$.
- *Enc*(k, m): takes as input a key k and a plaintext message $m \in \{0,\ 1\}^\star$, and outputs a ciphertext c. As *Enc* may be randomized, we denote $Enc_k(m) \rightarrow c$.

- *Dec*$(k,\ c)$: takes as input a key k and a ciphertext c, and outputs a message m or an error. *Dec* is assumed to be deterministic: $m := Dec_k(c)$ (without error), or $\perp$ when there is an error.

Let $\Pi = (Gen,\ Enc,\ Dec)$ be a scheme defined as above with a security parameter n. An adversary (eavesdropper) A's indistinguishability experiment $PrivK^{eva}_{A,\Pi}(n)$ is defined as follows:

1. A is given an input 1^n, and outputs a pair of messages m_0, m_1 with message lengths: $|m_0|$ $= |m_1|$.
2. A key k is generated by running $Gen(1^n)$, and a uniform bit $b \in \{0,\ 1\}$ is chosen. Ciphertext c $\leftarrow Enc(k,\ m_b)$ is computed and given to A. We refer to c as the challenge ciphertext.
3. A outputs a bit $b^{'}$.
4. The output of the experiment is defined to be 1 if $b^{'} = b$, and 0 otherwise. If $PrivK^{eva}_{A,\Pi}$ $(n) = 1$, we say that A succeeds.

Definition 3.2.8. A private-key encryption scheme $\Pi = ($*Gen, Enc, Dec*$)$ has indistinguishable encryptions in the presence of an adversary (eavesdropper), or is EAV-secure (eavesdropping secure), if for all probabilistic polynomial-time adversaries A there is a negligible function *negl* such that, for all n,

$\Pr\left[PrivK^{eva}_{A,\Pi}(n) = 1\right] \leq \frac{1}{2} + negl(n),$

where the probability is taken over the randomness used by A and the randomness used in the experiment (for choosing the key and the bit b, as well as any randomness used by *Enc* algorithm).

We also use ϵ to denote the output of *negl(n).* The above definition requires that no PPT adversary can determine which of two messages was encrypted, with probability significantly better than $\frac{1}{2}$. This is equivalent to a definition called *semantic security* [58, 59].

Definition 3.2.9. A private-key encryption scheme (*Gen, Enc*, *Dec*) is semantically secure in the presence of an eavesdropper if for every PPT algorithm A there exists a PPT algorithm $A^{'}$ such that for any PPT algorithm *Samp* and polynomial-time computable functions f and h, the following is negligible:

$$|Pr\left[A(1^n,\ Enc_k(m),\ h(m))\ =\ f(m)\right]| \text{ - } \left|Pr\left[(A^{'}(1^n,\ |m|\,,\ h(m))\ =\ f(m)\right]\right|,$$

where the first probability is taken over uniform $k \in \{0,1\}^n$, m output by $Samp(1^n)$, the randomness of A, and the randomness of *Enc* while the second probability is taken over m output by $Samp(1^n)$ and the randomness of $A^{'}$.

3.3 Complexity Assumptions

In computational complexity theory, a *reduction* is an algorithm for transforming one problem into another problem [128]. A reduction from one problem to another may be used to show that the second problem is as difficult as the first. For example, problem A is *reducible* to problem

B if an algorithm for solving problem B efficiently (if it exists in polynomial time) could also be used as a subroutine to solve problem A efficiently. When this is true, solving A cannot be harder than solving B. We denote this as $A \leqslant_m B$.

Reduction is very useful to prove that a cryptographic scheme is secure. That is, if we can prove that breaking a scheme means finding a solution to an assumed intractable problem, then it is also intractable to break the scheme. We refer to the assumed intractable problem as a *complexity assumption*. In the following sections we define complexity assumptions that will be used for analyzing the security of the systems in this thesis.

3.3.1 Diffie-Hellman Assumptions

The *Diffie-Hellman assumptions* are closely related to the difficulty of computing the discrete logarithm problem over a cyclic group [93].

Let $\mathcal{IG}$ be a polynomial-time algorithm that takes security parameter 1^n as input and outputs the tuple $\langle G,\ p,\ g\rangle$, where G is cyclic group, p is the order of G, and g is a generator of G.

Definition 3.3.1. The *Discrete Logarithm Problem* (DLP) in $\langle G,\ p,\ g\rangle$ is defined as follows: given $(g,\ g^a)$, compute a, where a is randomly chosen from Z_p. A polynomial-time adversary A has advantage ϵ in solving the DLP in $\langle G,\ p,\ g\rangle$ if :

$\Pr[A(g,\ g^a) = a] \geq \epsilon$, where the probability is over the random choice of $a \in Z_p$ and the random bits used by A.

A group G satisfies the DLP if there is no such adversary A with non-negligible advantage.

Definition 3.3.2. The *Computational Diffie-Hellman Assumption* (CDH) [22] in $\langle G,\ p,\ g\rangle$ is defined as follows: given $(g,\ g^a,\ g^b)$, where $a,\ b$ are randomly selected from Z_p, compute g^{ab}. A polynomial-time adversary A has advantage ϵ in solving the CDH problem in $\langle G,\ p,\ g\rangle$ if $\Pr\big[A(g,\ g^a,\ g^b)\ =\ g^{ab}\big] \geq \epsilon$, where the probability is over the random choice of $a,\ b \in Z_p$ and the random bits used by A.

We say that a group G satisfies the CDH if there is no such adversary A with non-negligible advantage.

Definition 3.3.3. The *Decisional Diffie-Hellman Assumption* (DDH) [22] in $\langle G,\ p,\ g\rangle$ is defined as follows: given $(g,\ g^a,\ g^b,\ g^c)$, where $a,\ b,\ c$ are randomly selected from Z_p, determine whether $c = ab$. A polynomial-time adversary A has advantage ϵ in solving the DDH problem in $\langle G,\ p,\ g\rangle$ if $\big|Pr\left[A(g,\ g^a,\ g^b,\ g^{ab})\ =\ 1\right]\ -\ Pr\left[A(g,\ g^a,\ g^b,\ g^c)\ =\ 1\right]\big| \geq \epsilon$, where the probability is over the random choice of $a,\ b,\ c \in Z_p$, and the random bits used by A.

We say that a group G satisfies the DDH if there is no such adversary A with non-negligible advantage.

It is not difficult to see that the DLP, CDH, and DDH are related. If an algorithm can solve DLP, it can also resolve the CDH and DDH. An algorithm which solves the CDH problem can be used to solve the DDH problem. But if an algorithm can solve the DDH, it is unknown whether it can solve the CDH or not. The DDH is a very important assumption for proving security of cryptographic systems [22]. If an adversary can solve the DDH, it can break the indistinguishability property of a scheme.

3.3.2 The Bilinear Maps and Bilinear Diffie-Hellman Assumptions

Many cryptographic schemes are constructed using bilinear maps [73, 129], such as the CP-ABE scheme of [20]. Computational complexity assumptions for bilinear maps, such as Bilinear Diffie-Hellman assumptions, are thus used for arguing the security of the schemes. We will use bilinear maps to construct our security systems.

3.3.2.1 The Definition of Bilinear Maps

Let G_{1}, G_{2}, and G_T be multiplicative cyclic groups of prime order p. Let g_1 and g_2 be generators of G_1 and G_{2}. A mapping e: $G_1 \times G_2 \rightarrow G_T$ is said to be *bilinear* if it has the following properties [23]:

- *Bilinearity*: for all $u \in G_1$, $v \in G_2$, and $a,\ b \in Z_p$, $e(u^a,\ v^b) = e(u,\ v)^{ab}$
- *Non-degeneracy*: $e(g_1,\ g_2) \neq 1$

Throughout this thesis, we consider $G_1 = G_2$, or e: $G \times G \rightarrow G_T$.

The existence of a bilinear map e: $G \times G \rightarrow G_T$ has two direct implications for the groups involved:

- The MOV Reduction [95]: the DLP is no harder in G than it is in G_T. To see this, let $P, Q \in G$. We would like to find $a \in Z_p$, such that $Q = P^a$. Let $g = e(P, P)$ and $h = e(P, Q)$. Then $h = g^a$. Therefore we reduce the DLP in G to the DLP in G_T.

- The DDH is easy [74]: The DDH in G is to distinguish between the distributions *(g, g^a, g^b, g^{ab})* and *(g, g^a, g^b, g^c)*, where *$a, b, c \in Z_p$* are randomly selected. The DDH in G_T is easy. To see this, observe that: given $g, g^a, g^b, g^c \in G^\star$, we have $c = ab \bmod q \Longleftrightarrow e(g, g^c) = e(g^a, g^b)$.

3.3.2.2 Bilinear Diffie-Hellman Assumptions

The Bilinear Diffie-Hellman assumptions will be used for arguing the security of our proposed systems.

Definition 3.3.4. The *Bilinear Diffie-Hellman Assumption* (BDH) [24] in $\langle G , G_T , p , g , e\rangle$ is defined as follows: given g, g^a, g^b, g^c, where a, b, c are randomly selected from Z_p, compute $e(g, g)^{abc}$. A polynomial-time adversary A has advantage ϵ to solve the BDH problem in $\langle G, G_T, p, g, e\rangle$ if $\left|Pr\left[A(g, g^a, g^b, g^c) = e(g,g)^{abc}\right]\right| \geq \epsilon$,

where the probability is over the random choice of a, b, $c \in Z_p$ and the random bits used by A.

We say that the group G and G_T satisfy the BDH if there is no such adversary A with non-negligible advantage.

Definition 3.3.5. The *Decisional Bilinear Diffie-Hellman Assumption* (DBDH) [24] is defined as follows: given g, g^a, g^b, g^c, $e(g,\ g)^z$, where a, b, c, z are randomly selected from Z_p, determine if $z = abc$. A polynomial-time adversary A has advantage ϵ in solving the DBDH problem in $\langle G,\ G_T,\ p,\ g,\ e\rangle$ if $\left|Pr\left[A(g,\ g^a,\ g^b,\ ,g^c,\ e(g,g)^{abc}) = 1\right] - Pr\left[A(g,\ g^a,\ g^b,\ g^c,\ e(g,g)^z) = 1\right|\right. \geq \epsilon$,

where the probability is over the random choice of a, b, c, $z \in Z_p$ and the random bits used by A.

We say that the group G and G_T satisfy the DBDH if there is no such adversary A with non-negligible advantage.

3.4 Provable security

Provable security aims to provide a mathematical basis to argue whether a scheme is secure or not based on its definition. As modern cryptography is mainly based on the hardness of computational security, which is based on computational complexity, it allows a negligible possibility of security failures, but should be infeasible to break the security of the cryptosystem by any known practical means. How to properly model the power of adversaries and precisely specify assumptions become critical to security proofs.

3.4.1 Game-Based Security Models

Modern cryptography security definitions include a complexity-bounded adversary. To argue the possibility of an adversary being able to break the cryptosystem, we need to define a model to measure its complexity. The model specifies the type of an adversary and the adversary's power over a cryptosystem. The type of an adversary can be classified by whether the adversary is polynomial-bounded or not, how the adversary interacts with a security game or simulation system, etc. The adversary's power over a cryptosystem defines whether the adversary has access to a single ciphertext, multiple ciphertexts, multiple keys, etc. It also defines precisely what constitutes a break of the cryptosystem.

There are two general approaches to forming security models: game-based and simulation-based [45, 107]:

- In the *game-based* security model, an adversary is modeled by a security game. During the game an adversary interacts with a hypothetical probabilistic algorithm called a *challenger*. The challenger generates keys used in the system, and may respond to queries made by the adversary. The game terminates when the adversary terminates, and we assess whether the adversary has met the condition for breaking the cryptosystem. Typically a reduction (see Section 3.3) is used to prove that if there exists an adversary capable of breaking the security algorithm, then that adversary is also able to solve a computationally intractable problem - a complexity assumption that is believed to be hard.

- In the *simulation-based* model, the environment in which the cryptographic scheme will be used is simulated. In a simulated environment, an arbitrary PPT adversary can interact with the algorithms of the cryptosystem in an arbitrary PPT environment. This includes all the parties that may have access to the algorithms of the cryptosystem. An idealized version of the cryptosystem that can never be broken is produced. The model generally involves using an abstract third party who can always be trusted to transport and/or vouch for data and whose operation is outside of the view of both the environment and the adversary. The assumption is that if the outputs of the environment and adversary are almost the same when the idealized cryptosystem is used in place of the real cryptosystem, then the real cryptosystem should be secure. Hence, this implies that the cryptosystem is secure if the probability of being able to tell the difference between these two outputs is small.

In this thesis we use game-based models and assume PPT adversaries. As this thesis concerns attribute-based encryption, a type of public-key cryptosystem, we will introduce the concept of game-based models for public-key encryption. To better understand the security games, we start off with the definition of public-key encryption.

Definition 3.4.1. A public-key encryption scheme is a triple of probabilistic polynomial-time algorithms (Gen, Enc, Dec) such that:

- $Gen(1^n)$: takes the security parameter 1^n as input and outputs a pair of keys (pk, sk), where pk is the public key and sk is the private key. For convenience we assume that $|pk|$

$\geq n$, $|sk| \geq n$, and n can be determined from *pk*, *sk*.

- *Enc(pk, m)*: takes a public key *pk* and a message $m \in \{0,\ 1\}^*$ as inputs and outputs a ciphertext c: $Enc_{pk}(m) \rightarrow c$.
- *Dec(sk, c)*: takes a private key *sk* and a ciphertext c as inputs and outputs a message m or $\perp$ for failure: $m = Dec_{sk}(c)$.

CPA – Security [77]: CPA stands for *chosen plaintext attack*. It is also referred as *indistinguishability* CPA (IND-CPA). In this security game, an adversary can query the encryption oracle multiple times for the messages the adversary chooses. The encryption oracle is a "black box" to the adversary. The adversary is modeled by a PPT Turing machine, meaning that it must complete the game and output a guess within a polynomial number of steps. The CPA game for a public-key encryption algorithm is as follows:

Let a public-key encryption scheme $\Pi = (Gen,\ Enc,\ Dec)$ and an adversary be A, consider the following experiment:

- The challenger sets up the system and generates a key pair $(pk,\ sk)$ by running $Gen(1^n)$, where n is the security parameter of the system. The challenger publishes *pk* to the adversary, and keeps *sk* secret.
- A submits two distinctly chosen plaintexts m_0, m_1 of the same length ($|m_0| = |m_1|$) to the challenger.

- The challenger selects a bit $b \in \{\ 0,\ 1\}$ uniformly at random, and sends the ciphertext $Enc_{pk}(m_b) \rightarrow c_b$ back to A.
- The adversary is free to request a polynomially-bounded number of additional encryptions of messages other than m_0, m_1.
- Finally, A outputs a guess $b^{'}$ for the value of b. If $b^{'} = b$, then A wins the game.

Definition 3.4.2. A public-key encryption scheme $\Pi = (Gen,\ Enc,\ Dec)$ is CPA-secure if for all PPT adversary *A,* there exists a negligible function *negl* such that:

$$\Pr\big[PubK^{CPA}_{A,\Pi}(n)\ =\ 1\big] \leq \frac{1}{2} + negl(n).$$

CPA-Security is the basic requirement of any public-key encryption scheme [136]. The security guarantee is that learning anything about the plaintext from the ciphertext is almost infeasible for a PPT adversary. CPA-Security can resist two types of adversaries: a passive adversary that only eavesdrops, and an active adversary who carries out a chosen-plaintext attack. However, it falls short for more active attacks where an attacker tampers with a ciphertext in order to modify the plaintext.

There is a general consensus within the cryptographic research community that in virtually every practical application, we require security against *chosen ciphertext attacks* (*CCA-Security*) [109], whereby an adversary is given the access of decryption oracle to obtain the decryption of arbitrary ciphertexts of its choice. CCA-security provides security guarantees against powerful attackers who can alter encrypted messages. Therefore CCA-security has become the accepted goal for building secure encryption schemes in the real world.

CCA – Security: CCA-Security is also referred as *indistinguishability* CCA (IND-CCA) [77].

Let a public-key encryption scheme be $\Pi = (Gen, Enc, Dec)$ and an adversary be A, consider the following experiment:

- The challenger sets up the system and generates a key pair (pk, sk) by running $Gen(1^n)$, where n is the security parameter of the system. The challenger publishes pk to the adversary, and keeps sk secret.
- A gets access to black boxes of Enc and Dec.
- A submits two distinct chosen plaintexts m_0, m_1 of the same length to the challenger.
- The challenger selects $b \in \{0, 1\}$ randomly, and sends A the ciphertext:$Enc_{pk}(m_b) \rightarrow c_b$.
- A may make further calls to Dec for any message with the restriction that c_b cannot be submitted to the Dec.
- Finally, A outputs a guess for the value of b'. A wins the game if $b = b'$.

Definition 3.4.3. A public-key encryption scheme $\Pi = (Gen, Enc, Dec)$ is CCA-secure if for all PPT adversary A, there exists a negligible function $negl$ such that:

$\Pr\left[PubK^{CCA}_{A,\Pi}(n) = 1\right] \leq \frac{1}{2} + negl(n).$

3.4.2 Types of Security Models

Security assumptions are critical in proving the security of a cryptosystem. Several models have been proposed with each making certain assumptions about properties of some parts of the scheme or the power of adversaries. In this section we briefly introduce two models that are generally used in security proofs.

3.4.2.1 Standard Model

In security proofs, reduction is often used to prove that if a scheme can be broken, then the same technique that breaks the scheme can be used to break a known complexity assumption. If we prove a scheme based on complexity assumptions, then we say that the scheme is secure in the *standard model.*

A proof by reduction in the standard model proceeds as follows: Suppose we want to prove the security of scheme $\mathbb{S}$. First we assume that a computational problem $\mathbb{P}$ is hard to solve. Then we fix a polynomial-time algorithm $\mathbb{A}$ against the scheme $\mathbb{S}$. We also fix a polynomial-time algorithm $\mathbb{B}$ trying to solve $\mathbb{P}$. If $\mathbb{A}$ breaks the scheme $\mathbb{S}$ with a non-negligible probability, then $\mathbb{B}$ solves the hard computational problem $\mathbb{P}$ with a non-negligible probability. However, since we have assumed that the computational problem $\mathbb{P}$ is hard to solve, we get a *contradiction.* At this point, we have "proved" the security of $\mathbb{S}$.

Proving the security of schemes in the standard model is usually difficult. Sometimes a given construction is hard to construct a reduction algorithm which reduces the problem of breaking the scheme to the problem of breaking a standard complexity assumption. In addition, practice has shown that most schemes with a security proof in the standard model are not practical [42]. As an alternative to proving security in the standard model, a number of practical schemes are proven secure in idealized models, including the random oracle model [18].

3.4.2.2 Random Oracle Model

Bellare and Rogaway [18] proposed the *random oracle model* which has been used to prove the security of numerous cryptosystems. In cryptography, a *random oracle* is an oracle (a theoretical black box) that answers every query with a random response chosen uniformly from its output domain. If a query is repeated, the response remains the same each time.

The random oracle model is a heuristic that assumes the existence of a truly random function accessed by all parties in a protocol. Since in reality no such function exists, random oracles are instantiated with hash functions which have been heuristically assumed to be capable of replacing the theoretical random oracles. The random oracle model allows us to prove the security of a cryptosystem with practical efficiency [7] .

In the random oracle model, an attacker only has the oracle access and can compute values only by querying the oracle. The oracle is simulated as follows:

- A simulator answers the adversary's queries and maintains a list of all queried inputs $\{x_i\}$

and their corresponding random values $\{y_i\}$:

- For each query x_i:
    - If the query x_i has been previously asked by the adversary, then the random oracle will output the same value y_i;
    - Otherwise the simulator generates a fixed length random output y_i, returns it to the adversary, and records x_i and its output y_i.

Similarly to the standard model, reduction is used for proving security. If there is an adversary that can break the cryptographic scheme using the random oracle, then the reduction contradicts a complexity assumption. Hence, the cryptographic scheme is argued to be secure. However, Canetti et al. [28] show the existence of encryption and signature schemes that are secure in the random oracle model, but are insecure for any instantiation of the random oracle as a hash function. Despite Canetti et al.'s finding, using random oracles instantiated with hash functions for security proving is still widely used and considered viable.

3.5 Conclusions

In this chapter we have provided the background information for the rest of the thesis. We briefly described groups, fields, elliptic curves and bilinear maps. We have reviewed some of the fundamental concepts in modern cryptography, and introduced the idea of security models.

Chapter 4

Attribute-Based Cryptography

Contents

Attribute-Based Encryption (ABE) and *Attribute-Based Signature* (ABS) schemes are emerging cryptographic primitives. ABE schemes are used to protect data secrecy with fine-grained access policies being automatically enforced during the decryption process. ABS schemes allow users to generate anonymous signatures over a set of attributes, which can protect user privacy during the process of authentication. In this chapter we formally introduce ABE and ABS schemes and provide the prerequisite information about them which is necessary to understand our proposed models and systems in the subsequent chapters.

4.1 Attribute-Based Encryption

In their landmark work [116], Sahai and Waters proposed *Fuzzy Identity Based Encryption* (FIBE) which allows error-tolerance in *Identity Based Encryption* (IBE) for biometric applications. Every identity is viewed as a set of descriptive attributes ω. A user is able to decrypt a ciphertext encrypted by identity attributes $\omega^{'}$ if and only if his identity attributes ω overlap with $\omega^{'}$ for at least a pre-determined threshold value d ($\omega \cap \omega^{'} \geq d$). This idea of designing fine-grained data protection was later formalized as ABE.

In ABE, an *attribute authority* (AA) is used to issue private keys (in the form of key shares) to users. When the system is initialized, a *public key*, a *master secret*, and *system parameters* are generated. Data is encrypted by the public key and system parameters. The master secret is used to derive user private keys that are used to decrypt data. There are two major classes of ABE schemes: *key-policy ABE* (KP-ABE) and *ciphertext-policy ABE* (CP-ABE). In KP-ABE schemes, a user's private key is issued based on an access policy. The ciphertext is associated with a set of attributes defined by the encryptor. A user is able to decrypt data if the attributes associated with the data's ciphertext comply with the access policy associated with the user's private key. The idea is reversed to CP-ABE schemes, where the AA issues a private key based on the attributes held by a user. A ciphertext is associated with an access policy defined by an encryptor. A user can decrypt a ciphertext if the user has the attributes satisfying the access policy of the ciphertext.

One of the crucial security features of ABE is *collusion-resistance*. Since user private keys are all derived from the same master secret, it is important that each user's private key is randomized sufficiently to prevent users from pooling their partial private keys (attribute key shares) together to decrypt any ciphertext.

A KP-ABE scheme controls which decryption keys can decrypt a ciphertext. The access policy is used to derive private keys. CP-ABE controls which users can decrypt a ciphertext. The access policy controls the data decryption process. Thus CP-ABE represents a more intuitive way of protecting data in a distributed system.

Throughout this thesis we sometimes interchange the terms *decryption key*, *attribute key shares*, and *user private keys*. They all refer to the keys that users use to decrypt the data in an ABE scheme.

4.1.1 ABE Scheme Definition

In this section we formally define KP-ABE and CP-ABE schemes. We begin by introducing the access control structure (tree) that is used by most ABE schemes for expressing access policies.

4.1.1.1 Access Structure

An access policy can be expressed in terms of a monotone boolean formula or monotone access structure.

Definition 4.1.1. Let $\mathbb{P} = \{P_1, \dots , P_n\}$ be a set of parties. A collection $\mathbb{A} \subseteq 2^{\{P_1, \cdots , P_n\}}$ is *monotone* if $\forall$ B, C: if $B \in \mathbb{A}$ and $B \subseteq C$ then $C \in \mathbb{A}$. A *monotone access structure* is a monotone collection $\mathbb{A}$ of non-empty subsets of $\mathbb{P}$, i.e. $\mathbb{A} \subseteq 2^{\mathbb{P}} \setminus \{\emptyset\}$. The sets in $\mathbb{A}$ are called *authorized* sets, and the sets not in $\mathbb{A}$ are called *unauthorized* sets.

Access structures can be represented by *access trees* [16]. Access trees are n-ary trees in which leaves are attributes and inner nodes are AND ($\wedge$) and OR ($\vee$) boolean operations. An inner node can also be a threshold gate consisting of a set of nodes (children) and a threshold value. For example, suppose an inner node x is a threshold node (gate), if num_x is the number of children of x, and t_x is the threshold value of x, then $0 < t_x \leq num_x$. In fact AND and OR can be seen as special cases of threshold gates, where AND is the gate of $t_x = 2$, $num_x = 2$, and OR is the gate of $t_x = 1$, $num_x = 2$.

Intuitively, an access tree expresses an access policy specifying which combination of attributes satisfies the access policy. Consider an example, where data can be accessed by the following access policy. The policy dictates that the data can be accessed by either an analyst in the marketing department, or a personnel in the IT, Operation, and Support Department. The access tree is illustrated in Figure 4.1:

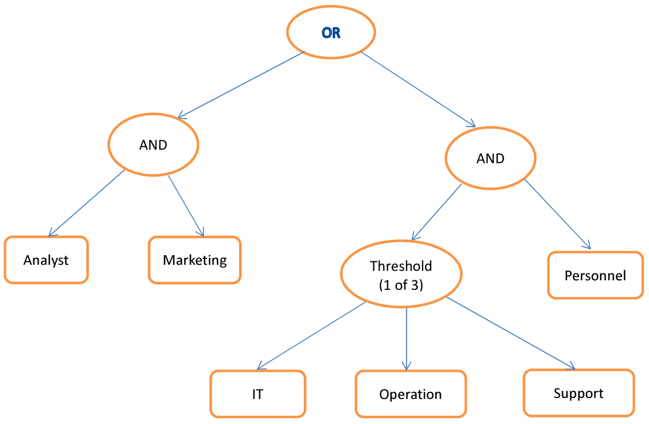


Figure 4.1: Access Tree

Γ_{data} = ((Analyst $\wedge$ Marketing Department) $\vee$ (one of the personnel in one of the following departments: IT, Operation, Support))

In ABE, instead of parties, we use attributes where the access structure $\mathbb{A}$ contains the set of authorized attributes. Most ABE schemes use *secret sharing schemes* [76, 101, 121] to realize access trees for splitting and re-constructing a secret.

4.1.1.2 Definition of a KP-ABE Scheme

A KP-ABE scheme typically consists of four algorithms as follows:

- *Setup* (1^n): takes the security parameter 1^n as input and outputs the public parameters *pk* and a master secret *mk*.
- *KeyGen*$(\Gamma$, *mk)*: takes an access structure Γ and the master key *mk* as inputs, and outputs a decryption key *sk*.

- *Enc*(*pk*, *m*, *S*): takes public parameters *pk*, a message *m*, and a set of attributes S as inputs. It outputs a ciphertext *c*.
- *Dec*(*sk*, *c*, *S*): takes a private key *sk*, a ciphertext *c*, and an attribute set S as inputs. It outputs the message m if the attribute set S complies with the access structure Γ of the decryption key *sk*: if $\Gamma(S) = 1$, $m :=$ *Dec*(*sk*, *c*, *S*); otherwise it outputs $\perp$.

4.1.1.3 Definition of a CP-ABE Scheme

A CP-ABE scheme also consists of four algorithms:

- *Setup*(1^n): takes security parameter 1^n as input and outputs the public parameters *pk* and a master secret *mk*.
- *KeyGen*(*mk, S*): takes master key *mk* and a set of attributes S as inputs, and outputs a decryption key *sk*.
- *Enc*(Γ, *pk, m*): takes an access tree Γ, the public parameters *pk*, and a message m as inputs. It outputs the ciphertext *c*.
- *Dec*(*sk, c, S*): takes a private key *sk*, a ciphertext *c*, and a set of attributes S as inputs. It outputs m if the attribute set S satisfies the access tree: if Γ(S) = 1, $m :=$ *Dec*(*sk*, *c*, S); otherwise it outputs $\perp$.

4.1.2 Related Work

The concept of KP-ABE was first introduced by Goyal et al. [130]. The initial construction of [116] can only handle one threshold gate in an access tree. The expressibility of the access policy was greatly improved in [130]. The work of Goyal et al. [105] proposed a scheme that supports a non-monotone access tree. The primary drawback of [105] is that the size of a private key can blow up with the maximum number of attributes associated with a ciphertext. Lewko et al. [80] improved [105] to dramatically reduce the size of private keys.

The first CP-ABE scheme was proposed by Bethencourt et al. [20]. This scheme allows an access policy expressed by a monotone tree and uses a secret sharing scheme [121] in the construction. Waters [135] took a different approach by using a linear secret sharing scheme (LSSS) [76] to represent an access policy. This approach was later extended by Lewko et al. [82] and Okamoto et al. [102]. Since [20] was only proved to be secure in the generic model, multiple approaches were proposed to construct expressive and efficient CP-ABE schemes in the standard model [62, 71, 82, 135]. Non-monotone access structures in CP-ABE were studied in Cheung et al. [40]. The initial construction was proved to be CPA-Secure under the DBDH assumption, and then was improved to be CCA-Secure by using the Canetti-Halevi-Katz technique [29].

The aforementioned ABE schemes focus on the expressiveness of the access policy and security. However, the size of the ciphertext can grow linearly with the number of attributes. To improve efficiency, schemes with a constant-size ciphertext have been studied in [14, 37, 47, 69] for threshold access trees, and [13, 38] for non-monotone access trees.

Multi-authority ABE (MA-ABE) schemes have been studied in [34, 35, 64, 81, 82, 84, 88]. In a MA-ABE system, multiple AAs work together to generate user private keys, with each AA being responsible for a disjoint set of attributes. One of the biggest challenges is to prevent users from colluding. This requires that the key shares issued by different AAs are linked in a unique way for different users.

4.2 Attribute-Based Signatures

In an ABS scheme, a central AA uses the master secret to issue private signature keys (in the form of key shares) to signers according to the attributes that they have. A signer with an attribute set ω can use his private key to sign a message with any subset of ω. Unlike digital signature schemes [60, 112], an ABS scheme is capable of supporting a complex predicate (expressing an access policy). For example, a signature can be produced by a user who is: ((Student) AND (in Department of Mathematics OR Department of Computer Science)). The person who validates the signature can only know that the signature is generated by a user who is a student in the Department of either Mathematics or Computer Science. No more attributes or identity information are disclosed. In addition, an ABS scheme is collusion resistant, which means that multiple parties cannot collude and combine all their attributes to produce a valid signature if any one of them cannot generate the signature independently.

ABS schemes have three important properties [90] :

- *Unforgeability.* Users cannot forge signatures with attributes they do not possess, even through collusion.
- *Signer privacy.* A legitimate signer can remain anonymous. Signatures generated based on a predicate are indistinguishable among all the users whose attributes satisfy the predicate.
- *Unlinkability.* It is impossible to link different signatures to the same party who generates the signatures, even for the signatures of a fixed message that are generated multiple times by the same party.

4.2.1 ABS Scheme Definition

Generally speaking, an ABS scheme consists of four algorithms. Given a security parameter λ, these algorithms are listed as follows:

- $Setup(1^n)$: The *Setup* algorithm takes the security parameter n as an input. It outputs public parameters pk_{params} and the master secret key mk. The universal attribute set $\mathbb{A}$ is also included in the public parameters pk_{params}.
- $Extract(mk, pk_{params}, S)$: The *Extract* algorithm takes the master secret mk, the public parameters pk_{params}, and a user's attribute set $S \subseteq \mathbb{A}$ as inputs. It generates the user's private key (in the form of attribute private key shares) sk.
- $Sign(pk_{params}, m, \Upsilon, sk)$: This algorithm takes a user's private key sk for the attribute

set S, and public parameters pk_{params} as inputs. It produces the signature σ on m if the user's attribute set satisfies the predicate $\Upsilon(S) = 1$.

- $Verify$(pk_{params}, m, Υ, σ): On receiving a signature σ on message m, the verify algorithm takes the public parameters pk_{params}, and the signing predicate Υ to validate the signature. It outputs a boolean value, either *accept* if the signature is validated or *reject* otherwise.

4.2.2 Related Work

A *threshold ABS* (*t-ABS*) scheme was defined and formalized by Shahandashti and Safavi-Naini [120]. Li et al. [83] proposed a scheme to reduce signature sizes of the scheme [120] by half. Herranz et al. [68] constructed a constant-size signature scheme based on [63] and [127]. Ge et al. [52] presented a construction that only needs three pairing operations.

Monotone predicate ABS schemes were studied in [30, 31, 53, 91, 126]. Maji et al. [91] proposed a generic framework for constructing ABS schemes. Ge et al. [53] constructed a short signature scheme in the standard model. Cao et al. [30, 31] proposed schemes in which the predicate can be expressed in AND, OR, threshold gates. Su et al. [126] constructed a scheme supporting AND, OR, and threshold gates in the standard model. A non-monotone predicate ABS scheme was proposed by Okamoto [103]. This scheme combines inner-product with non-monotone access tree, where attribute x for signing key sk_x is a tuple of attribute vectors, and predicate v for verification is a span program (M, ρ) along with a tuple of attribute vectors.

The concept of *multi-authority* ABS (MA-ABS) was introduced by Maji et al. [90], where multiple authorities generate a user secret key together, with each authority only responsible for issuing a partial secret key associated with a subset of the attributes. Cao et al. [32] proposed a MA-ABS scheme with a centralized trusted authority for generating the secret keys used by distributed authorities to derive user private keys. Okamoto and Takashima [104] removed the dependency of the centralized trusted authority and proposed a completely decentralized MA-ABS scheme, with each authority independently issuing a partial signing key to users.

Attribute-based group signatures (ABGS) [48, 79] and *ring signatures* (ABRS) [85, 134, 138] are two variations of ABS. ABGS is based on the concept of the group signature (GS) [36], which is a method allowing a member of a group to sign a message on behalf of the group. However, the resulting signature keeps the identity of the signer secret. The main difference between ABGS and ABS is that there is a group manager in ABGS, who can identify a signature and reveal the identity of the signer when necessary. ABRS is based on the concept of a ring signature (RS) [113], in which a signer can form a ring to include anyone whose public key is accessible by the signer and verifiers. In ABRS, those users who form a ring are replaced by a set of attributes that are additional to the necessary attributes.

4.2.3 Construction of an ABS Scheme Construction

In this section, we explain the threshold ABS scheme construction of [83]. We will use this scheme to demonstrate our idea for providing anonymous user revocation in Chapter 7. There are three constructions presented in [83]. The first construction is proved secure under the random oracle model, while the second construction removed the random oracle assumption.

Those two constructions use a central AA to manage and issue user private keys (or signing keys). To reduce the trust of the central AA, they proposed a *MA-ABS threshold* scheme. Since the second and third constructions are based on the first construction, we will only show the first construction described in their paper as follows:

- *Setup:* First, randomly select an element from $\mathbb{Z}_p$ for each attribute in the universe attribute set $\mathbb{U}$. A default set of d - 1 attributes from $\mathbb{Z}_p$ is defined as $\Omega = \{\Omega_1, \Omega_2, \cdots, \Omega_{d-1}\}$. Two multiplicative cyclic groups G_1 and G_2 of prime order p are defined. A bilinear map is defined as: e: $G_1 \times G_1 \rightarrow G_2$. Select a random generator $g \in G_1$, a random $x \in Z_p^\star$, and set $g_1 = g^x$. Next, pick a random element $g_2 \in \mathbb{Z}_p^\star$ to compute $Z = e(g_1, g_2)$. Two hash functions are chosen: H_1, H_2: $\{0, 1\}^\star \rightarrow G_1$. The public parameters are $params = \{g, g_1, g_2, Z, d, H_1, H_2\}$ and the master secret is x.
- *Extract:* To generate a private (signing) key for an attribute set ω, the following steps are taken:
  - Randomly choose a d - 1 degree of polynomial $q(y)$ and let $q(0) = x$;
  - Generate a new attribute set $\hat{\omega} = \omega \cup \Omega$. For each $i \in \hat{\omega}$, choose $r_i \in_R \mathbb{Z}_p$ and compute $d_{i0} = g_2^{q(i)} \cdot H_1(i)^{r_i}$ and $d_{i1} = g^{r_i}$;
  - Finally, output $D_i = (d_{i0}, d_{i1})$ as the private key for each $i \in \hat{\omega}$.
- *Sign:* Suppose a user has a private key for his attribute set ω. To sign a message m with predicate $\Upsilon_{k,\omega^\star}(\cdot)$, that is to prove owning at least k attributes among an n-element attribute set $\omega^\star$, select a k-element subset $\omega^{'} \subseteq \omega \cap \omega^\star$ and proceed as follows:

- First, select a default attribute subset $\Omega^{'} \subseteq \Omega$ with $\left|\Omega^{'}\right| = d$ - k, and $n + d$ - k random values $r_i^{'} \in \mathbb{Z}_p$ for $i \in \omega^\star \cup \Omega^{'}$;
- Compute $\sigma_0 = \left[\prod_{i\in\omega^{'}\cup\Omega^{'}} d_{i0}^{\triangle_{i,S}(0)}\right]\left[\prod_{i\in\omega^\star\cup\Omega^{'}} H_1(i)^{r_i^{'}}\right] H_2(m)^s$, $\{\sigma_i = d_{i1}^{\triangle_{i,S}(0)} g^{r_i^{'}}\}_{i\in\omega^{'}\cup\Omega^{'}}$, $\{\sigma_i = g^{r_i^{'}}\}_{\omega^\star/\omega^{'}}$, and $\sigma_0^{'} = g^s$, with a randomly selected value $s \in \mathbb{Z}_p$;
- Finally, output the signature $\sigma = \{\sigma_0, \{\sigma_i\}_{i\in\omega^\star\cup\Omega^{'}}, \sigma_0^{'}\}$.

- *Verify*: To verify the signature $\sigma = \{\sigma_0, \{\sigma_i\}_{i\in\omega^\star\cup\Omega^{'}}, \sigma_0^{'}\}$ of message m with threshold k for attributes $\omega^\star \cup \Omega^{'}$, check if the following equation holds:

$$\frac{e(g, \sigma_0)}{\left[\prod_{i\in\omega^\star\cup\Omega^{'}} e(H_1(i), \sigma_i)\right] e(H_2(m), \sigma_0^{'})} \stackrel{?}{=} \mathrm{Z} .$$

4.3 Conclusions

In this chapter we provide an introduction to ABE and ABS schemes with some formal definitions. We also present the constructions of a specific CP-ABE and threshold ABS schemes that will be used in subsequent chapters.

Chapter 5

Generic User Revocation Systems for ABE in Cloud Storage

Contents

In Chapter 4, we introduced ABE. As ABE schemes have the inflexible user, and attribute management issues (see section 2.6.2.3), they are potentially not practical in cloud storage environments. In this chapter, we construct two user revocation systems for ABE schemes to manage user revocations dynamically. Although user revocation has been studied in the literature, most current user revocation systems only work with some particular ABE schemes, and are not flexible enough to be adapted to work with general ABE schemes.

User revocation is the process of removing a user's privilege of data access. In most cases, user identities are used for user identification in the first part of revocation. However, in untrusted cloud storage, user identities need to be protected and kept away from CSPs for the reasons we specified in Section 2.5.1. Since most existing user revocation schemes have not considered user privacy protection, a data owner has to mediate every request in order to protect user privacy. This means that a user's request needs to be routed from CSPs to the data owner before the user's privilege is determined. This is not scalable. Therefore a practical user revocation system for ABE schemes in an untrusted cloud storage environment should be capable of identifying a revoked user anonymously without including a data owner or a trusted party in the data retrieval process.

With those aspects in mind, we design and build two dynamic user revocation systems working with existing ABE schemes to protect data and user privacy in cloud storage. These two user revocation systems have the following three features:

- Generic: the revocation capability can be directly applied to any ABE scheme.
- Dynamic: the revocation is instantaneous with no need to re-issue any ABE private key to users.

- Anonymous: users are anonymous to CSPs. A revoked user is anonymously prevented from accessing data in the data retrieval or decryption process.

5.1 Related Work

There has been prior research into dealing with the practical problems with implementation of ABE schemes, particularly with respect to revocation issues.

Pirretti et al. in [108] associated attributes with expiry times. This idea was enhanced by Bethencourt et al. [20] who suggested to associate private keys (or key shares) with expiry times. Both schemes require users to periodically contact the AA for new private keys (or key shares), which has potential scalability issues as well as being incapable of revoking users instantly.

Junod and Karlov [75] constructed a CP-ABE based broadcast encryption scheme that supports direct (or instant) user revocation. In their scheme, each receiver's identity is mapped to an individual attribute. The access policy consists of a set of system attributes with a set of identity attributes. Individual user revocation is achieved by updating the set of identity attributes in the access policy. This scheme is not efficient when applied to cloud storage systems since mapping each user's identity to an attribute can make the ciphertext grow linearly. It also discloses user identities to CSPs. In addition, data owners should not be directly involved in controlling data distribution after the data has been stored at CSPs.

Jahid et al. [72] achieved user revocation by utilizing a semi-trusted proxy to participate in the decryption process. In their proposed scheme, each user obtains an identity key in addition to their attribute key shares. The identity keys are generated by a data owner using a secret sharing scheme. The data owner also generates a proxy key for the proxy, who uses the proxy key to transfer the ciphertext in such a way that only non-revoked users with their identity keys can decrypt the data. The proxy key is regenerated whenever a user is revoked. Although the scheme achieves dynamic user revocation without attribute key regeneration, it can only revoke a predefined number of users. In addition, adding a new user to the system can trigger re-keying all the existing identity keys, which exhibits a potential scalability issue and some key management issues.

Hur and Noh [70] used attribute key encryption keys (KEKs) to address user revocation for BSW's CP-ABE scheme [20]. Their scheme requires a data service manager (such as a CSP) to generate attribute KEKs and distribute the keys to users. The attributes in the access policy of a ciphertext are re-encrypted by their KEKs before the ciphertext is sent to a user. When a user is revoked, the impacted attribute KEKs are updated and redistributed. This approach brings potential management overheads. The attribute KEKs are generated and maintained via a global binary tree that assigns users to the leaf nodes. For a large group of users, maintaining the binary tree becomes much harder when the system needs to add or delete users. The data service manager also has to know every user's attribute set in order to generate and distribute their attribute KEKs, which requires trust in the data service manager. This kind of trust does not exist in untrusted cloud storage environments. In addition every user needs to have two sets of keys: secret attribute key shares and attribute KEKs.

Another user revocation approach is to associate ciphertext with expiration times. In this approach, ciphertexts are encrypted with additional time stamp attributes along with periodic key updates. Sahai et al. [117] proposed a revocable storage that prevents revoked users from accessing the data once their access rights are removed. They introduced the notion of *ciphertext delegation*, where a ciphertext encrypted under a certain policy can be re-encrypted to a more restrictive policy using only public information. Using ciphertext delegation, CSP can re-encrypt the ciphertexts from time t to time $t + 1$. However, this mechanism requires user private keys to be updated periodically for all non-revoked users, which can potentially create a key distribution overhead in a large group of users. Users also have to keep different versions of the keys to decrypt the data.

As most of the existing user revocation schemes work with individual schemes, we proposed a flexible revocation model (dynamic user revocation and key refreshment (DURKR)) which can work with CP-ABE schemes for dynamic user revocation and key refreshing [141]. In the model, the master secret is split into two parts. One part is used by CP-ABE schemes to issue the attribute private keys. The other part is used to issue a proxy delegation key that is used to re-encrypt ciphertexts of CP-ABE so that only non-revoked users can decrypt data eventually. Although the model is intended to be flexible to work with any type of CP-ABE schemes, it requires some modification of the underlying scheme to incorporate an additional key share. In our other work [142], a generic framework was proposed to be adopted by any type of ABE scheme. In addition, the framework also avoids the potential key escrow in ABE schemes with centralized AA settings. However, the framework does not effectively prevent revoked and non-revoked users from colluding.

The application of ABE to cloud environments has been studied in the literature. Yu et al. [147] proposed a scheme to protect data files in a semi-trusted cloud environment. Wang et al. [132] built a hierarchical CP-ABE (HABE) model using the notion of a hierarchical IBE (HIBE) scheme. Parno et al. [106] constructed a verifiable computation scheme with public delegation and verifiability based on an ABE scheme. Yang et al. [145] proposed an access control framework using a CP-ABE scheme for cloud storage systems. The framework also enables dynamic attribute revocation. Yu et al. [146] proposed a CP-ABE scheme with a hidden access policy for untrusted content distribution networks. Their scheme also uses periodic key expiration to manage user revocation.

5.2 Our Contributions

We design and construct two data protection systems using ABE schemes in untrusted cloud storage. As ABE protects data privacy, the newly developed systems focus on building generic user revocation mechanisms that could work with any ABE scheme and also protect user privacy during the revocation process.

To achieve the generic and anonymous user revocations, we aim at controlling the user access to ABE ciphertexts. If a revoked user cannot obtain an ABE ciphertext, then the user will not be able to decrypt the data even though the user might hold a valid ABE private (decryption) key. If such a control of ciphertext access is independent from ABE, then the revocation mechanism can be generically applied to any ABE scheme. With these two considerations in mind, our solution is to build an extra layer on the top of an ABE scheme to control access to ABE ciphertexts. This layer, utilizing a dynamic and anonymous accumulator technique, treats ABE as a black box. It removes the need of a data owner to mediate every data retrieval and makes users anonymous to CSPs. Using this layer, two systems are built as follows:

- User Revocation via Ciphertext Re-encryption (UR-CRE): This system is to re-encrypt ABE ciphertexts with accumulator (group) keys, so that only non-revoked users are able to decrypt the re-encrypted ABE ciphertexts to get ABE ciphertexts. Users are anonymous to CSPs during the data retrieval process. The benefit of this re-encryption system is three-fold:
    - It enables the desirable user revocation control (generic, dynamic, and anonymous) for any type of ABE scheme.
    - It protects the integrity and secrecy of the access policy embedded in ABE ciphertexts. Since access policies of CP-ABE can be parts of the ciphertexts in plaintext format, they are vulnerable to alteration and information leakage in regard to who can decrypt the data. The same issue exists in KP-ABE, where attributes are also in plaintext format.
    - Data is self-protected. Revoked users are not able to decrypt the re-encrypted ciphertexts during the decryption process automatically.

The drawback of this system is the overhead associated with the ciphertext re-encryption and key management of accumulators on the data owner side.

- User Revocation via Cloud Storage Providers (UR-CSP): This system separates the user revocation control from data encryption, and utilizes untrusted CSPs to eliminate the overhead associated with ciphertext re-encryption. We use the anonymous accumulator mechanism to achieve the desired user verification (generic, dynamic, and anonymous), so that user privacy is protected during the verification process. Any revoked user is anonymously verified by CSPs and prevented from accessing ABE ciphertexts after the user's revocation.

  Since ABE ciphertexts are not re-encrypted when stored at CSPs, the possible downside of this approach is that access policies included in ciphertexts are exposed to CSPs or any adversary in the unencrypted channels during transmission. The integrity and privacy of access policies are at risk.

As accumulators are the cornerstones of building our revocation systems, we will introduce the dynamic accumulator concept first, followed by the algorithm construction used for the rest of this thesis.

5.3 Accumulator

Accumulators were first introduced by Benaloh and de Mare [19] as a method to condense a set of values (elements) into one value (referred as an accumulator's aggregate value or simply an aggregate value in this thesis), such that a short *witness* is used to demonstrate that a value has been condensed in an aggregate value. The *witness* has the following characteristics:

- It is unique if the value being aggregated is unique.
- It is infeasible to forge a witness for a value that has not been accumulated.
- No information about the accumulated value can be leaked through the value of the witness or the aggregate value.

An accumulator is called *universal* if it supports efficient zero-knowledge membership and non-membership proofs. An accumulator is called *dynamic universal* if the accumulator's aggregate value, membership, and non-membership witnesses can be updated efficiently.

Camenisch and Lysyanskaya introduced the notion of a *dynamic universal accumulator* (DUA) [27]. The cost of adding or deleting an element is independent of the number of elements accumulated. Using a zero-knowledge protocol, the scheme can prove whether a committed value is in an accumulator or not. The scheme was constructed under the strong RSA assumption. A DUA based on a bilinear map was proposed by Nguyen [100]. Au et al. [15] extended [100] to build a DUA under the DDH assumption. The DUA has been used for anonymous credential attestation [26].

5.3.1 A Bilinear Map Based Accumulator

The accumulator scheme used in our systems is based on the scheme of Au et al in [15]. The following algorithm descriptions come from the paper [15] directly.

- *Setup*(1^n): This algorithm takes the security parameter 1^n as the input. It initializes and outputs system parameters as follows:
  - Let e: $G \times G \rightarrow G_T$ be a bilinear map of prime order p.
  - Let A_k: $2^{Z_p^\star} \rightarrow G$ be an accumulator's function that is parametrized by a randomly selected $\beta \in Z_p^\star$: $A_k(X\ ) = g^{\prod_{x \in X}(x+\beta)}$, where $x \in Z_p$, $A_k(X) \in G$, and β is the trapdoor or auxiliary information.
  - Finally output the system parameters: $\{G,\ G_T,\ e,\ A_k,\ \beta\}$.
- $MembershipVerification(w, x, v)$: The algorithm takes a witness w, element x, and accumulator aggregate value v as the inputs. It proves the membership of x as follows:
  - If $e(w,\ g^x \cdot\ g^\beta) = e(v,\ g)$, output "$true$";
  - Otherwise output "$false$";
- $MembershipWitness(x)$: This algorithm takes an element x as input and computes the witness of x as follows:
  - For a set of elements $Y = \{x_1,\ x_2,\ ...\ ,\ x_n\} \in Z_p$, a membership witness for $x \in\ Y$ is calculated as:

$$w = \left[g^{\prod_{i=1}^{n}(x_i+\beta)}\right]^{\frac{1}{x+\beta}} .$$

- Output the witness: w.

- *UpdateAccumulator*$(x^{'}, v, Ops)$: The algorithm takes a new element $x^{'}$, an accumulator aggregate value v, and the operation type Ops ("*Add*" or "*Delete*") as inputs. It updates v to $v^{'}$ as follows:
  - If Ops = "*Add*", then $v^{'} = v^{x^{'}+\beta}$;
  - If Ops = "*Delete*", then $v^{'} = v^{\frac{1}{x^{'}+\beta}}$;
  - Output $v^{'}$.

- *UpdateMembershipWitness*$(w, x, v, \widehat{v}, Ops, x^{'})$: This algorithm takes the original membership witness w of x with respect to the accumulator aggregate value v, the new aggregate value $\widehat{v}$ having $x^{'}$, the operation type Ops, and the element $x^{'}$ as inputs. It computes the new witness of x in regard to $\widehat{v}$ as follows:
  - If Ops = "*Add*", then $x^{'}$ is the newly added element to $\widehat{v}$. The new membership witness $w^{'}$ of x can be computed as $w^{'} = vw^{x^{'}-x}$;
  - If Ops = "*Delete*", then $x^{'}$ has been removed from v, the new membership witness $w^{'}$ of x can be calculated as $w^{\frac{1}{x'-x}}\widehat{v}^{\frac{1}{x-x'}}$;
  - Output $w^{'}$.

5.3.2 Dynamic Accumulator (DA)

We build a dynamic accumulator (DA) using the scheme in [15] to manage user access to data. We make the following adjustments to fit our systems:

- We remove the $UpdateMemebershipWitness$ algorithm. Users and CSPs are neither trusted in our trust models with the system parameters, nor allowed to update their witnesses. Data owners are responsible for centrally managing accumulators and user witness updates.
- We want to improve the efficiency and scalability of the scheme in [15]. Like most of the existing accumulator schemes, the scheme in [15] has two potential limitations for large-scale systems: i) an accumulator has a pre-defined number of elements to be aggregated; ii) adding or deleting an element triggers witness updates of all elements aggregated in the same accumulator. To make the scheme more practical and minimize the impact of witness updates, we will use multiple accumulators to manage non-revoked users. This adjustment can be scalable in the following ways:
  - The witness update only impacts a small numbers of users. Non-revoked users will be divided into groups. Each user only belongs to one group and is aggregated to an accumulator. When a user is revoked, the user is removed from the user's accumulator. The witness updates thus only impact a smaller number of users.
  - There is no limitation on the number of users in the system. The number of accumulators can be dynamically increased or decreased depending on the number of

users in the systems.

- Each accumulator is associated with one internal identifier and one external identifier. The differences between the internal and external identifiers are as follows:
    * The internal identifiers (or indexes) are kept by data owners. They are static and sequential. An accumulator is deleted when all associated users being removed. However its index is not recycled.
    * The external identifiers are given out to users and CSPs to identify or look up accumulators. They are randomized every time when the associated accumulators are updated, so that the static index numbers are not directly exposed to users and CSPs for extra protection. Furthermore, the numbers used to randomize external identifiers have two main purposes: 1) to randomize ciphertext re-encryption keys when ABE ciphertexts need to be re-encrypted in UR-CRE (Section 5.4.2); 2) to randomize accumulators' aggregate values each time when the aggregate values are updated in UR-CSP (in Section 5.4.3).

To achieve better efficiency and scalability, the actual implementations can delegate the management of accumulators, especially witness updates and distribution, to a data-owner trusted third party.

Our modification does not change the mathematical and complexity theory that DUA [15] is based on. Its security strength remains the same without relying on the number of elements that can be accumulated into an accumulator, nor the number of accumulators in a system.

Notation	Description
gid	A user's global unique identifier
Φ	Accumulator container containing a set of accumulators
k	The maximum number of elements that can be aggregated into an accumulator
A_k	The accumulator function
β	The trapdoor value for the accumulator function A_k
α	The container of an accumulator and its information
v	An accumulator aggregate value
Ops	The accumulator operational type - *Add* or *Delete* an element to or from an accumulator
ω_α	The witness container storing user witnesses of accumulator α
w_{gid}	User gid's witness of an accumulator
Ind	An integer counter storing the next created internal identifier (index) value for an accumulator

Table 5.1: DA Notation

5.3.2.1 Data Structure Definition

Table 5.1 lists the commonly used notation in this chapter.

The major data structures used by DA are defined as follows:

- An accumulator α contains: $\{v,\ \{gid_x\}_{1\leq x\leq|\{gid_x\}|},\ i,\ y\}$, where v denotes the aggregate value by aggregating the elements in $\{gid_x\}$, $|\{gid_x\}|$ denotes the number of gids in α, i is the internal identifier (also referred as internal index) of α, y is randomly selected from Z_p to generate the external identifier in the form of g^y.

  Each component of α can be further denoted as follows:

  - $\alpha^1 = v$
  - $\alpha^2 = \{gid_x\}_{1\leq x\leq|\{gid_x\}|}$
  - $\alpha^3 = i$

- $\alpha^4 = y$

- Φ stores a set of accumulators: $\Phi = \{\alpha_i\}_{1 \le i \le |\Phi|}$, where $|\Phi|$ denotes the number of accumulators included.

- w_{gid} is the witness of user *gid* for an accumulator. Each witness consists of two parts: $w_{gid} = \{w^1_{gid}, w^2_{gid}\}$. The details can be found in Section 5.4.

- ω_a is a witness container to store witnesses of the users in α: $\{\{w_{gid_i}, gid_i\}_{1 \le i \le |\alpha^2|}\}$.

5.3.2.2 Algorithm Construction

Our DA consists of the following algorithms. The details of their usage and information sharing among parties will be specified in the following sections.

1. *AccSetup*(G, g): takes a group G and a generator g of G as the inputs, as shown in Algorithm 5.1. It defines the system parameters used by DA: the maximum number of elements k in each accumulator, the trapdoor value β, the accumulator function A_k, and the index counter Ind of internal accumulator identifiers. It initializes Φ and returns $\{k, \beta, A_k, \Phi\}$.

2. *AccAdd*(*gid,* Φ): takes a user's global identifier *gid* and the accumulator container Φ as the inputs, as shown in Algorithm 5.2. The algorithm finds the first accumulator in Φ that has less than k *gids*. A new accumulator is created if none is found. The algorithm adds the *gid* to the accumulator and outputs the updated accumulator and Φ.

3. *AccDelete*(*gid*, Φ): takes a user's global identifier *gid* and the accumulator container Φ as the inputs, as shown in Algorithm 5.3. It looks for *gid* in accumulators stored in Φ:
   (a) If an accumulator can be found, it removes the *gid* from the accumulator. If the *gid* is the only element in the accumulator, the accumulator is emptied out. Finally, it returns the accumulator and Φ.
   (b) If no accumulator can be found, return $\perp$.
4. *AccWitUpdate*(α): takes an accumulator α as the input, as shown in Algorithm 5.4. It updates all the witnesses of *gids* in α, and outputs witnesses set ω_α.

Algorithm 5.1 AccSetup

```
procedure AccSetup(G, g)
1: Begin
2:     Select $k$ to be the maximum number of elements aggregated in an accumulator.
3:     Randomly select $\beta \leftarrow Z_p^\star$; // The trapdoor of the accumulator function.
4:     Define $A_k$: $2^{Z_p^\star} \rightarrow G$ to be an accumulator function. // $A_k(X) = g^{\prod_{gid_i \in X}(gid_i+\beta)}$, where $g$ is the generator of $G$.
5:     Let $v \leftarrow g$; $\alpha \leftarrow \{v, \{\}, 1, null\}$; $\Phi \leftarrow \{\alpha\}$; // Initialize an accumulator and accumulator container.
6:     Set $Ind = 2$; // Set the internal index counter to be the next value.
7:     // Note: $Ind$ is internally used. For simplicity, it is not returned.
8:     Return $\{k, \beta, A_k, \Phi\}$;
9: End.
```

5.4 Generic User Revocation Systems

We design and construct two generic user revocation systems based on ABE schemes in untrusted cloud storage environments. In our systems, each user is given a unique global identifier *gid*. Non-revoked users are randomly divided into groups so that each user only belongs to one

Algorithm 5.2 AccAdd

```
procedure AccAdd(gid, $\Phi$)
1: Begin
2:    Let $N = |\Phi|$; // Get the number of accumulators in $\Phi$.
3:    Set $j = 0$;
4:    For $i = 1$ to $N$ do // Find the first $\alpha$ that has less than $k$ gids.
5:       If $|\alpha_i^2| < k$ then
6:          $j = i$; abort For loop; // The accumulator is found.
7:       end If;
8:    End For
9:    If $j == 0$ then // No accumulator is found. Create a new one.
10:      $j = N + 1$;
11.      $\alpha_j \leftarrow \{g, \{\}, Ind, null\}$; // Initialize an accumulator.
12:      $Ind = Ind + 1$; // Increase the counter by 1, so it always points to the next value.
13:   End If
14: // Add gid to t $\alpha_j$.
15: // $\alpha_j^3$ is static and not changed once it is set. $\alpha_j^4$ is dynamic and will be changed outside this
    algorithm.
16:  $\alpha_j \leftarrow \{v_j^{gid+\beta}, \alpha_j^2 + \{gid\}, \alpha_j^3, \alpha_j^4\}$;
17:   Update $\Phi$ with $\alpha_j$; //Using $\alpha_j^3$, the internal index, to find and replace the existing one, or add
the new one.
18:   Return $\{\alpha_j, \Phi\}$.
19: End.
```

group. Each group of user *gid*s are aggregated into an accumulator. We may interchange the terms of group and accumulator as they are one-to-one relationship with respect to a user. Each accumulator is identified by one internal index (or identifier) and one external identifier. The internal index is fixed when the accumulator is created. The external identifier is changed and randomized each time a user is added or removed from the accumulator. Only external identifiers are given to users and CSPs to request or locate the re-encrypted ciphertext in UR-CRE, or to generate membership proofs in UR-CSP.

Algorithm 5.3 AccDelete

```
procedure AccDelete(gid, $\Phi$)
1: Begin
2:   Let $N = |\Phi|$; // Get the number of accumulators in $\Phi$.
3:   Set $j = 0$;
4:   For $i = 1$ to $N$ do
5:     If $gid \in \alpha_i^2$ then // Find the the accumulator having the gid.
6:       If $\left|\alpha_i^2\right| > 1$ then
7:         $v_i = v_i^{\frac{1}{gid+\beta}}$; // Remove gid from the aggregate value $\alpha_i^1$.
8:         $\alpha_i \leftarrow \{v_i, \alpha_i^2 - \{gid\}, i, y_i\}$ ; // Update the values in $\alpha_i$.
9:         Update $\alpha_i$ in $\Phi$.
10:      Else
11:        $\alpha_i \leftarrow \{g, \{\}, N, null\}$; // Empty the accumulator since gid is the last one.
12:        $\Phi \leftarrow \Phi - \{\alpha_i\}$; // Remove $\alpha_i$ from $\Phi$.
13:      End If;
14:      $j = i$; abort;
15:    End If
16:  End For
17:  If j > 0 then
18:    Return $\{\alpha_j, \Phi\}$; // Return the accumulator. $\alpha_j$ can be an empty accumulator removed from $\Phi$.
19:  Else return $\perp$;
20:  End if;
21: End.
```

5.4.1 Algorithm Definition and Construction

We define the commonly used algorithms in our two systems. Whether the algorithms are run privately or publicly, and which information should be kept secret depend on the trust models. These will be explained in the system design and construction sections of the corresponding user revocation system.

Since our systems can work with any ABE scheme, the $\cdot$ will be used to generically denote the parameters needed by ABE algorithms: $Setup_{abe}$, $KeyGen_{abe}$, Enc_{abe}, and Dec_{abe}.

Algorithm 5.4 AccWitUpdate

```
procedure AccWitUpdate($\alpha$)
1: Begin
2:     Let $\omega_\alpha = \{\}$; // Start with an empty set.
3:     Let $N = |\alpha^2|$; // Set the number of gids to $N$.
4:     For $i = 1$ to $N$ // Compute witnesses.
5:         $w^1_{gid_i} = v^{\frac{1}{gid_i+\beta}}$; // $v$ is the accumulator value: $\alpha^1$.
6:         $w^2_{gid_i} = g^{gid_i+\beta}$;
7:         $w_{gid_i} = \{w^1_{gid_i} , w^2_{gid_i}\}$
8:         $\omega_\alpha \leftarrow \omega_\alpha + \{w_{gid_i}, gid_i\}$; // Add the user's witness to $\omega_\alpha$.
9:     End For
10: Return $\omega_\alpha$;
11: End.
```

Table 5.2 lists the commonly used notation in addition to table 5.1.

- *Setup*(1^n): The algorithm takes the security parameter 1^n as an input and proceeds as follows:
  - Let U contain the *gid*s who are granted for data access: $U = \{gid_1 , \ldots gid_n\}$, where $gid_i \in Z_p^\star$ (p is a n-bit prime).
  - Call $Setup_{abe}$ (1^n) with input 1^n to initialize the ABE scheme. It outputs an ABE public key pk_{abe} and master secret mk_{abe}.
  - Initialize DA using the security parameter 1^n:
    * Let e: $G \times G \rightarrow G_1$ be a bilinear map for an n-bit prime p.
    * Let g be a generator of G.
    * Call *AccSetup*(G, g) (Algorithm 5.1) with G and g as the inputs. It outputs $\{k$, β, A_k, $\Phi\}$.

Notation	Description
c	An ABE ciphertext
AA	The ABE attribute authority
$Setup_{abe}$	The setup algorithm of an ABE scheme
$KeyGen_{abe}$	The key generation algorithm of an ABE scheme
Enc_{abe}	The encryption algorithm of an ABE scheme
Dec_{abe}	The decryption algorithm of an ABE scheme
pk_{abe}	The public key of an ABE scheme
mk_{abe}	The master secret of an ABE scheme
usk_{abe}	A user ABE private key
U	The data set containing non-revoked *gid*s in the system
CSP	Cloud Storage Provider
DO	Data owner
wsk_{gid}	The witness private key of user *gid*
Λ	The container storing accumulator aggregate values (v)
$c^{'}$	A re-encrypted ciphertext of c for an accumulator
$C^{'}$	The container storing multiple $c^{'}$s.

Table 5.2: User Revocation System Notation

- $KeyGen_{abe}(\cdot)$: This is the ABE key generation algorithm for generating user private keys. The symbol $\cdot$ denotes the input parameters. It outputs a user private key usk_{abe} if the user is authenticated. Otherwise the output depends on the output of AA.

- $UserManager(gid,\ Ops,\ \Phi)$: The algorithm takes a user's *gid*, an operation type *Ops,* and the accumulator container Φ as inputs. It proceeds as follows:

  – If *Ops* == "*Add*", call *AccAdd*(*gid*, Φ) (Algorithm 5.2) to add the *gid* to an accumulator:

    $\{\alpha,\ \Phi\}$ = *AccAdd*(*gid*, Φ).

  – If *Ops* == "*Delete*", call *AccDelete*(*gid, Φ)* (Algorithm 5.3) to remove the *gid* from an accumulator:

$\{\alpha, \Phi\} = AccDelete(gid, \Phi)$.

- If $\alpha^2 \neq \{\}$, re-randomize the external identifier and issue a new witness for each $gid \in \alpha^2$:
    - Select a random number uniformly: $y \leftarrow Z_p$. Set $\alpha^4 = y$; update α in Φ.

      We randomize an accumulator external identifier each time the accumulator is updated. y will be used to generate the external identifier (in the form of g^y) for α.
    - Call $AccWitUpdate(\alpha)$ (Algorithm 5.4) to re-compute all witnesses for users in α^2:

      $\omega_\alpha = AccWitUpdate(\alpha)$.
- Return $\{\omega_\alpha, \alpha\}$ if $\alpha^2 \neq \{\}$, otherwise return $\perp$.

• $WitnessKeyGen(\omega_\alpha, \alpha)$: The algorithm takes witnesses container ω_α, and accumulator α as the inputs. It generates witness keys as follows:

- Extract y from α^4 and compute g^y as the external identifier of α;
- Let $witKeys = \{\}$ be the container to store witness keys of users in α^2.
- For each $gid_i \in \alpha^2$:
    - Extract w_{gid_i} from ω_α.
    - Randomly select $r_i \leftarrow Z_p$, and compute the witness key as:

$wsk_{gid_i} = \{wsk^1_{gid_i} = (w^1_{gid_i})^{r_i y} = g^{\frac{r_i y (\prod_{gid_x \in \alpha^2}(gid_x+\beta))}{gid_i+\beta}}$, $wsk^2_{gid_i} = (w_{gid_i})^{1/r_i} = g^{\frac{(gid_i+\beta)}{r_i}}$, $wsk^3_{gid_i} = g^y\}$;

        * Add the witness key to $witKeys$: $witKeys \leftarrow witKeys + (gid_i,\ wsk_{gid_i})$;

    – Return $witKeys$.

- $Enc_{abe}(m,\ \cdot)$: This is the ABE encryption algorithm. It takes data m and the rest of parameters $\cdot$ as inputs. It encrypts data m and outputs the ciphertext c: $\{c_m,\ \cdot\}$.

- $ReEnc(c,\ \Phi)$: The algorithm takes an ABE ciphertext c and the accumulator container Φ as the inputs. It re-encrypts c for each accumulator $\alpha_i \in \Phi$ as follows:

    – For $\alpha_i \in \Phi$, re-encrypt c:
    $c_i^{'} = \{ce(g^{\alpha_i^4},\ \alpha_i^1),\ g^{\alpha_i^4}\} = \{ce(g^{y_i},\ v_i),\ g^{y_i}\} = \{ce(g,\ g)^{y_i \prod_{gid_x \in \alpha_i^2}(gid_x+\beta)},\ g^{y_i}\}$.

    Since $\cdot$ of c may contain the access policy or attributes in clear text format, there is a possibility of leaking user information, such as user attributes, to untrusted CSPs. Re-encrypting the entire c provides the protection of integrity and privacy for access policies and attributes from malicious modifications.

    Although this algorithm is only used by the ciphertext re-encryption system, a possible issue to consider is that ABE ciphertext c may have different message space than the group G_1 used in DA. One way to address this is to define a reversible converting method that maps a ciphertext of ABE to a group element of G_1 in the process of re-encryption, then converts results back in the process of decryption. We will not elaborate it here. It can be addressed in the actual implementation.

    – Finally, output $C^{'} = \{c_i^{'}\}_{1 \leq i \leq |\Phi|}$.

- $Dec(c^{'},\ wsk_{gid})$: The algorithm takes a re-encrypted ciphertext $c^{'}$ and a user's witness key wsk_{gid} as the inputs. It decrypts $c^{'}$ and outputs the ABE ciphertext c as follows:

– Decrypt ABE ciphertext c if wsk_{gid} is a valid key:

$c^{'}\ /\ e(wsk^{1}_{gid},\, wsk^{2}_{gid})$

$= ce(g,\, g)^{y\prod_{gid_x \in \alpha^2}(gid_x+\beta)} / e(\ g^{\frac{r_i y(\prod_{gid_x \in \alpha^2}(gid_x+\beta))}{gid+\beta}},\, g^{\frac{(gid+\beta)}{r_i}})$

$= ce(g,\, g)^{y\prod_{gid_x \in \alpha^2}(gid_x+\beta)} / e(g,\, g)^{y\prod_{gid_x \in \alpha^2}(gid_x+\beta)}$

$= c$

We want to point out a couple of details:

1. Every $c^{'}$ is tagged with an accumulator external identifier (described in $ReEnc()$). $c^{'}$ is selected by the calling process based on a given external identifier. The details will be specified in the system construction section.

2. Although some notations, such as y versus y_i, $c^{'}$ versus $c^{'}_{i}$, or gid versus gid_i, look similar, the notation without subscript specifies individual entities, and the notation with subscript indicates members inside an entity set. Therefore they may refer to the same value in algorithms.

– return c. (Note: an incorrect or random string or value is returned if wsk_{gid} cannot decrypt $c^{'}$.)

- $Dec_{abe}(c,\, \cdot)$: This is the ABE decryption algorithm. It takes the ciphertext c and the remaining parameters denoted as $\cdot$ as inputs. A user's ABE private key usk_{abe} is assumed to be a part of $\cdot$. It outputs m if usk_{abe} can decrypt c. Otherwise the output depends on the output of Dec_{abe}.

5.4.2 User Revocation via Ciphertext Re-Encryption (UR-CRE)

Our first approach is to achieve user revocation through ciphertext re-encryption.

5.4.2.1 Trust Model

We assume that the storage system consists of four entities:

1. *Attribute Authority* (AA): This is the AA of an ABE scheme. It is trusted to generate ABE private keys (or attribute key shares) for users, publish the ABE public key pk_{abe}, and protect the ABE master secret mk_{abe}.
2. *Data Owners* (DO): DO is a trusted party who is responsible for data and user privacy protection. DO initializes the system, defines data access policy, and centrally manages user access rights to the encrypted data in cloud storage.
3. *Cloud Storage Provider* (CSP): CSPs provide cloud storage for a DO to store data and fulfill user requests. It is considered to be untrusted by DOs.
4. *Users*: Users can decrypt data only if they are eligible and have attributes complying with the access control policy of an ABE scheme. Users are considered to be untrusted by DOs.

We assume that, where necessary, communications between entities in our system are protected via suitable secure communication mechanisms, such as using SSL/TLS. We also assume that

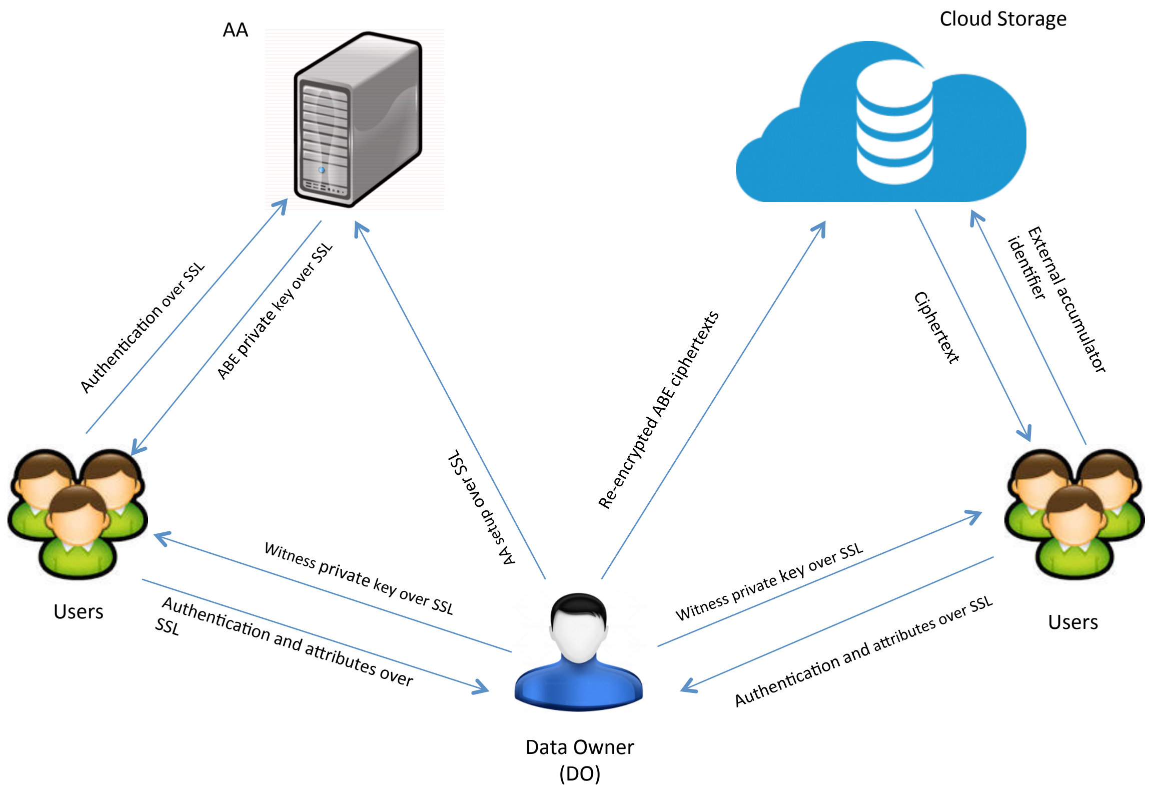

AA
Cloud Storage
Authentication over SSL
ABE private key over SSL
AA setup over SSL
Re-encrypted ABE ciphertexts
Ciphertext
External accumulator identifier
Users
Witness private key over SSL
Authentication and attributes over SSL
Data Owner (DO)
Witness private key over SSL
Authentication and attributes over SSL
Users

Figure 5.1: Trust Model of User Revocation via Ciphertext Re-encryption for ABE in Cloud Storage

CSPs and users do not collude with each other during data retrieval process.

We assume that each user is assigned with a unique *gid* that is not linked to their true identity. To simplify the description of our system, we do not elaborate the user verification process. There are multiple ways to verify whether a *gid* belongs to a user or not.

5.4.2.2 System Description

Figure 5.2 shows the interactions among a user, user groups, a DO, AA, and a CSP. The ABE scheme might have multiple attribute authorities. However, since ABE schemes are treated as black boxes, the AA in our description is the generic.

The system works as follows:

1. System setup. This is a private process run by a DO. The process takes in a security parameter 1^n, and initializes system parameters, an ABE scheme and DA. It then adds eligible (non-revoked) users to accumulators, issues and sends witness private keys to users, encrypts data using ABE, and re-encrypts ABE ciphertext c for each accumulator.

   The outputs of the process are the following:

   - The ABE public key pk_{abe} and master secret mk_{abe} are sent to AA. pk_{abe} is public. mk_{abe} needs to be kept secretly.
   - An witness private key wsk_{gid} is sent to each user and kept secretly.
   - The re-encrypted ciphertext C' is sent to CSP and kept public.

   The detailed steps are as follows:

   - DO calls $Setup(1^n)$ to initiate an ABE scheme and DA. The algorithm returns the ABE public key pk_{abe}, master secret mk_{abe}, and DA's system parameters $\{k, \beta, A_k, \Phi\}$. DA's system parameters are kept secretly by DO.

- DO sends AA the public key pk_{abe} and master secret mk_{abe}.
- DO creates accumulators and issues witness private keys:
  - Let U contain the existing non-revoked *gid*s.
  - Let $WitnessSet = \{\}$ be the temporary data structure for the returned witnesses of an accumulator.
  - For each $gid_i \in U$, DO adds gid_i to an accumulator and generates its witness:
    - Call $UserManager(gid_i,$ "*Add*", $\Phi)$ and get $\{\omega_\alpha, \alpha\}$ as the output;
    - Check if α has been added to $WitnessSet$ or not by using the internal index α[3].
      - If α exists, then replace the existing $\{\omega_\alpha, \alpha\}$ with the new one;
      - Else $WitnessSet = WitnessSet + \{\omega_\alpha, \alpha\}$.

    Note: This is a one-time bulk process to add users into accumulators. Although each *gid* is newly added to the system, it could be added to an existing accumulator. Therefore, the witnesses of the accumulator need to be updated. An accumulator can be updated k - 1 times until *gid*s in the accumulator reach the maximum number k of the allowed elements k. Since this internal computation takes place on the DO side, no user is aware of it until the next step. This operation can be optimized in the actual implementation.
  - DO issues the witness private keys as follow:

    For each $\{\omega_\alpha, \alpha\} \in WitnessSet$:
    - Call $WitnessKeyGen(\omega_\alpha, \alpha)$ and get $witKeys$;

        * For each $gid_i \in witKeys$, send the witness key wsk_{gid_i} to user gid_i.

- DO calls $Enc_{abe}(m, \cdot)$ to encrypt the message m and gets ciphertext c.
- DO calls $ReEnc(c, \Phi)$ to re-encrypt c and gets $C^{'} = \{\{c_i^{'}, g^{y_i}\}\}_{1 \leq i \leq |\Phi|}$
- Finally DO sends $C^{'}$ to CSP.

2. A new user (gid) requests a witness private key. This process is privately run by DO. If the user is eligible, the process will add the user to an accumulator. All users in the accumulator are updated with new witness private keys. The ABE ciphertext is re-encrypted.

   The outputs of the process are the following:

   - New witness private keys are sent to users of the updated accumulator and kept secretly.
   - The updated ciphertext $C^{'}$ is sent to CSP and kept public.

   The detailed steps are as follows:

   - The user sends a gid to DO.
   - DO authenticates and validates the user:
     - If the user is eligible, DO does the following:
       * Call $UserManager(gid, \text{“}Add\text{”}, \Phi)$ to add gid and gets $\{\omega_\alpha, \alpha\}$ as the output.
       * Call $WitnessKeyGen(\omega_\alpha, \alpha)$ and gets $witKeys$.

        * For each $gid_i \in witKeys$, send the witness key wsk_{gid_i} to user gid_i.
        * Call $ReEnc(c,\ \Phi)$ to re-encrypt c for each $\alpha_i \in \Phi$ and get $C^{'} = \{\{c_i^{'}, g^{y_i}\}\}_{1 \leq i \leq |\Phi|}$

          We re-encrypt the ABE ciphertext c for every accumulator in Φ although only one accumulator has been changed. In this way, CSP simply replaces the previous $C^{'}$ with the newly received $C^{'}$.
        * Send $C^{'}$ to CSP.
    - Otherwise DO returns$\perp$ to the user.

3. A new user (gid) requests an ABE private key. This is a private process run by AA.

   The output of the process is one of the following:

   - An ABE private key usk_{abe} is sent to the user if the user is authenticated.
   - Otherwise the output of AA, which could be $\perp$ as an example.

   The detailed steps are as follows:

   - The user contacts AA to request his ABE private key.
   - AA authenticates the user:
     - If the user can be authenticated, then AA does the following:
       * Call $KeyGen_{abe}(\cdot)$ and generate usk_{abe}.
       * Send usk_{abe} to the user.
     - Otherwise AA returns its output.

4. A user (gid) retrieves a ciphertext from CSP. This process is run publicly by CSP and a user. The user sends CSP the external identifier included in the user's witness private key. CSP locates the re-encrypted ciphertext tagged with the external identifier if the re-encrypted ciphertext exists. Upon receiving the re-encrypted ciphertext, the user decrypts it by calling Dec to get the ABE ciphertext, and Dec_{abe} to decrypt the data.

   The output of the process is one of the following:

   - $\perp$ if the re-encrypted ciphertext is not found.
   - Otherwise, either data m if the user is not revoked and has a valid ABE private key, or the output of Dec_{abe}.

   The detailed steps are as follows:

   - User gid sends the external accumulator identifier wsk_{gid}^{3} (g^y) to CSP.
   - CSP returns the ciphertext $c^{'}$ identified by g^y if $c^{'}$ exists, otherwise $\perp$ is returned.
   - If the return value is not $\perp$:
     - The user calls $Dec(c^{'}, wsk_{gid})$:
       * If wsk_{gid} is a valid witness key of g^y, c is returned.
       * Otherwise a random string is returned.
     - The user calls $Dec_{abe}(c, \cdot)$:
       * If c is in a correct form and usk_{abe} is able to decrypt it, m is returned.
       * Otherwise the output of Dec_{abe} is returned.

5. DO revokes a user (gid). This is a private process run by DO. DO removes the user from the user's accumulator, updates the remaining users with new witness private keys, and re-encrypts ABE ciphertext c to $C^{'}$.

   The outputs of the process are the following:

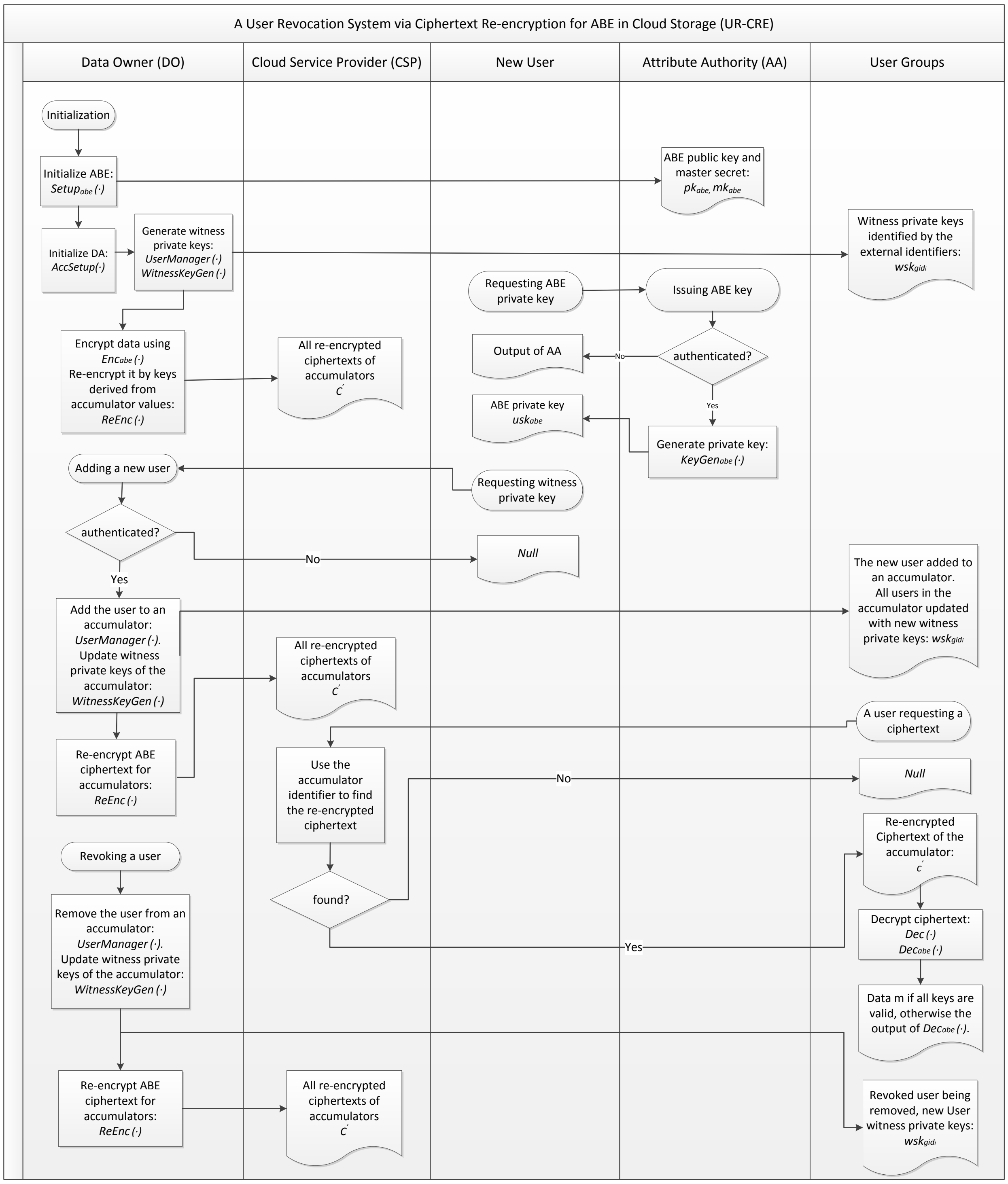
   - New witness private keys is sent to the users of the updated accumulator and kept secretly.
   - The re-encrypted ciphertext $C^{'}$ is sent to CSP and kept public.

   The detailed steps are as follows:

   - DO calls $UserManager(gid,\ \text{“}Delete\text{”},\ \Phi)$ to remove the gid and gets the output $\{\omega_\alpha,\ \alpha\}$.
   - If $\alpha^2 \neq \{\}$, DO does the following:
     - Call $WitnessKeyGen(\omega_\alpha,\ \alpha)$ to generate witness private keys $witKeys$.
     - For $gid_i \in \alpha^2$, send wsk_{gid_i} to user gid_i.
   - DO calls $ReEnc(c,\ \Phi)$ to re-encrypt c and gets $C^{'} = \{\{c_i^{'},\ g^{y_i}\}\}_{1\leq i\leq|\Phi|}$ as the output.
   - DO sends $C^{'}$ to CSP for the update.

5.4.2.3 Security Analysis

The ciphertext re-encryption system (UR-CRE) enables an anonymous user revocation generically for ABE in untrusted cloud storage environments. The system is built on top of an ABE



A User Revocation System via Ciphertext Re-encryption for ABE in Cloud Storage (UR-CRE)
Data Owner (DO)
Cloud Service Provider (CSP)
New User
Attribute Authority (AA)
User Groups
Initialization
Initialize ABE: $Setup_{abe}(\cdot)$
ABE public key and master secret: pk_{abe}, mk_{abe}
Initialize DA: $AccSetup(\cdot)$
Generate witness private keys: $UserManager(\cdot)$ $WitnessKeyGen(\cdot)$
Witness private keys identified by the external identifiers: wsk_{gid_i}
Requesting ABE private key
Issuing ABE key
Encrypt data using $Enc_{abe}(\cdot)$ Re-encrypt it by keys derived from accumulator values: $ReEnc(\cdot)$
All re-encrypted ciphertexts of accumulators c'
Output of AA
No
authenticated?
Yes
ABE private key usk_{abe}
Generate private key: $KeyGen_{abe}(\cdot)$
Adding a new user
Requesting witness private key
authenticated?
No
Null
Yes
Add the user to an accumulator: $UserManager(\cdot)$. Update witness private keys of the accumulator: $WitnessKeyGen(\cdot)$
The new user added to an accumulator. All users in the accumulator updated with new witness private keys: wsk_{gid_i}
All re-encrypted ciphertexts of accumulators c'
A user requesting a ciphertext
Re-encrypt ABE ciphertext for accumulators: $ReEnc(\cdot)$
Use the accumulator identifier to find the re-encrypted ciphertext
No
Null
Re-encrypted Ciphertext of the accumulator: c'
Revoking a user
found?
Decrypt ciphertext: $Dec(\cdot)$ $Dec_{abe}(\cdot)$
Yes
Remove the user from an accumulator: $UserManager(\cdot)$. Update witness private keys of the accumulator: $WitnessKeyGen(\cdot)$
Data m if all keys are valid, otherwise the output of $Dec_{abe}(\cdot)$.
Re-encrypt ABE ciphertext for accumulators: $ReEnc(\cdot)$
All re-encrypted ciphertexts of accumulators c'
Revoked user being removed, new User witness private keys: wsk_{gid_i}

Figure 5.2: Interaction Diagram of User Revocation via Ciphertext Re-encryption for ABE in Cloud Storage

scheme and leverages a dynamic accumulator (DA) based on the scheme of ([15]). Therefore, the fundamental security of the system relies on the security of the selected ABE scheme and DA. We assume that:

- The ABE scheme is at least CPA-secure.
- The DA scheme is secure against the forgeability of witnesses and provides anonymity protection for elements aggregated into accumulators.

The revocation capability aims to equip an ABE scheme with two additional security features:

- A revoked user is not able to access ABE ciphertext immediately after the revocation. However, this protection does not apply to the ciphertext that has been requested or possibly kept by the user locally before the revocation.
- Users can anonymously request re-encrypted ABE ciphertexts from CSPs.

The two security features require the system to have:

1. Dynamic user revocation: Revoked users, having valid ABE private keys, are immediately prevented from being able to access ABE ciphertext.
2. Ciphertext indistinguishability: The re-encrypted ABE ciphertext should remain indistinguishable against eavesdropping attacks.

3. Unforgeability: Users should not be able to forge their witness private keys to decrypt the re-encrypted ABE ciphertexts.

4. Anonymity: Users remain anonymous to CSPs. CSPs are not required for any user management or administration to fulfill data retrieval requests. In addition, the privacy and integrity of ABE access policies (expressed by user attributes) are protected by the re-encryption.

The system security is analyzed as follows:

The Security of Dynamic User Revocation

Dynamic user revocation is realized by preventing the revoked users from accessing ABE ciphertexts.

Each user has two types of private keys: a witness private key for decrypting a re-encrypted ABE ciphertext and an ABE private key for decrypting an ABE ciphertext. Once a user is revoked, AA does not re-issue ABE private keys. Instead, DO takes the following actions to invalidate the user's witness private key:

- First DO removes the user from the user's accumulator. Let us assume that accumulator α_i is the one containing the user. After the user is removed, α_i is updated with the new aggregate value ($\alpha_i^1 = v_i^{'}$). The external identifier g^{y_i} is re-generated as $g^{y_i^{'}}$, where $y_i^{'}$ is randomly selected from Z_p. All accumulator data structures, α and Φ, are kept secret by

DO. y_i and $y_i^{'}$ are not provided to users and CSPs. Only the external identifiers (in the form of g^y) are public and given to users and CSP.

- Then DO generates new witness private keys to the remaining users in α_i. As being described in the algorithm $WitnessKeyGen$ (in Section 5.4.1), each user's witness private key is derived from the user's new witness of α_i by randomizing it with $y_i^{'}$ and $r^{'}$ which is also randomly and uniquely selected from Z_p to each user every time.

  As the revoked user is removed from α_i and does not belong to any other accumulator, the user is not updated with a new witness key. The user is not able to forge a witness of any accumulator based on the unforgeability of DA. All numbers y_i are kept privately by DO. Based on the DLP assumption, the revoked user is not able to discover $y_i^{'}$ even though the user can obtain $g^{y_i^{'}}$. Without the needed components: $v_i^{'}$ (α_i^1) and $y_i^{'}$, the revoked user is not able to forge a valid witness private key once being revoked.

- The DO re-encrypts the ABE ciphertext with the new accumulator value $v_i^{'}$ and the external identifier $y_i^{'}$: $c_i^{'} = ce(g^{y_i^{'}}, v_i^{'})$. Although the revoked user is not able to forge the new witness of an accumulator, the user can try to forge the new aggregate value. Let us assume that this user is the only user who has been removed from the accumulator. As the DA's system parameters (such as accumulator trapdoor β and function A_k) are kept secretly by DO, the user needs to get the previous v_i from his witness private key obtained before his revocation. Assuming the witness private key is: wsk_{gid}= $\{wsk_{gid}^1 = (w_{gid}^1)^r = g^{\frac{ry_i(\prod_{gid_x \in \alpha_i^2}(gid_x+\beta))}{gid+\beta}}$, $wsk_{gid}^2 = (w_{gid})^{1/r} = g^{\frac{(gid+\beta)}{r}}$, $wsk_{gid}^3 = (g)^{\alpha^4}$= $g^{y_i}\}$. Under the DLP assumption, the user cannot get $\prod_{gid_x \in \alpha_i^2}(gid_x + \beta)$ to compute $v_i = g^{\prod_{gid_x \in \alpha_i^2}(gid_x+\beta)}$.

Therefore, the user cannot forge v_i', which is $v_i^{\frac{1}{gid+\beta}}$.

Based on the above analysis, the user cannot decrypt any re-encrypted ABE ciphertext, since without the witness private key, the original ABE private key is insufficient for decryption.

It is possible that a non-revoked user has a valid witness private key, but the attributes fail to comply with the ABE's access policy. This might happen because the witness private keys are issued and used separately from the keys used in ABE. In this case, the security of the ABE scheme should prevent the user from decrypting the ciphertext.

The Security of Ciphertext Indistinguishability

The system should resist eavesdropping adversaries.

The communication channels between DO and CSPs, or users and CSPs, might be unprotected. An adversary can eavesdrop on ciphertexts sent in unsecured channels. Assume that an adversary is able to get the ABE public (encryption) key and the re-encrypted ciphertexts from unencrypted channels. If the adapted ABE is CPA-secure, ABE ciphertext is indistinguishable against eavesdropping attacks. As the re-encryption keys (in the form of $e(g^y, v)$) are randomized (by uniformly selected y) for each accumulator as well as changed for every subsequent ciphertext re-encryption, intuitively the re-encrypted ciphertext is also indistinguishable.

The Security of Unforgeability

The system prevents a revoked user from forging the new ciphertext re-encryption key based on the witness private key that the user acquired before the revocation. This security is based on the membership unforgeability of the DA (scheme [15]) and the DLP assumption. We use the following scenario to informally analyze this. Let us assume that a user (A) with gid is aggregated in α_i.

1. At the beginning, user A is given his witness key of α_i as: $wsk_{gid}= \{wsk^1_{gid} = (w^1_{gid})^r = g^{\frac{ry_i(\prod_{gid_x\in\alpha_i^2}(gid_x+\beta))}{gid+\beta}}$, $wsk^2_{gid} = (w_{gid})^{1/r} = g^{\frac{(gid+\beta)}{r}}$, $wsk^3_{gid} = (g)^{\alpha^4}= g^{y_i}\}$.

2. When A is revoked, DO removes gid from α_i and $y_i^{'}$ is randomly selected. The accumulator α_i has the new external identifier $g^{y_i^{'}}$ and the new aggregate value $v_i^{'}$. DO then refreshes the witness private keys of the remaining members in α_i^2 (which does not include the revoked gid any more) .

3. A is free to retrieve any number of re-encrypted ciphertexts from CSPs within polynomial time. As A is not a member of any accumulator, A is not able to forge any witness of an accumulator based on the security of DA. A is also not able to extract v_i from his previous witness private key, as explained in the security of dynamic user revocation analysis (in Section 5.4.2.3). Therefore A is not able to reconstruct any re-encryption key of the ciphertext even though $g^{y_i^{'}}$ might be obtained by A.

The Security of Anonymity

The anonymity protection is for users being anonymous to CSPs. Users identities are protected in the following ways:

1. Each user is assigned a pseudo identifier *gid* that is unique and not linked to a real identity.
2. An accumulator's aggregate value (v) does not reveal any aggregated *gid*s based on the security of DA ([15]).
3. CSPs are provided with any accumulator value and information other than re-encrypted ciphertexts.

Based on the above analysis, we can conclude that the proposed user revocation system has met the security goals.

Although collusion protection is one of the security requirements of ABE schemes, our proposed system does not consider it. This is because ABE schemes use one master secret to derive user private keys. An encryption and decryption key can consist of multiple portions, each related to a particular attribute. As attributes are shared by users, users without all valid key portions or attributes can pool the key portions to construct a valid private key. Therefore collusion prevention is a critical security requirement of ABE. However, this type of collusion does not apply to our proposed user revocation system, which builds an additional layer above the ABE scheme. There is no attribute sharing between the ABE scheme and the revocation system.

5.4.3 User Revocation via Cloud Storage Providers (UR-CSP)

Although using the ciphertext re-encryption approach UR-CRE enables user revocation at data item level and requires no user management on CSPs, it introduces extra overheads in the following aspects:

- It requires CSPs to store multiple re-encryption copies of an ABE ciphertext.
- It requires a DO to re-encrypt the ABE ciphertext for an accumulator whenever it adds or removes a user.

To eliminate the above overhead, the second generic user revocation system, UR-CSP, leverages CSPs to anonymously identify revoked users during the ciphertext retrieval process. In this way, one copy of the ABE ciphertext needs to be stored. Only non-revoked users can get the ABE ciphertext from CSPs.

5.4.3.1 Trust Model

The trust model still has four entities (see Section 5.4.2.1). The only difference is that CSPs become semi-trusted instead of being completely untrusted. The CSPs are trusted to perform user management in regard to anonymously identifying a user's eligibility to access an ABE ciphertext.

- *Cloud storage provider* (CSP): CSPs store ABE ciphertexts, validate user access rights, and fulfill data retrieval requests. CSPs are semi-trusted entities to execute the assigned tasks, but might be curious to know the data and user identities.

The communication channels between entities in our framework is assumed to be secure, as showed in Figure 5.3. We also assume that CSPs and users do not collude with one another.

Each user is assigned a global unique identifier *gid* that is not linked to the user's true identity. We do not elaborate the verification process of identifying whether a *gid* belongs to a user or not. There are multiple ways of verifying a user's identity against the given *gid*, such as certifications.

5.4.3.2 System Description

This second user revocation system, UR-CSP, still uses DA to manage user access rights as in UR-CRE, except that users are given witnesses instead of witness private keys. Each witness is used to prove a user (*gid*) being aggregated to an accumulator. If CSPs can validate a user's claim to an accumulator, the requested ABE ciphertext is returned.

The external accumulator identifiers are not tagged to any user witness and accumulators' aggregate values. Instead, α^4 (the randomly selected number when α gets created or updated) is used to randomize the witnesses that are sent to users, and aggregate values (α^1) that are sent to CSP. In this way, the aggregate values and witnesses get further protection from CSPs

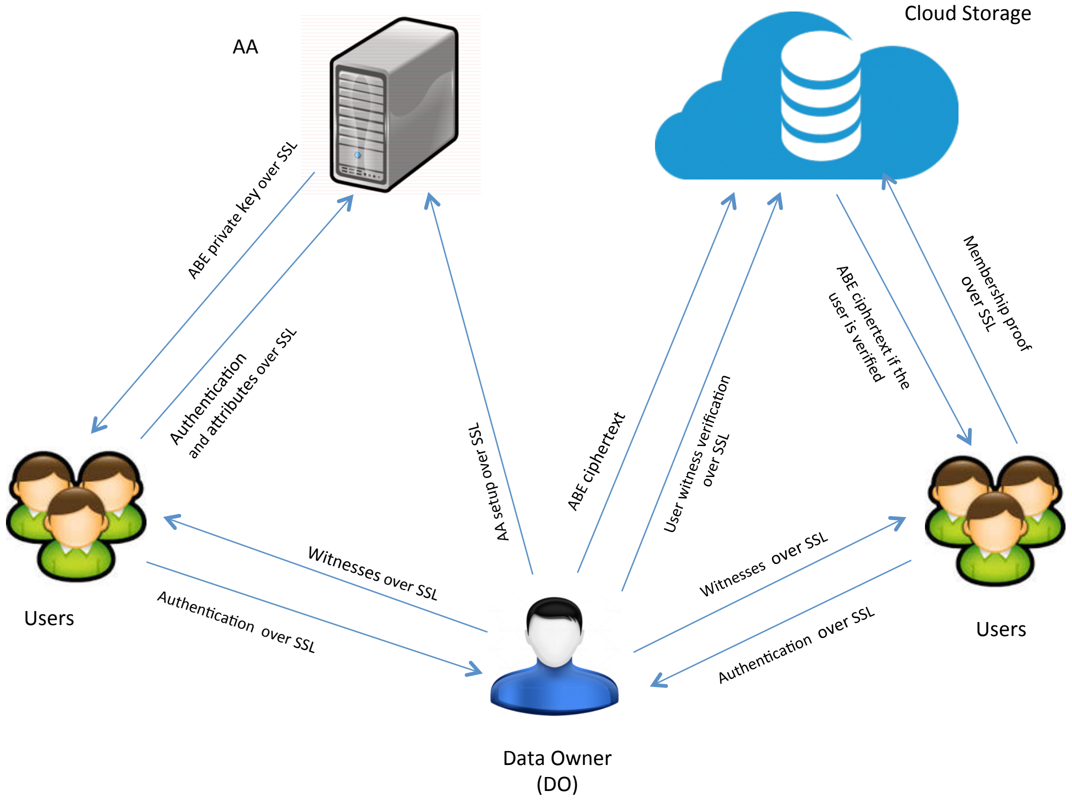


Figure 5.3: Trust Model of User Revocation via Cloud Service Providers for ABE

and users.

One way for a user to attest the membership of an accumulator is to send his/her witness directly to a CSP. Although each witness does not directly link to a user's identity, it does identify the individual user and the user's access privilege to a ciphertext. Since CSP is not fully trusted, directly sharing a witness with CSP could have potential risks, such as impersonation attacks launched by the malicious insiders of CSP. Therefore, the witnesses are considered to be private and kept secret by users.

Inspired by zero knowledge proofs [27], our approach is to make users compute witness proofs. Each time a user requests the ABE ciphertext, the user computes a witness proof using his/her witness. The witness proof is randomized each time, even on the same witness. The CSP is able to validate the proof against the accumulators' aggregate values provided by DO. Figure 5.4 shows the interactions between different parties in the system.

The details of the system construction are as follows:

1. System setup. This is a private process run by DO. It initializes the system by setting up system parameters, an ABE scheme, and DA. It then adds non-revoked users into accumulators, issues and sends witnesses, encrypts data using ABE, randomizes accumulators' aggregate values, and sends the ABE ciphertexts and randomized aggregate values to CSPs.

   The outputs of the process are the following:

- ABE pk_{abe} and mk_{abe} are sent to AA. pk_{abe} is public. mk_{abe} is kept secretly.
- Witnesses w_{gid} is sent to eligible users to be kept secretly.
- The ABE ciphertext c and $\mathit{\Lambda}$ (the container described below for storing randomized aggregate values) are sent to CSP. Values c and $\mathit{\Lambda}$ can be public.

The detailed steps are as follows:

- DO calls Setup(1^n) to initialize an ABE scheme and DA. The algorithm returns the ABE public key pk_{abe}, master secret mk_{abe}, and DA's system parameters $\{k$, β, A_k, $\Phi\}$. DA's system parameters are kept secretly by DO.
- DO provides AA with the public key pk_{abe} and master secret mk_{abe}.
- Let U contain the existing non-revoked gids.
  - Let $WitnessSet = \{\}$ be the container to temporarily store the returned witnesses of accumulators.
  - For each $gid_i \in U$, DO proceeds as the follows:
    - Call $UserManager(gid_i$, "*Add*", $\Phi)$ to add gid_i to an accumulator and get $\{\omega_\alpha, \alpha\}$ as the output;
    - Check whether α has been added to $WitnessSet$ or not by using the internal index α^3. If α exists, remove the existing $\{\omega_\alpha, \alpha\}$.
    - Update $WitnessSet$: $WitnessSet \leftarrow WitnessSet + \{\omega_\alpha, \alpha\}$.

  Note: This again is a one-time bulk process for adding all users into accumulators.

  - For each $\{\omega_\alpha, \alpha\} \in WitnessSet$:
    * For each $gid_i \in \alpha^2$, DO re-computes $w_{gid_i} = \{(w^1_{gid_i})^{\alpha^4} = (w^1_{gid})^y, w^2_{gid_i}\}$, and sends w_{gid_i} to user gid_i.

- DO calls $Enc_{abe}(m, \cdot)$ to encrypt the message m and get ABE ciphertext c.
- Let $\mathit{\Lambda}$ be the container to store all the current aggregate values.
  - For each accumulator $\alpha_i \in \Phi$:
    * DO adds v_i to $\mathit{\Lambda} = \mathit{\Lambda} + \{v_i^{y_i}\}$, where v_i is α_i^1 and y_i is α_i^4.

  We use y_i to randomize aggregate values.
- DO sends $\{\mathit{\Lambda}, c\}$ to CSP.

2. DO adds a new user (gid) to the system. This process is run by DO privately. It adds the user to an accumulator and updates all users in the accumulator with new witnesses. It then updates CSP with the new randomized aggregate values.

   The outputs of the process are as follows:

   - New witnesses of the updated accumulator are sent to users to be kept secretly.
   - Updated Λ is sent to CSP. It can be public.

   The detailed steps are as follows:

   - DO first validates and authenticates the user gid.
   - If the user is eligible:

  - DO calls $UserManager(gid,$ *"Add"*, $\Phi)$ to add gid to an accumulator and has $\{\omega_\alpha, \alpha\}$ returned.
  - For each $gid_i \in \alpha^2$, DO gets y from α^4, re-computes $w_{gid_i} = \{(w^1_{gid_i})^y, w^2_{gid_i}\}$, and sends the witness w_{gid_i} to the user.
  - Let $\Lambda = \{\}$.
  - For each $\alpha_i \in \Phi$, DO adds v_i to Λ: $\Lambda = \Lambda + \{v_i^{y_i}\}$, where v_i is α_i^1 and y_i is α_i^4.

  Note: The updated α has been included in Φ. Therefore, all the aggregate values are currently in Φ. Although only one accumulator is updated, Λ is re-generated for the CSP to simply replace the previous one completely.

  - DO sends Λ to CSP to replace the previous one.

- If the user is not eligible, DO returns $\perp$.

3. A new user (gid) requests an ABE private key. This process is the same as described in UR-CRE.

   The output of the process is one of the following:

   - The ABE private key usk_{abe} is kept secretly by the user if the user is authenticated.
   - Otherwise the output is the output of AA, which could be $\perp$ as an example.

   The detailed steps are as follows:

   - The user contacts AA with his/her attributes.
   - AA verifies the user and his/her attributes.

- If the user is successfully verified, AA calls $KeyGen_{abe}(\cdot)$ and sends the user usk_{abe}.
- Otherwise AA outputs its output.

4. A user gid requests an ABE ciphertext. This process consists of the following communications between CSP and the user :

   - First the user computes a witness proof and sends it to CSP.
   - Upon getting the request and the proof, CSP checks the proof against the accumulator's aggregate values.
   - If the proof can be verified, CSP sends ciphertext c to the user. Otherwise CSP returns $\perp$.
   - If the user gets c back, he/she calls Dec_{abe} to decrypt the data.

   The output of this process is one of the following:

   - Data m, if the user can prove his/her membership and has a valid usk_{abe}.
   - The output of Dec_{abe}, if the user can prove his/her membership, but does not have a valid usk_{abe}.
   - $\perp$, if the user cannot prove his/her membership of an accumulator.

   The detailed steps are as follows:

   - User gid computes a proof:
     - Randomly select γ uniformly from Z_p, and compute:
       $atte = e(w^1_{gid}, (w^2_{gid})^{\gamma}) = e(v^{\frac{y}{gid+\beta}}, g^{\gamma(gid+\beta)}) = e(g^{\prod_{gid_x \in \alpha^2}(gid_x+\beta)}, g^{\gamma y})$

$= e(g,\, g)^{ry \prod_{gid_x \in \alpha^2}(gid_x+\beta)}$

- Compute g^{γ};

- User gid sends $\{atte,\, g^{\gamma}\}$ to CSP;
- CSP validates $atte$:
  - For each $\{v_i^{y_i}\} \in \Lambda$:
    - Compute $e(v_i^{y_i},\, g^{\gamma}) = e(v_i,\, g^{y_i\gamma}) = e(g,\, g)^{ry_i \prod_{gid_x \in \alpha_i^2}(gid_x+\beta)}$.
    - Check whether atte $\overset{?}{=} e(g,\, g)^{ry_i \prod_{gid_x \in \alpha_i^2}(gid_x+\beta)}$ or not;
  - If one match found, ABE ciphertext c is returned;
  - Otherwise $\perp$ is returned.
- If c is returned, the user calls $Dec_{abe}(c,\, \cdot)$ to get m if sk_{abe} is able to decrypt c.
- Otherwise the output depends on the output of $Dec_{abe}(c,\, \cdot)$.

5. DO revokes a user gid. This process is run by DO privately. It removes the user from an accumulator and updates the remaining users with new witnesses. CSP is also provided with the updates.

   The outputs of the process are the following:

   - New witnesses are sent users to be kept secretly, if the updated accumulator α has the remaining users.
   - Updated Λ is sent to CSP and can be public.

   The detailed steps are as follows:

- DO calls $UserManager(gid,$ "$Delete$", $\Phi)$ to remove gid from an accumulator and has $\{\omega_\alpha,\ \alpha\}$ returned.
- If $\alpha^2 \neq \{\}$:
  - For each $gid_i \in \alpha^2$, DO gets the random number y from α^4, re-computes w_{gid_i} $= \{(w^1_{gid_i})^y,\ w^2_{gid_i}\}$, and sends the witness w_{gid_i} to the user gid_i.
- Let $\Lambda = \{\}$.
- For each $\alpha_i \in \Phi$, DO computes $v_i^{y_i}$ and adds it to Λ: $\Lambda = \Lambda + \{v_i^{y_i}\}$, where v_i is α_i^1 and y_i is α_i^4.
- DO sends Λ to CSP to replace the previous one.

5.4.3.3 Security Analysis

As UR-CSP is still built on top of an ABE scheme and leverages DA to achieve dynamic and anonymous user revocation, the fundamental security of UR-CSP still relies on the security of ABE and DA. We assume that:

- The selected ABE scheme is at least CPA-secure.
- DA, which is build out of the scheme in [15], is secure against witness forgeability and provides anonymity of elements aggregated into accumulators.

UR-CSP aims to achieve similar goals to the previous system, UR-CRE:

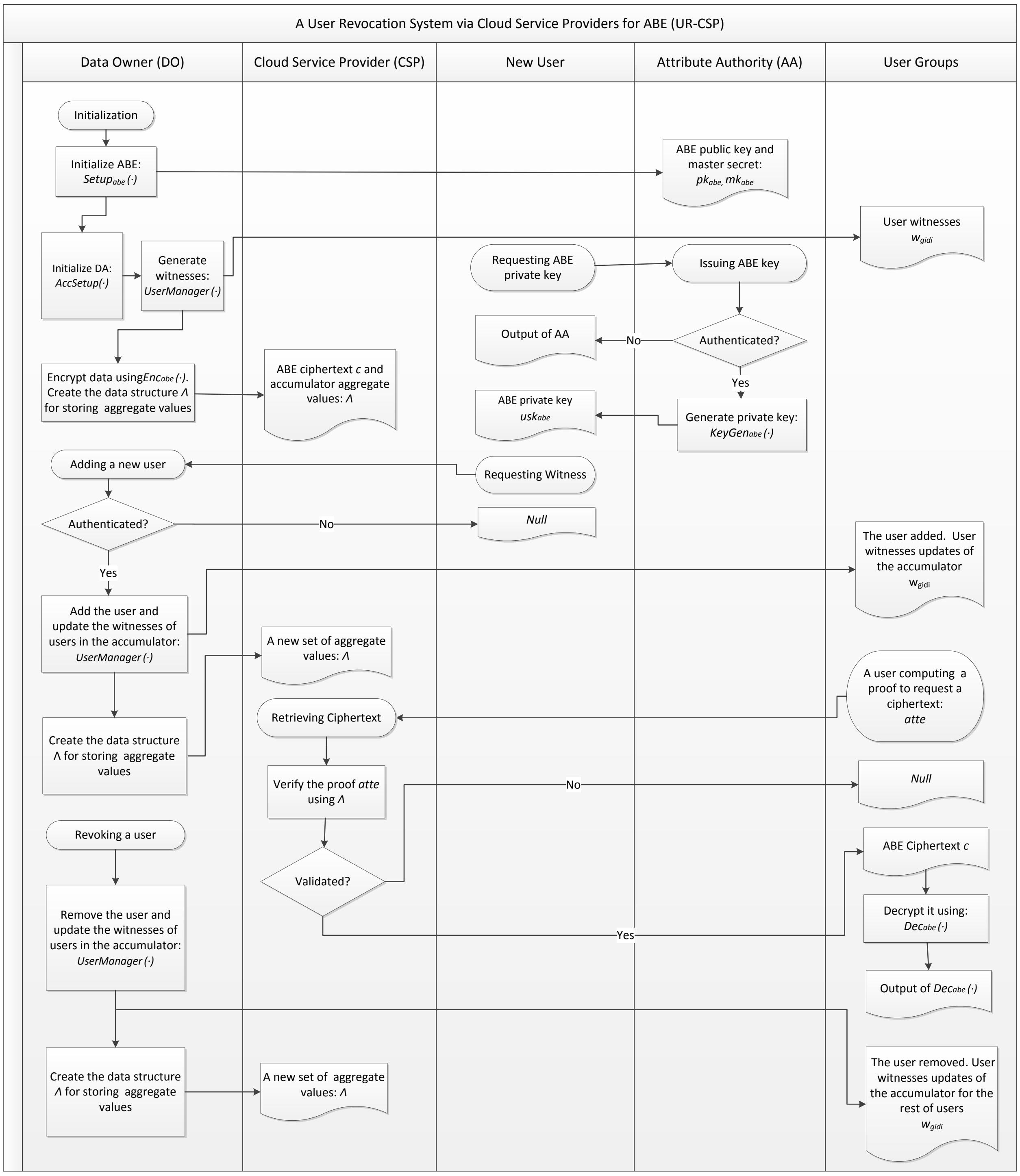

A User Revocation System via Cloud Service Providers for ABE (UR-CSP)
Data Owner (DO)
Cloud Service Provider (CSP)
New User
Attribute Authority (AA)
User Groups
Initialization
Initialize ABE: Setup_abe (·)
ABE public key and master secret: pk_abe, mk_abe
User witnesses w_gidi
Initialize DA: AccSetup(·)
Generate witnesses: UserManager (·)
Requesting ABE private key
Issuing ABE key
Output of AA
No
Authenticated?
Yes
Encrypt data using Enc_abe (·). Create the data structure Λ for storing aggregate values
ABE ciphertext c and accumulator aggregate values: Λ
ABE private key usk_abe
Generate private key: KeyGen_abe (·)
Adding a new user
Requesting Witness
Authenticated?
No
Null
Yes
The user added. User witnesses updates of the accumulator w_gidi
Add the user and update the witnesses of users in the accumulator: UserManager (·)
A new set of aggregate values: Λ
A user computing a proof to request a ciphertext: atte
Retrieving Ciphertext
Create the data structure Λ for storing aggregate values
Verify the proof atte using Λ
No
Null
Revoking a user
ABE Ciphertext c
Validated?
Decrypt it using: Dec_abe (·)
Yes
Remove the user and update the witnesses of users in the accumulator: UserManager (·)
Output of Dec_abe (·)
Create the data structure Λ for storing aggregate values
A new set of aggregate values: Λ
The user removed. User witnesses updates of the accumulator for the rest of users w_gidi

Figure 5.4: Interaction Diagram of User Revocation via Cloud Service Providers for ABE

- A revoked user is unable to decrypt ABE ciphertexts after the revocation.
- CSPs can anonymously identify revoked users during the ciphertext retrieval process.

These goals require the same security as for UR-CRE:

- Dynamic user revocation: revoked users, having valid ABE private keys, are prevented from accessing any ABE ciphertext after the revocations.
- Ciphertext indistinguishability: The ABE ciphertext should remain indistinguishable against eavesdropping attacks.
- Unforgeability: users should not be able to forge their witnesses to prove the membership of an accumulator.
- Anonymity: users remain anonymous to CSPs in any user identification and data retrieval request.

The detailed analysis is as follows:

The Security of Dynamic User Revocation

Assume a revoked user still holds a valid ABE private key for decryption, the system prevents the user from accessing the ABE ciphertext as follows:

- When the user (gid) is revoked, DO removes the gid from its accumulator. Suppose α is the accumulator. The external randomizing number α^4 is re-selected and the aggregate value α^1 is recalculated. Let $y^{'}$ and $v^{'}$ denote the updated values of α^4 and α^1. Each of the remaining users $gid_i \in \alpha^2$ receives a new witness $w^{'}_{gid_i}$ randomized by $y^{'}$:

  $w^{'}_{gid_i} = \{w^{'1}_{gid_i} = v^{'\frac{y^{'}}{gid_i+\beta}}, w^{'2}_{gid_i} = g^{gid_i+\beta}\}$

  The CSP is also updated with a new Λ (containing $(v^{'})^{y^{'}}$).

- The revoked user (gid) is not given any new or updated witness. Suppose that the revoked user has the "old" witness and is denoted as follows:

  $w_{gid} = \{w^1_{gid} = v^{\frac{y}{gid_i+\beta}} = g^{\frac{y(\prod_{gid_i \in \alpha}(gid_i+\beta))}{gid+\beta}}, w^2_{gid} = g^{gid+\beta}\}$, where y and v are the "old" values.

  – Based on the DLP assumption, the user cannot extract $\frac{y\prod_{gid_x \in \alpha}(gid_x+\beta)}{gid+\beta}$ and $gid+\beta$ from his/her witness.

  – Suppose the user compute $e(w^1_{gid}, w^2_{gid})$ to get $e(g,g)^{y\prod_{gid_x \subseteq \alpha}(gid_x+\beta)}$. $y\prod_{gid_x \in \prod \alpha}(gid_x+\beta)$ cannot be extracted based on the DLP assumption, nor the original aggregate value v. Since DA's system parameters and data structures are kept secretly from users, the user is prevented from computing $v^{'}$.

  – Based on the witness unforgebility of DA, the user cannot forge a witness of any accumulator.

  – Suppose that the user assumes him/her to be the only person removed from the accumulator. The user tries to forge a proof ($atte$) by randomly select $\gamma^{'}$ and only uses w^1_{gid} as follows: $atte = e(w^1_{gid}, (g)^{\gamma'}) = e(g, g)^{r' y \prod_{gid_x \in \alpha}(gid_x + \beta)}$. Since y has been changed to y', which is randomly selected and kept secretly by DO, the *atte* cannot be validated correctly.

Based on the above analysis, the revoked user cannot generate a valid proof to CSP. He/She is not given any ABE ciphertext after the revocation.

The Security of Ciphertext Indistinguishability

Since UR-CSP uses an ABE scheme for data encryption, we assume that the security of ciphertext indistinguishability is provided by the ABE scheme selected by DO.

The Security of Unforgeability:

This security is based on the membership unforgeability of the scheme in [15]. Any user, whether revoked or not, cannot forge a valid witness of an accumulator that does not have the user's *gid* aggregated in it.

The Security of Anonymity:

The user verification process does not need user identities or *gid*s:

- Users use witnesses to compute a claim to prove their memberships of a particular accumulator:

$atte = e(w^1_{gid}, (w^2_{gid})^\gamma) = e(v^{\frac{y}{gid+\beta}}, g^{\gamma(gid+\beta)}) = e(g, g)^{ry\prod_{gid_x \in \alpha^2}(gid_x+\beta)}$.

γ is randomly chosen each time so that *atte* is differently formed even using the same witness.

- Using the given aggregate values $\Lambda = \{\{v_i^{y_i}\}_{1 \le i \le |\Phi|}\}$, CSP verify a claim by checking the following:

  For each $v_i^{y_i} \in \mathit{\Lambda}$:

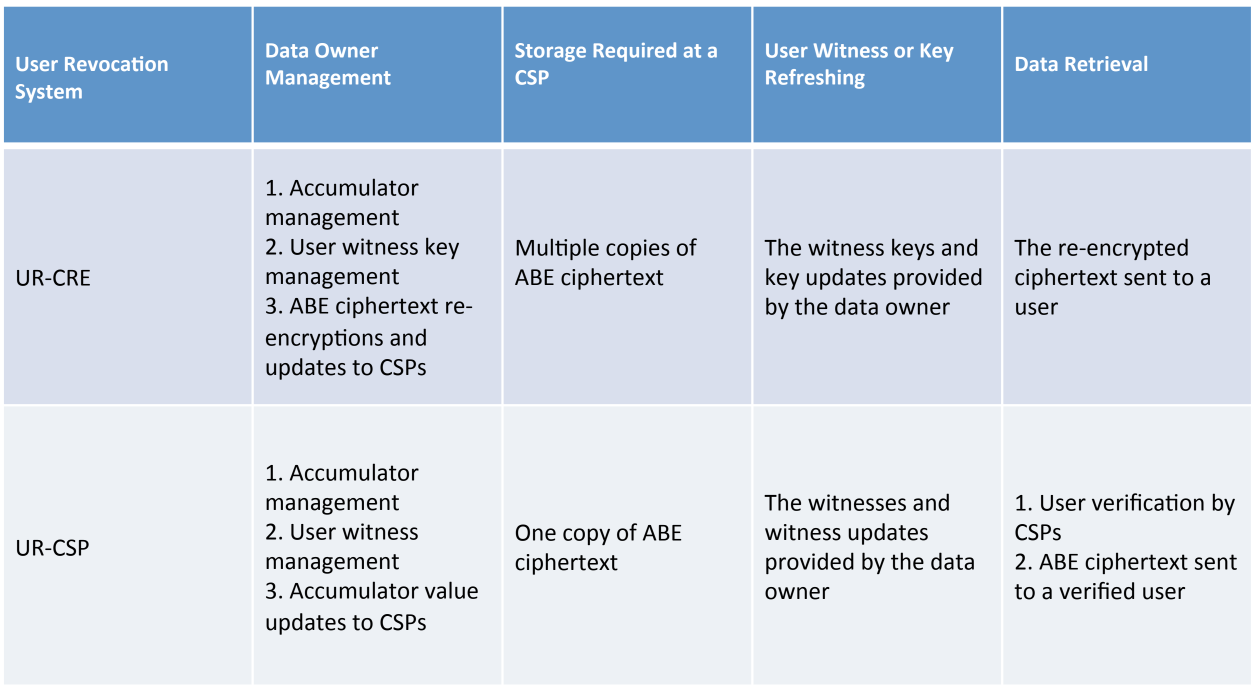
  - Compute $e(v_i^{y_i}, g^\gamma) = e(g, g)^{ry_i \prod_{gid_x \in \alpha_i^2}(gid_x+\beta)}$.
  - Check whether atte $\overset{?}{=} e(g, g)^{ry_i \prod_{gid_x \in \alpha_i^2}(gid_x+\beta)}$ or not;

None of the above steps and information will disclose user identities to CSP. Thus, users remain anonymously to CSP in the process of user versification and data retrieval.

Based on the above analysis, we can conclude that the second proposed user revocation system, UR-CSP, has met the security requirements.

5.5 User Revocation Overhead Analysis

We are not aware any similar system that can generically work with any existing ABE scheme. Thus, we simply compare our two systems in this section (in Figure 5.5).



User Revocation System	Data Owner Management	Storage Required at a CSP	User Witness or Key Refreshing	Data Retrieval
UR-CRE	1. Accumulator management 2. User witness key management 3. ABE ciphertext re-encryptions and updates to CSPs	Multiple copies of ABE ciphertext	The witness keys and key updates provided by the data owner	The re-encrypted ciphertext sent to a user
UR-CSP	1. Accumulator management 2. User witness management 3. Accumulator value updates to CSPs	One copy of ABE ciphertext	The witnesses and witness updates provided by the data owner	1. User verification by CSPs 2. ABE ciphertext sent to a verified user

Figure 5.5: Overhead Comparison of User Revocation Systems

The user revocation systems add integration and management overheads. As we can see, the overheads are mainly coming from the following aspects:

1. Accumulator management;
2. User witness or witness private key management;
3. Ciphertext re-encryption and re-encryption updates to CSPs;
4. User membership construction and verification;
5. Storage required for storing multiple copies of the re-encrypted ciphertext.

UR-CRE has the overheads 1, 2, 3, and 5. Most of those overheads, (1, 2, 3), are on the data owner side. CSPs are only required to provide extra storage space for multiple copies of the re-encrypted ABE ciphertexts (overhead 5). The data owner can be a potential bottleneck for updates and witness private key management. However, the performance and scalability can be greatly improved by delegating the witness private key updates and ciphertext re-encryption to a trusted party or servers. Transferring updates to users and CSPs can also be out-of-band.

UR-CSP has the overheads 1, 2, and 4. Most of those overheads, (1, 2), are still on the data owner side, but the data owner does not need to re-encrypt ABE ciphertexts. The data owner can still be a bottleneck to manage accumulators and witness updates. Delegation can still be the implementation strategy to reduce the bottleneck. Only one copy of ABE ciphertext is stored at CSPs. However, ciphertext retrieval is a protocol of user membership verification process (overhead 4). It requires a user to construct a proof for a CSP to validate the proof every time. CSPs are required to be semi-trusted to perform user membership verification. This trust is not required in UR-CRE.

Overall, the overheads of UR-CSP appear to be more evenly distributed compared to UR-CRE. It is slightly more scalable than UR-CRE, but UR-CRE can improve the scalability by delegating the tasks and computations.

5.6 Conclusions

In this chapter, we proposed two dynamic user revocation systems for ABE schemes. Our systems are generic and can be applied directly to any ABE scheme. We build user privacy protection into the data retrieval process, making ABE schemes more suitable and practical for deployment in untrusted cloud storage systems.

Chapter 6

A Generic Attribute Revocation System for ABE in Cloud Storage

Contents

User revocation concerns user access rights being revoked from an ABE system. It might be excessive in situations where a user has only a few attributes needing to be revoked, while other attributes continue to satisfy certain access policies. Resolving this situation requires revocation to be more granular, which means that revocation is required at both attribute and user levels, rather than only at the user level.

In ABE, an attribute can be shared by multiple users. Any shared attribute being revoked from a user should not impact the use of the attribute by other users. In other words, attribute revocation should allow any number of attributes to be revoked from one or multiple users without impact on other users who share or use the same attributes for data decryption.

Attribute revocation has been studied in the literature, however most existing revocation schemes only work with certain individual ABE schemes. Some are inflexible in the number of attributes that can be revoked, or not granular enough to support attribute revocation both at user and attribute level (two-level revocation): revoking any number of attributes from a user or revoking any attribute from a number of users. We propose an attribute revocation system that is generic to all ABE schemes, and supports two-level attribute revocation. User privacy is also protected in the revocation system.

6.1 Related Work

Yu et al. [148] proposed a CP-ABE scheme to accomplish revocation of user access rights via attribute revocation. When a user's access right is revoked, the AA generates a new re-encryption key for the semi-trusted on-line proxy server. On behalf of the AA, the proxy server generates and distributes new updated attribute key shares (private keys) to each non-revoked user. Then the proxy server re-encrypts the data with the new encryption key. Although the scheme offloads the data and attribute private key updates to a semi-trusted proxy server, each revocation triggers a round of user attribute key updates and data re-encryption.

Yang et al. [145] proposed a CP-ABE revocation scheme which generates a new attribute public key and private key whenever an attribute is revoked. The difference between this scheme and [148] is that only those components associated with the revoked attribute in secret keys and ciphertexts, instead of all the components in the secret keys and ciphertexts, need to be updated. Users are required to obtain those updates of their private keys. Data is re-encrypted by semi-trusted CSPs using the new attribute public key. While these schemes enable instantaneous user revocation, each revocation still triggers a round of attribute key share updates and ciphertext re-encryption.

Based on Hur and Noh's user revocation system [70], Xie et al. [140] constructed a user and attribute revocation system aimed at reducing the computational overhead of the data service manager in [70]. The new construction in [140] improved the key update computation of the data service manager by half. Despite this efficient new construction, it inevitably inherits the potential issues of service managers discussed in Section 5.1.

Wang et al. [133] proposed attribute revocation for a KP-ABE scheme. In their scheme, each user is associated with two access trees. Each tree has its own private key. The first tree is used for data decryption if the user does not have any revoked attributes. Otherwise the second tree is used as long as the user's valid attributes still satisfy the policy of the KP-ABE scheme. Although this revocation scheme does not require attribute key re-issuing, it only works for revoking one attribute at a time.

6.2 Our Contributions

We propose an attribute revocation system, AR-ABE, for ABE schemes with the following characteristics:

- Generic: The revocation process can directly work with any ABE scheme.
- Dynamic: The revocation is instantaneous without requiring ABE private keys to be re-issued or data to be re-encrypted.
- Granular: The revocation can take place at either attribute level or user level. At attribute level, an attribute can be revoked from a user or a number of users. At the user level, a user can have one or multiple attributes being revoked. Any level of revocation does not impact other users using the same attributes.
- Collusion resistant for re-encryption keys: The revocation system utilizes ciphertext re-encryption to control the access to ABE ciphertexts. Ciphertext re-encryption keys are re-constructed by users with their attribute witnesses. The revocation system (AR-ABE) prevents users from pooling their attribute witnesses to construct any ciphertext re-encryption key. Re-encryption keys are completely independent with regard to ABE private keys.
- Anonymous: Users are anonymous to CSPs.

6.3 The Attribute Revocation System (AR-ABE)

Granular attribute revocation is not as straightforward as user revocation. In ABE schemes, each user's private key is uniquely formed. A special link is typically used to bind the private key shares of a user's attributes together so that users are prevented from pooling their key shares together to construct a decryption key. This type of prevention is referred to as collusion resistance. Revoking an attribute without re-issuing the private key can break the bond among the attribute key shares, and prohibit the user from reconstructing other decryption keys. This makes it difficult to implement a dynamic and granular revocation system.

To make an attribute revocation system dynamic and granular as well as generic to all types of ABE schemes, we separate the revocation control from the underlying ABE schemes. This revocation control is again achieved by controlling access to ABE ciphertexts via ciphertext re-encryption. Re-encryption keys are randomly generated one-time keys. If a user does not have the non-revoked attributes that are required by the ABE policy, the one-time re-encryption key cannot be recovered during the decryption process, even though the user still possesses a valid ABE private key. The proposed system still uses DA (defined in Section 5.3.2) to manage user access to ABE ciphertext. Again, users are uniquely identified by global identifiers (*gid*s) that are not linked to their attributes and identities.

- At the attribute level, each attribute is associated with a set of accumulators aggregated with users (*gid*s) who possess the attribute that has not been revoked. When an attribute is revoked from a user, the user's *gid* will be removed from its accumulator. The rest of users of the accumulator will receive their new witnesses. When a new user is granted access to an ABE ciphertext, the user will be added to an accumulator for each attribute that the user has. Other users in those accumulators will receive new witness updates.

- At the user level, we employ an access tree called an *attribute accumulator tree* (AATree) (see Section 6.2). AATree is constructed by a data owner (DO) with the attributes required by the access policy of an ABE scheme. The tree is provided to CSPs.

  The AATree is used for the following purposes:

  – It contains the necessary attributes (via their accumulators) needed by the selected ABE scheme. Although AATree is inspired by the access tree of ABE schemes, it is completely independent in regard to the content and relationship with the underlying ABE scheme. AATree is not used by any ABE scheme and does not replace any ABE scheme's access policy. It simply contains the necessary attributes that are used by the selected ABE scheme in the decryption process.

  – It is used to control user access to ABE ciphertexts. When a user requests an ABE ciphertext, a CSP randomly generates a one-time re-encryption key, re-encrypts the ABE ciphertext, and embeds the re-encryption key using the AATree in the re-encrypted ciphertext. Only users who have non-revoked attributes with their accumulator witnesses satisfying the AATree are able to re-construct the re-encryption key to get the ABE ciphertext.

  – It is used to protect user privacy. As the AATree and accumulators are managed and updated by data owners, CSPs do not need to know, or keep a list of users with their revoked attributes. A CSP only needs to be posted with the latest AATree and uses it to embed re-encryption keys.

6.3.1 Trust Model

The trust model for AR-ABE is the same as the trust model of the one with UR-CSP described in Section 5.4.3.1. Data owners (DO) and the ABE attribute authority (AA) are trusted, while CSP are semi-trusted, and users are not trusted. Figure 6.1 is the diagram of the trust model.

The communications between DO and users, as well as AA, are protected via suitably secured channels encrypted by SSL/TLS. We also assume that users are properly authenticated before ABE private keys or attribute witnesses are issued.

6.3.2 Algorithm Definition and Construction

We first define the data structures and algorithms used in AR-ABE.

6.3.2.1 Data Structure Definition

There are two main data structures for AATree and accumulators. AATree data structures are used for managing user access to ABE ciphertexts. Accumulator data structures are used to manage user attributes and their revocation statuses.

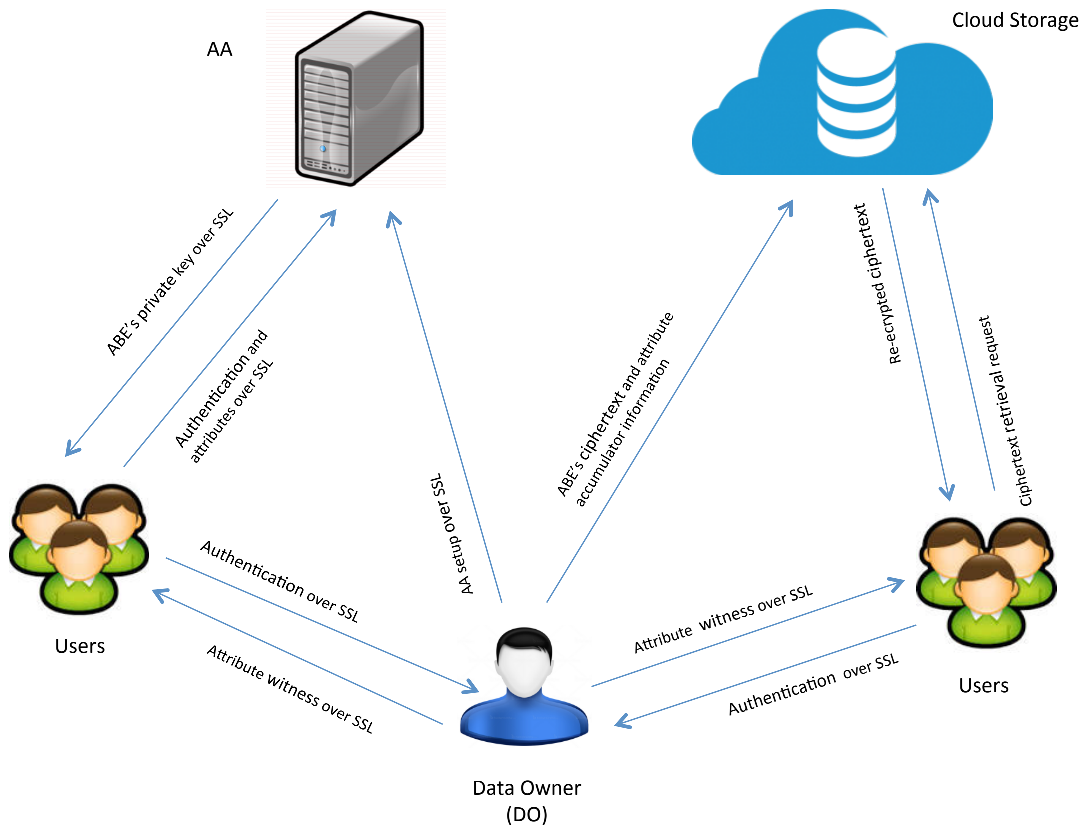

AA
Cloud Storage
ABE's private key over SSL
Authentication and attributes over SSL
AA setup over SSL
ABE's ciphertext and attribute accumulator information
Re-ecrypted ciphertext
Ciphertext retrieval request
Users
Authentication over SSL
Attribute witness over SSL
Attribute witness over SSL
Authentication over SSL
Users
Data Owner (DO)

Figure 6.1: Trust Model of Attribute Revocation System for ABE in Cloud Storage

- Attribute Accumulator Tree (AATree): Figure 6.2 shows an example of an AATree. The internal nodes are attributes that are required by the selected ABE scheme to check for the compliance of its access policy. Each leaf node is one of the accumulators of the attribute. Since each user is only aggregated to one accumulator of an attribute, the internal nodes (attributes) are $1 - out - of - n$ threshold gates. The n denotes the number of accumulators for an attribute, that can be different among attributes. As the number of accumulators for an attribute can be changed for adding or removing users, each n is not static and subject to change throughout its lifetime. AATree is dynamically updated by DO. Figure 6.2 also illustrates how a re-encryption key is split and embedded into the tree.

- Data structures used to manage accumulators: the system and DO use the following data structures to manage accumulators and user attributes.

  - Let $Att = \{att_i\}_{1 \leq i \leq |Att|}$ contain attributes in the system. $|Att|$ denotes the number of attributes in the system.
  - Let $\mathit{\Lambda}_{policy} = \{att_i\}_{1 \leq i \leq \left|\mathit{\Lambda}_{policy}\right|}$ contain the attributes that are used by an ABE access policy. $|\Lambda_{policy}|$ denotes the number of attributes included.
  - Let $UAtt_{gid} = \{att_i\}_{1 \leq i \leq \left|UAtt_{gid}\right|}$ contain the attributes of user *gid*. $|UAtt_{gid}|$ denotes the number of attributes of user *gid*.
  - Let α denote an accumulator as it is defined in Section 5.3.2.1:

    $\alpha = \{\alpha^1 = v, \, \alpha^2 = \{gid_x\}_{1 \leq x \leq |\{gid_x\}|}, \, \alpha^3 = i, \, \alpha^4 = y\}.$

  - Let Φ_{att} denote the attribute accumulator container containing all the accumulators of attribute att:

    $\Phi_{att} = \{\Phi^1_{att} = att,\ \Phi^2_{att} = \Phi = \{\alpha_i\}_{1 \le i \le |\Phi|}\ \}$, where $|\Phi|$ is the number of the accumulators of att.

  - Let $\Omega = \{\Phi_{att_i}\}_{att_i \in Att}$ contain all the attribute accumulator containers.

  - Let $uw_{gid} = \{\{att_i,\ w_{gid},\ g^{y_j}\}\}_{1 \le i \le |uw_{gid}|}$ contains the witnesses of the attributes that user gid has had and not been revoked.

- Data structures used to express AATree: As accumulators are solely managed by a DO, they are kept and stored on the DO side. But information, such as the accumulator's aggregate values and attributes required by ABE access policy, is needed by CSPs to control user access to an ABE ciphertext. Δ is for a DO to provide the information to CSPs. Ψ is created by a CSP to embed re-encryption keys in the ciphertext based on Λ_{policy}.

  - Δ is used to store accumulator aggregate values of attributes in Λ_{policy} :

    $\Delta = \{att_i,\ \{v_j^{y_j},\ g^{y_j}\}_{1 \le j \le |\Phi^2_{att_i}|}\}_{1 \le i \le |\Lambda_{policy}|}$, where v_j is the aggregate value of accumulator α_j for attribute att_i, g^{y_j} is the external identifier of accumulator α_j, and $|\Phi^2_{att_i}|$ $(= |\Phi|)$ is the total number of accumulators for att_i.

  - Ψ is used to carry a ciphertext re-encryption key. An example is like the following:

    $\Psi = \{\{(e(g,g)^{s_i - gid \cdot y_j \prod_{gid_x \in \alpha_j} (gid_x + \beta)},\ g^{y_j})\}_{1 \le j \le |\Phi^2_{att_i}|}\}_{1 \le i \le |\Lambda_{policy}|}$, where s_i is a randomly split share of a randomly selected s at the re-encryption time. Users reconstruct re-encryption keys if their non-revoked attributes have the valid witnesses of those accumulators.

Table 6.1 lists the commonly used notation in this chapter.

6.3.2.2 Algorithm Construction

We define the commonly used algorithms in our system. Whether the algorithms can be run privately or publicly and which information is kept secret depend on the trust model. These will be described in Section 6.3.3.

- $Setup(1^n)$: The algorithm takes the input 1^n as security parameter and proceeds as follows:
  - Let U contain the existing user *gids*: $U = \{gid_1\ , \dots gid_n\}$, where $gid_i \in Z_p^\star$ (p is a n-bit prime).
  - Call $Setup_{abe}(1^n)$ to initialize the ABE scheme using the security parameter 1^n. The algorithm outputs the public key pk_{abe} and the master secret mk_{abe}.
  - Initialize DA using the security parameter 1^n:
    - Let e: $G \times G \rightarrow G_1$ be a bilinear map for p.
    - Let g be a generator of G.
    - Call $AccSetup(G, g)$ (Algorithm 5.1) with G and g as the inputs. It outputs $\{k,$

Notation	Description
c	An ABE ciphertext
AA	The ABE attribute authority
$Setup_{abe}$	The setup algorithm of an ABE scheme
$KeyGen_{abe}$	The key generation algorithm of an ABE scheme
Enc_{abe}	The encryption algorithm of an ABE scheme
Dec_{abe}	The decryption algorithm of an ABE scheme
pk_{abe}	The public key of an ABE scheme
mk_{abe}	The master secret of an ABE scheme
usk_{abe}	An ABE private key of a user
gid	A user's global unique identifier
U	The set containing user *gid*s in the system
CSP	Cloud Storage Providers
DO	Data Owner
Φ	Accumulator container used by Φ_{att} to store the accumulators of attribute *att*
Φ_{att}	Accumulator container to store the accumulators of attribute *att*
Ω	Accumulator container to store all the Φ_{att} in the system
k	The maximum number of elements that can be aggregated to an accumulator
A_{κ}	The accumulator function to generate aggregate values
β	The trapdoor value used by the accumulator function A_k
α	The container of a particular accumulator and its information
v	An accumulator aggregate value
Ops	The operational type - *Add* or *Delete* a *gid* to or from an accumulator
ω_{α}	The witness container of α to store user witnesses
Att	The set of attributes in the system
$UAtt_{gid}$	The set of attributes for user *gid*
Λ_{policy}	The container storing the attributes that may be required by the ABE scheme
uw_{gid}	The container storing attribute witnesses of user *gid*
Δ	The data structure storing accumulator aggregate values of attributes in Λ_{policy}
Ψ	The data structure storing shares of a ciphertext re-encryption key
C'	The container storing the re-encrypted ABE ciphertext

Table 6.1: Attribute Revocation System Notation

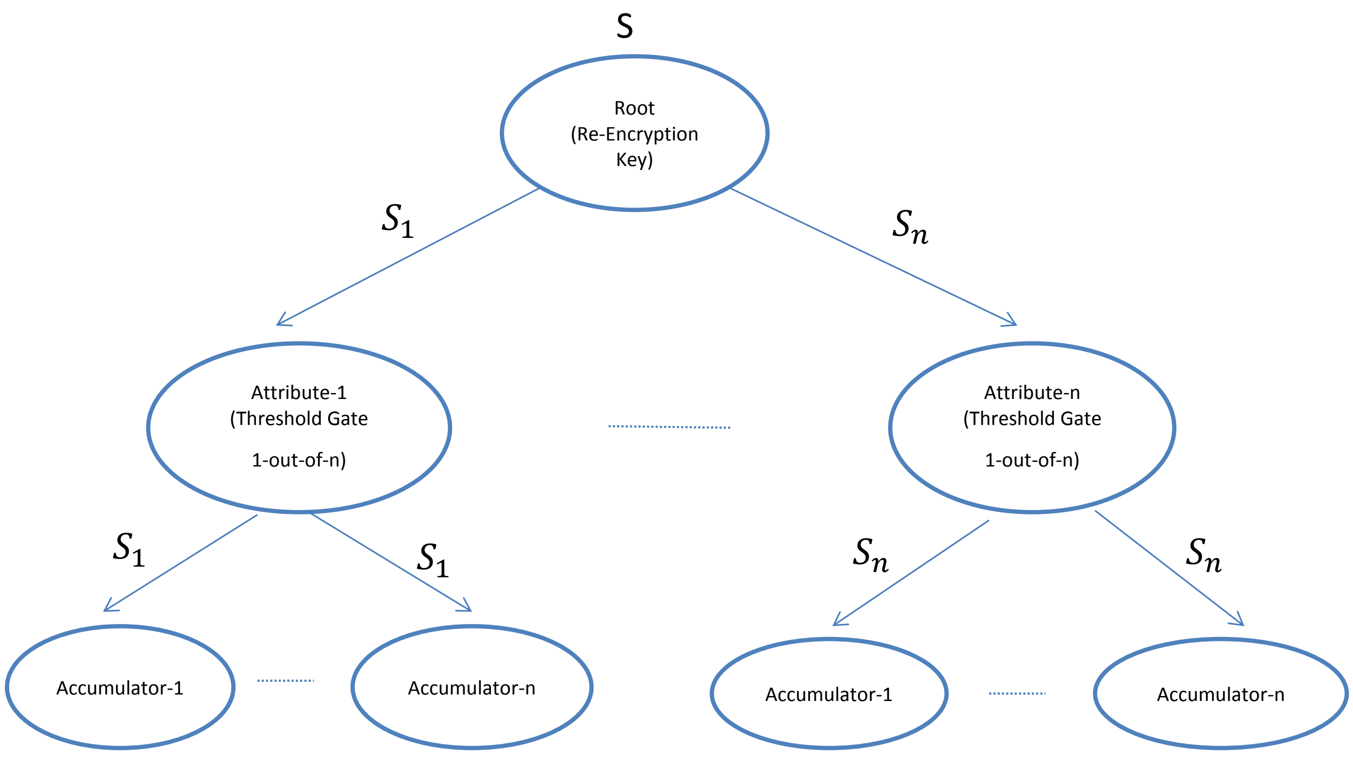


Figure 6.2: Attribute Accumulator Tree

β, A_k, $\Phi\}$, where Φ is the container used by Φ_{att} to store the accumulators for attribute *att*. *att* is a generic notation of an attribute in the system.

- $KeyGen_{abe}(\cdot)$: This is the ABE algorithm to generate user private keys. The $\cdot$ generically denotes the input parameters. It outputs a user private key usk_{abe}.

- $UserAttributeManager(gid, Ops, \Phi_{att})$: This algorithm takes a user's *gid*, the operation type *Ops*, and the accumulator container Φ_{att} of attribute *att* as the inputs. It proceeds as follows:

  - Extract accumulator container Φ: $\Phi = \Phi^2_{att} = \{\alpha_i\}_{1 \le i \le |\Phi^2_{att}|}$.
  - If $Ops ==$ "*Add*", the algorithm calls *AccAdd*(*gid*, Φ) (Algorithm 5.2) to add *gid* to an accumulator: $\{\alpha, \Phi\} = AccAdd(gid, \Phi)$.
  - If $Ops ==$ "*Delete*", the algorithm calls *AccDelete*(*gid,* Φ) (Algorithm 5.3) to remove *gid*:

    $\{\alpha, \Phi\} = AccDelete(gid, \Phi)$.
  - Replace the old Φ in Φ_{att} with the newly returned Φ: $\Phi^2_{att} = \Phi$.
  - Set $w^{'}_{\alpha} = \{\}$.
  - If $\alpha^2 \neq \{\}$, issue witness updates for the users in the accumulator:
    * Select a random number uniformly from Z_p: $y \leftarrow Z_p$, and set $\alpha^4 = y$.
    * Find and replace the old α in Φ^2_{att} (using the internal index α^3).
    * Call $AccWitUpdate(\alpha)$ (Algorithm 5.4) to compute witnesses of users:

      $\omega_\alpha = AccWitUpdate(\alpha)$.

  * Embed gids and accumulator random number α^4 (y) in user witnesses to prevent collusions and witness forgery:
    - For each $gid_i \in \omega_\alpha$: re-compute witness as:
      $w^{'}_{gid_i} = \{w^{'1}_{gid_i} = (w^1_{gid_i})^{gid_i \alpha^4} = g^{\frac{gid_i y (\prod_{gid_x \in \alpha^2} (gid_x + \beta))}{gid_i + \beta}},\ w^2_{gid_i} = g^{(gid_i + \beta)}\}$;
      add $w^{'}_{gid_i}$ to $\omega^{'}_\alpha$: $\omega^{'}_\alpha + \{w^{'}_{gid_i},\ gid_i\}$.

– Finally return $\{w^{'}_\alpha,\ \alpha,\ \Phi_{att}\}$.

- $Enc_{abe}(m,\ \cdot)$: This is the ABE encryption algorithm. It takes the inputs of message m and the rest of parameters denoted as $\cdot$. It outputs ABE ciphertext c.

- $ReEnc(c,\ \Lambda_{policy},\ \Delta,\ gid)$: This algorithm inputs an ABE ciphertext c, the attributes needed by the ABE access policy Λ_{ploicy}, the aggregate values with external identifiers Δ, and a user's gid. It re-encrypts c as follows:

  – Let $\Lambda_{policy} = \{att_i\}_{1 \le i \le |\Lambda_{policy}|}$, and $\Delta = \{\{att_i,\ \{v_j^{y_j},\ g^{y_j}\}\}_{1 \le j \le \left|\Phi^2_{att_i}\right|}\}_{1 \le i \le \left|\Lambda_{policy}\right|}$. (Φ_{att_i} cannot be accessed by this algorithm. We use this denotation to express the total number of accumulators for attribute att_i.)

  – Randomly select $s \leftarrow Z_p$, and re-encrypt c to $c^{'} = ce(g,\ g)^s$.

  – Embed s in Ψ:

    * s is split into $|\Lambda_{policy}|$ number of shares differently and randomly, so that no share is equal to any other shares:
      $s = s_1 + s_2 + \cdots + s_I$, where $I = |\Lambda_{policy}|$.
    * Each share is assigned to an $att_i \in \Lambda_{policy}$.

∗ To prevent collusion, *gid* is embedded into shares. As accumulators' aggregate values do not disclose the aggregated *gid*s, *gid* is blindly embedded in every accumulator. However, this is not a concern. The AATree is $1 - out - of - n$ threshold (described in 6.2). A *gid* can only belong to one accumulator at most. Using the accumulators' external identifiers, correct components can be located for reconstructing the key in the decryption algorithm.

The following is the steps of building Ψ:

· For each $att_i \in \wedge_{policy}$:

Find $\{att_i, \{v_j^{y_j}, g^{y_j}\}_{1 \leq j \leq |\Phi^2_{att_i}|}\}$ in Δ:

For each accumulator of attribute att_i, compute:

$e(g^{s_i}, (v_j^{y_j})^{-gid}) = e(g, g)^{s_i - gidy_j(\prod_{gid_x \in \alpha_j^2}(gid_x + \beta))}$, where j is the jth accumulator α_j for att_i.

· The final Ψ is as the following:

$\Psi = \{\{(e(g, g)^{s_i - gidy_j(\prod_{gid_x \in \alpha_j^2}(gid_x + \beta))}, g^{y_j})\}_{1 \leq j \leq |\Phi^2_{att_i}|}\}_{1 \leq i \leq |\Lambda_{policy}|}$.

– Finally the algorithm outputs $C' = \{c' = ce(g, g)^s, \Psi\}$.

• *Dec*(C', uw_{gid}): This algorithm inputs the re-encrypted ciphertext C', and attribute witness set uw_{gid} for user *gid*. It uses the attribute witnesses in uw_{gid} to recover the re-encryption key, and decrypts C' to get the ABE ciphertext c.

– Let uw_{gid} be expressed as: $\{\{att_i, w_{gid}, g^{y_j}\}\}_{1 \leq i \leq |UAtt_{gid}|}$, where $|UAtt_{gid}|$ is the number of attributes of user *gid*, w_{gid} is the witness of attribute att_i, and g^{y_j} is the external identifier of the accumulator.

– The re-encrypted key can be recovered as follows:

  * For each att_i in uw_{gid}, use the external identifier g^{y_j} to find the re-encryption key share in Ψ. If the share can be found, then proceed as follows:

    · Compute the intermediate value using the witness: $e(w^1_{gid}, w^2_{gid}) = e(g^{\frac{gidy_j(\prod_{gid_x \in \alpha^2_j}(gid_x+\beta))}{gid+\beta}}, g^{(gid+\beta)}) = e(g,g)^{gidy_j(\prod_{gid_x \in \alpha^2_j}(gid_x+\beta))}$.

    · Recover the key share: $e(g,g)^{s_i - gidy_j(\prod_{gid_x \in \alpha^2_j}(gid_x+\beta))} \cdot e(g,g)^{gidy_j(\prod_{gid_x \in \alpha^2_j}(gid_x+\beta))} = e(g,g)^{s_i}$.

  * Reconstruct the re-encryption key if all the shares can be recovered: $\prod_{1 \le i \le |\Lambda_{policy}|} e(g,g)^{s_i} = e(g,g)^s$.

– Decrypt c', which is a part of C', to get the ABE ciphertext: $c = c' \ / \ e(g,g)^s$.

– The algorithm returns c. Note: if the re-encryption key can be successfully constructed, c is a well-formed ABE ciphertext; otherwise c is a random string.

- $Dec_{abe}(c, \cdot)$: This is the ABE decryption algorithm. It takes the ciphertext c and required parameters denoted as $\cdot$ as the inputs. $\cdot$ should include the user's usk_{abe}. It returns data m if usk_{abe} can decrypt c, otherwise the output depends on Dec_{abe}.

6.3.3 System Description

Figure 6.3 illustrates interactions and communications between a DO, AA, a CSP, and users.

1. System setup. This is run by DO privately. It takes in a security parameter 1^n and initializes the system by setting up system parameters, an ABE scheme, and DA. It then

adds users into accumulators, issues and distributes attribute witnesses, encrypts data using ABE, and constructs data structures.

The outputs of the process are the following:

- pk_{abe} and mk_{abe} are sent to AA. pk_{abe} is public. mk_{abe} needs to kept secret.
- Non-revoked attribute witnesses are sent to users. They need to be kept secret.
- The ABE ciphertext c, data structures $\mathit{\Delta}$ and $\mathit{\Lambda}_{policy}$ are sent to CSP. They can be public.

The detailed steps are as follows:

- DO calls $Setup(1^n)$ to initialize an ABE scheme and DA. The algorithm returns the ABE public key pk_{abe}, master secret mk_{abe}, and DA's system parameters $\{k, \beta, A_k, \Phi\}$.
- DO sends AA the ABE public key pk_{abe} and master secret sk_{abe}.
- DO keeps the system parameters of DA privately.
- DO calls $Enc_{abe}(m, \cdot)$ to encrypt the message m to c.
- DO creates the data structure $\mathit{\Lambda}_{policy} = \{att_i\}_{1 \leq i \leq |\mathit{\Lambda}_{policy}|}$ containing all the attributes needed by the access policy of the ABE scheme.
- DO issues attribute witnesses to users:
  - Let U contain the existing gids.
  - For each $gid_j \in U$:

- $*$ Let $UAtt_{gid_j}$ contain all the attributes of gid_j.
- $*$ For each $att_i \in UAtt_{gid_j}$:
  - $\cdot$ If att_i has not been revoked:

    Call $UserAttributeManager(gid_j,$ "*Add*", $\Phi_{att_i})$ and have $\{w_\alpha,\ \alpha,\ \Phi_{att_i}\}$ returned.

    If $\alpha^2 \neq \{\}$, compute the external identifier $g^{\alpha^4} = g^y$; for each $gid_i \in \alpha^2$, send gid_i the witness of att_i:

    $\{att_i,\ w_{gid_i},\ g^y\}$.

    Update Φ_{att_i} in Ω, where Ω is the container containing all the Φ_{att} in the system.

- DO constructs $\Delta = \{att_i,\ \{v_j^{y_j},\ g^{y_j}\}_{1\leq j\leq |\Phi^2_{att_i}|}\}_{1\leq i\leq |\Lambda_{policy}|}$ using Φ_{att_i} in Ω.
- DO sends c, Δ , and Λ_{policy} to CSP.

2. A new user (gid) requests witnesses for the attributes he/she has. This process is privately run by DO. It first validates the user's eligibility of data access. If the user is eligible, the process adds the user to an accumulator of each non-revoked attribute. DO then updates attribute witnesses for all users in those changed accumulators. Δ is also re-constructed at the end.

   The outputs of the process are the following:

   - Attribute witnesses in the updated accumulators are sent to the users to be kept secretly.
   - Updated Δ is sent to CSP. It can be public.

The detailed steps are as follows:

- User gid contacts DO and with attribute set $UAtt_{gid} = \{att_i\}_{1 \leq i \leq |UAtt_{gid}|}$.
- DO first authenticates the user and verifies the attributes in $UAtt_{gid}$.
- If the user is authenticated, DO issues the witnesses as follows:
  - For every non-revoked $att_i \in UAtt_{gid}$:
    - Call $UserAttributeManager(gid,$ "*Add*", $\Phi_{att_i})$ and have $\{w_\alpha, \alpha, \Phi_{att_i}\}$ returned.
    - If $\alpha^2 \neq \{\}$:
      - Compute the external identifier $g^{\alpha^4} = g^y$;
      - For each $gid_i \in \alpha^2$, send gid_i the witness update of attribute att_i: $\{att_i, w_{gid_i}, g^y\}$.
    - Replace the old Φ_{att_i} with the new Φ_{att_i} in Ω.
- DO re-constructs $\Delta = \{att_i, \{v_j^{y_j}, g^{y_j}\}_{1 \leq j \leq |\Phi^2_{att_i}|}\}_{1 \leq i \leq |\Lambda_{policy}|}$ using Φ_{att_i} in Ω.
- DO sends Δ to the CSP to replace the previous one.

3. A new user (gid) contacts AA for an ABE's private key. This is the private process run by AA.

   The output of the process is one of the following:

   - usk_{abe} is sent to the user and kept secretly if the user is authenticated.
   - The output of AA is sent to the user otherwise.

The detailed steps are as follows:

- The user contacts AA with the required input parameters.
- AA authenticates the user:
  - If the user can be authenticated, AA:
    - Call $KeyGen_{abe}(\cdot)$ and generate usk_{abe}.
    - Send usk_{abe} to the user.
  - Otherwise the output depends on the output of AA.

4. A user (gid) retrieves ciphertext for decryption. This is a process run by CSP and a user. The user contacts the CSP with his/her user gid to retrieve ABE ciphertext c. The CSP re-encrypts c to $c^{'}$ using a randomly selected key. The re-encryption key is embedded into the data structure Ψ. The user is provided with $c^{'}$ and Ψ that are stored in $C^{'}$. If the user has all the required attributes that have not been revoked, the re-encryption key can be re-constructed. Once the user gets the ABE ciphertext c, the user calls Dec_{abe} to decrypt the data.

   The output of the process is one of the followings:

   - Data m if the user has all the required witnesses and a valid ABE private key.
   - The output of Dec_{abe} otherwise.

   The detailed steps are as follows:

   - User gid contacts CSP to request ABE ciphertext c with gid.

- CSP calls $ReEnc(c,\ \Lambda_{policy},\ \Delta,\ gid)$ and returns $C^{'} = \{ce(g,\ g)^{s},\ \Psi\}$ to the user.
- Let $uw_{gid} = \{\{att_i\ w_{gid},\ g^{y_j}\}\}_{1 \leq i \leq \left|UAtt_{gid}\right|}$ denote the container having all the attribute witnesses for gid.
- The user calls $Dec(C^{'},\ uw_{gid})$. If uw_{gid} has all the required and valid attribute witnesses, the ABE ciphertext c is returned; otherwise a random string is returned.
- User gid calls $Dec_{abe}(c,\ \cdot)$ to decrypt m if c is correctly returned and usk_{abe} is able to decrypt c according to the access policy of the ABE. Otherwise the output is the output of Dec_{abe}.

5. DO revokes an attribute of user gid. This is the private process run by DO. The process takes the user's gid and the attribute att to be revoked as the inputs. It first removes the gid from an accumulator for att, then updates the rest of users in the accumulator with new attribute witnesses. It reconstructs Δ and sends it to CSP.

   The outputs of the process are as follows:

   - Attribute witnesses in the updated accumulator are sent to the users and kept secretly.
   - Re-constructed Δ is sent to CSP and can be public.

   The detailed steps are as follows:

   - DO extracts the attribute accumulator container Φ_{att} of att from Ω.
   - DO calls $UserAttributeManager(gid,\ \text{“}Delete\text{”},\ \Phi_{att})$ and have $\{w_{\alpha},\ \alpha,\ \Phi_{att}\}$ returned.

- If $\alpha^2 \neq \{\}$, DO
  - Compute the external identifier $g^{\alpha^4} = g^y$.
  - For each $gid_i \in \alpha^2$, send the updated witness: $\{att, w_{gid_i}, g^y\}$ to user gid_i.
- DO replaces the old Φ_{att} with the new Φ_{att} in Ω.
- DO re-constructs $\Delta = \{att_i, \{v_j^{y_j}, g^{y_j}\}_{1 \leq j \leq |\Phi^2_{att_i}|}\}_{1 \leq i \leq |\Lambda_{policy}|}$ using Φ_{att_i} in Ω.
- Finally, DO sends Δ to the CSP to replace the previous one.

6.3.4 Security Analysis

The goal of the generic attribute revocation system is to prevent users from using their revoked attributes to decrypt ABE ciphertexts after revocations. As the revocation capability is built on top of an ABE scheme and leverages a dynamic accumulator (DA) scheme ([15]) to achieve dynamic and anonymous user revocation, the fundamental security of the system first relies on the security of the selected ABE scheme and DA. We assume that:

- The ABE scheme is at least CPA-secure.
- The DA scheme is secure against any forgeability of the witnesses and provides the anonymity protection for elements aggregated into the accumulators.

The attribute revocation capability aims to equip an ABE scheme with two additional security features:

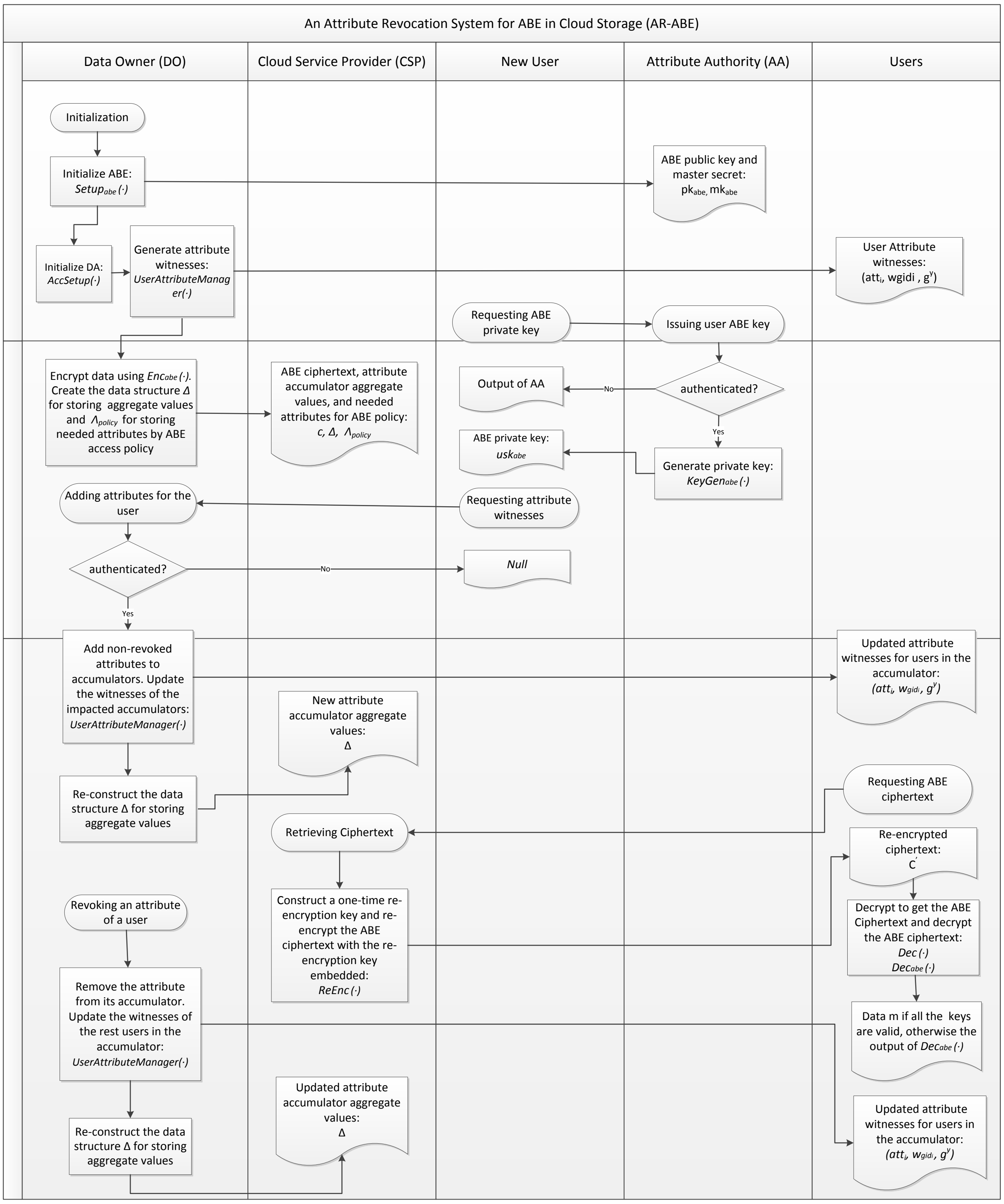

An Attribute Revocation System for ABE in Cloud Storage (AR-ABE)
Data Owner (DO)
Cloud Service Provider (CSP)
New User
Attribute Authority (AA)
Users
Initialization
Initialize ABE: Setup$_{abe}$(·)
ABE public key and master secret: pk$_{abe}$, mk$_{abe}$
Initialize DA: AccSetup(·)
Generate attribute witnesses: UserAttributeManager(·)
User Attribute witnesses: (att$_i$, wgidi , g^y)
Requesting ABE private key
Issuing user ABE key
Encrypt data using Enc$_{abe}$(·). Create the data structure Δ for storing aggregate values and Λ$_{policy}$ for storing needed attributes by ABE access policy
ABE ciphertext, attribute accumulator aggregate values, and needed attributes for ABE policy: c, Δ, Λ$_{policy}$
Output of AA
No
authenticated?
Yes
ABE private key: usk$_{abe}$
Generate private key: KeyGen$_{abe}$(·)
Adding attributes for the user
Requesting attribute witnesses
authenticated?
No
Null
Yes
Add non-revoked attributes to accumulators. Update the witnesses of the impacted accumulators: UserAttributeManager(·)
Updated attribute witnesses for users in the accumulator: (att$_i$, w$_{gidi}$, g^y)
New attribute accumulator aggregate values: Δ
Re-construct the data structure Δ for storing aggregate values
Requesting ABE ciphertext
Retrieving Ciphertext
Re-encrypted ciphertext: C$^{'}$
Construct a one-time re-encryption key and re-encrypt the ABE ciphertext with the re-encryption key embedded: ReEnc(·)
Decrypt to get the ABE Ciphertext and decrypt the ABE ciphertext: Dec(·) Dec$_{abe}$(·)
Revoking an attribute of a user
Remove the attribute from its accumulator. Update the witnesses of the rest users in the accumulator: UserAttributeManager(·)
Data m if all the keys are valid, otherwise the output of Dec$_{abe}$(·)
Updated attribute accumulator aggregate values: Δ
Updated attribute witnesses for users in the accumulator: (att$_i$, w$_{gidi}$, g^y)
Re-construct the data structure Δ for storing aggregate values

Figure 6.3: Interaction Diagram of Attribute Revocation System for ABE in Cloud Storage

- Any number of attributes revoked from one user or multiple users does not impact other users using the same attribute or attributes. The users can still use their non-revoked attributes despite of their other attributes being revoked. No ABE private key is required to be re-generated after a revocation.
- Users can anonymously request re-encrypted ABE ciphertexts from CSPs.

With those in mind, the system should meet the following security requirements:

- Dynamic attribute revocation: Users are prevented from using their revoked attributes to decrypt ABE ciphertexts, even the users hold valid ABE private keys. However, this protection applies to the ciphertexts retrieved after the revocation.
- Witness unforgeability: Users should not be able to forge their attribute witnesses.
- Collusion prevention: A user cannot combine his/her attribute witnesses with other users' to successfully construct a re-encryption key.
- Anonymity: Users remain anonymous to CSPs.
- Ciphertext indistinguishability (CPA-Secure): The re-encrypted ABE ciphertext is CPA-secure against eavesdropping attacks.

Dynamic Attribute Revocation and Collusion Prevention

This security feature relies on the ciphertext re-encryption to control the access of ABE ciphertext, in particular, the inability of recovering ciphertext re-encryption keys using forged attribute witnesses. The witnesses unforgeability of DA should prevent users from forging any attribute witness.

We use the following scenario to informally analyze that a user is prevented from using revoked attribute(s) to recover any re-encryption keys. The system is assumed to be concerned with PPT adversaries.

1. Assume user gid_A has a set of attributes which can be denoted by $UAtt_{gid_A} = \{att_i\}_{1 \le i \le |UAtt_{gid_A}|}$. They are not revoked at the beginning. gid_A is given the witnesses of his/her attributes: $\{\{att_i,\, w_{gid},\, g^{y_j}\}\}_{1 \le i \le |UAtt_{gid_A}|}$.
2. Suppose user gid_A has attribute $att_j \in \Lambda_{policy}$ revoked. DO removes gid_A from the accumulator for the attribute att_j and re-generates the witnesses for the remaining users in that accumulator. DO also updates and sends CSP: $\Delta = \{att_i,\, \{v_j^{y_j},\, g^{y_j}\}_{1 \le j \le \left|\Phi_{att_i}\right|}\}_{1 \le i \le \left|\Lambda_{policy}\right|}$, which contains the accumulator's updated aggregate value.
3. User gid_A can freely ask for any number of re-encrypted ciphertexts from CSP within polynomial time. Each time the re-encryption key is randomly generated and embedded in Ψ. Since the attribute att_j for user gid_A is not a member of any accumulator in Δ after its revocation, gid_A does not have the valid witness of an accumulator for att_j.

(a) Suppose user gid_A still holds the old witness of att_j. Based on the membership unforgeability of [15], user gid_A should not be able to forge any valid witness of any accumulator if gid_A is not aggregated to any accumulator for att_j.

(b) Suppose user gid_A tries to re-construct the re-encryption key using either his old attribute witness or colluding with other users. (Assume that α_i, one of the accumulators for att_j, has gid_A removed.)

As it has been described in *Dec* algorithm, the following steps recover a share of the re-encryption key:

- The external identifier g^{y_i} provided in the witness of the attribute: $\{att_j$, $w_{gid_A} = (g^{\frac{gid_A y_i(\prod_{gid_x \in \alpha_i^2}(gid_x+\beta))}{gid_A+\beta}}, g^{gid_A+\beta}), g^{y_i}\}$ is used to locate the share in Ψ.
- Then, a share for this attribute att_j can be recovered as follows:
  1. $e(w^1_{gid_A}, w^2_{gid_A}) = e(g^{\frac{gid_A y_i(\prod_{gid_x \in \alpha_i^2}(gid_x+\beta))}{gid_A+\beta}}, g^{(gid_A+\beta)}) = e(g,g)^{gid_A y_i(\prod_{gid_x \in \alpha_i^2}(gid_x+\beta))}$
  2. $e(g,g)^{s_i - gid_A y_i(\prod_{gid_x \in \alpha_i^2}(gid_x+\beta))} \cdot e(g,g)^{gid_A y_i(\prod_{gid_x \in \alpha_i^2}(gid_x+\beta))} = e(g,g)^{s_i}$.

If user gid_A has the old witness, he/she does not have the accumulator's correct external identifier g^{y_i} as this identifier has been changed after his/her revocation. Since y_i is randomly selected from Z_p and kept secretly by DO, gid_A is not able to guess the new g^{y_i} within PPT, assuming the prime number p is large enough. Even though user gid_A can collude with other users to get the new g^{y_i}, gid_A cannot forge a valid witness of an existing accumulator under the witness unforgeability of DA.

Suppose user gid_A can get the user gid_B to share his/her valid witness of att_j to compute #1: $e(g,g)^{gid_B \cdot y_i \prod_{gid_x \in \alpha_i^2}(gid_x+\beta))}$ (Note: α_i^2 here does not contain gid_A

as his/her attribute has been revoked). But with gid_B being embedded into the share, gid_A and gid_B cannot be cancelled out in #2. Furthermore, based on the DLP assumption, A should not be able to recover the aggregate value v_i by given $e(g,g)^{gid_B y_i (\prod_{gid_x \in \alpha_i^2}(gid_x+\beta))}$ (#1) within polynomial time. Therefore, A cannot collude with other users to reconstruct the re-encryption key.

Let us assume that gid_A is the only one removed from α_i for att_j. Given g^{y_i}, gid_A can not extract y_i under the DLP assumption. Therefore gid_A cannot replace the old y_i with the new y_i in his/her $w^1_{gid_A}$.

(c) As re-encryption key is a one-time randomly selected key, the key and shares are different each time the ABE ciphertext is requested. If gid_B can provide gid_A a share of att_j from his/her retrieved ciphertext, the share cannot work with the rest of shares gid_A has to recover the ciphertext re-encryption key. The previous re-encryption keys recovered by gid_A also cannot decrypt the newly retrieved ciphertext.

Based on the above analysis, the system achieves dynamic user attribute revocation and collusion prevention.

Witness Unforgeability

This protection is assumed to be provided by the scheme [15] that DA is built upon.

Anonymity:

The anonymity protection is the same one as in the user revocation system, UR-CSP (see Section 5.4.2.3). We omit the analysis here.

Ciphertext Indistinguishablilty (CPA-Secure):

Ciphertext sent over the various channels can be eavesdropped by an adversary. Although the adversaries can obtain the ABE public keys, all the re-encryption keys are one-time keys that are randomly and uniformly generated. Assume that the selected ABE scheme is CPA-secure. Intuitively the re-encrypted ciphertext should remain CPA-secure.

The user attribute revocation control is implemented on top of an ABE scheme which is treated as black boxes and "wrapped" by the revocation layer. ABE private keys are generated independently from the attribute witnesses. Although an adversary may be given valid witnesses for their attributes, if those attributes do not comply with the ABE access policy, the adversary should not be able to decrypt an ABE ciphertext due to the security of the underlying ABE scheme. The control layer only manages user access to the ABE ciphertext.

Based on the above analysis, we conclude that the proposed attribute revocation system, AR-ABE, has met the security requirements.

6.4 Attribute Revocation Overhead Analysis

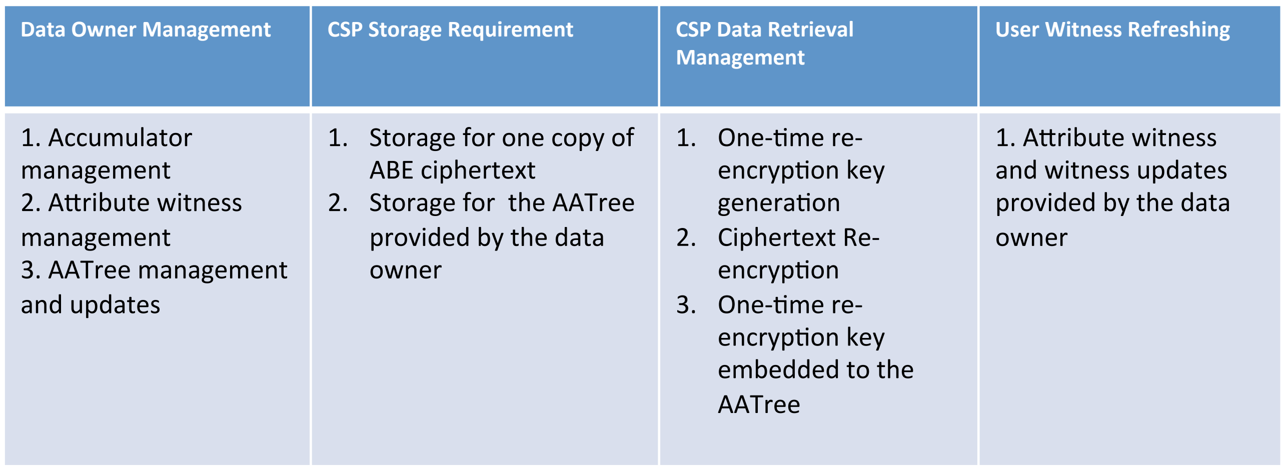
To the best of our knowledge, we are not aware any similar system that can generically work with any existing ABE scheme. The analysis below is only pertaining to our system.

Adding the attribute revocation system to an ABE scheme introduces the following overheads:

1. Accumulator management
2. Attribute witness updates and management
3. One time encryption key generation and ciphertext re-encryption

Data owners are responsible for managing item 1 and 2. As was discussed in our user revocation systems (see Section 6.3), the overheads associated with items 1 and 2 potentially have scalability issues if a user's attributes or status is changed frequently. Delegating management tasks to a trusted entity and servers can dramatically improve the scalability. The third overhead is at CSPs and takes place whenever a CSP fulfills a data retrieval request. This can also become a bottleneck during a peak time of data retrieval. However, parallel processing in re-encrypting ciphertext and constructing the data structure to carry the re-encryption key can help ease the bottleneck.

Figure 6.4 shows the overheads incurred by different entities in the system:



Data Owner Management	CSP Storage Requirement	CSP Data Retrieval Management	User Witness Refreshing
1. Accumulator management 2. Attribute witness management 3. AATree management and updates	1. Storage for one copy of ABE ciphertext 2. Storage for the AATree provided by the data owner	1. One-time re-encryption key generation 2. Ciphertext Re-encryption 3. One-time re-encryption key embedded to the AATree	1. Attribute witness and witness updates provided by the data owner

Figure 6.4: Overheads of the Attribute Revocation System (AR-ABE)

6.5 Conclusions

In this chapter we propose an attribute revocation system (AR-ABE) for ABE schemes. Our system is generic and can be directly applied to any ABE scheme without the need to modify the underlying ABE scheme. The revocation is dynamic. It neither requires the ABE scheme to re-issue user private keys, nor requires it to re-encrypt the message for any revoked attribute. There is no limitation on the number of attributes to be revoked from a user or multiple users. AR-ABE also protects user privacy and is thus practical for deployment in untrusted cloud storage environments.

Chapter 7

An Anonymous User Revocation Model for ABS in Cloud Storage

Contents

Attribute-based signature (ABS) schemes allow a party to sign a message with fine-grained control over a set of defined attributes. When an ABS scheme is used for access control in a cloud storage environment, a data owner specifies an access policy defined by system attributes.

This access policy is also referred to as the *predicate*. The CSP is given the predicate and its verification key. During an authentication process, the CSP challenges a user by giving the user a piece of data to sign. The user uses his/her private key to generate a signature of the data based on the given predicate. The CSP validates the signature using the verification key. The signature can be verified if the user has the attributes complying with the predicate. Since the user owns his/her private key, the user is authenticated if the signature is validated. On the other hand, the CSP gains no more knowledge other than the fact that a signature is generated by a user who has attributes satisfying the predicate.

Since attributes are not associated with any user's identity, an ABS scheme provides anonymous access control over an expressive fine-grained policy. It seems to be a promising access control method for use in untrusted cloud storage environments. However, like ABE, basic ABS schemes have fundamental issues in regard to dynamic user revocation, especially anonymous user revocation.

We propose a model (AUR-ABS) for ABS schemes to enable an anonymous and dynamic user revocation feature. The AUR-ABS model is flexible and can be applied to different ABS schemes with minimum modifications to the underlying ABS schemes. The content of this chapter has been published in [143].

7.1 Related Work

Dynamic user revocation for ABS has been investigated in the literature. Escalate et al. [49] proposed a revocable ABS scheme, assigning a randomly selected identifier to each user in addition to their attributes. An external entity keeps a secret verification key and a list of identifiers for the revoked users. The verification key can trace a signature to its signer to identify the signatures generated by revoked users. However, this revocation method relies on an external trust entity with a verification key. The verification process conflicts with the unlinkability and anonymity properties of ABS schemes.

Cao et al. [44] proposed a user revocation scheme that also uses an external party (a *mediator*). In their scheme, the AA issues two parts of attribute private keys for each user. One part is sent to the user, and the other part is given to the mediator. Every signature consists of two parts. One part is generated by a user, and the other part comes from the mediator. A revoked user cannot obtain the signature portion from the mediator. Letting the mediator keep every user's partial private key increases the workload on key management and distribution tasks. This approach also introduces performance and scalability overheads.

Lian et al. [87] proposed a user revocation scheme using timestamps. Each user's private key is tagged with a timestamp via a time attribute. The time attribute is added to the user's private key. The predicate also includes an attribute for the expiration time. User attribute private keys are therefore updated periodically. Revoked users are prevented from obtaining new private keys. Periodic key re-issuing cannot immediately revoke users. It also has a potential performance impact.

User revocation has also been studied in a special type of ABS scheme called an *Attribute-Based Group signature* (ABGS) scheme. We can view ABGS schemes as a special type of group signature (GS) scheme [36] that consists of members who generate signatures corresponding to a predicate. Khader [79] constructed a revocable ABGS scheme. However, this scheme relies on the group manager to link a signature to a signer, so the signature can be verified and revoked if that is the case. The approach also violates the unlinkability of ABS.

Applying ABS to data protection in a cloud environment has also been studied. Ruj et al. [115] designed a privacy preserving authentication scheme that was built in [81] and [91]. Zhao et al. [150] proposed a data sharing protocol using ABS to anonymously identify readers and writers who can modify and re-encrypt data. Liu et al. [89] used ABS to construct an attribute-based proxy signature system. This allows a data owner to delegate signature generation to a cloud proxy. Li et al. [86] leverage the computing resource of the cloud to propose a mechanism that allows a signer to outsource intensive computation during the signing process.

7.2 Our Contributions

Most of the existing user revocation proposals for ABS take one of the follow approaches:

- Periodically updating user private keys. This approach generates a new master secret periodically, and reissues user private keys to all non-revoked users. It is simple and straightforward, but potentially very expensive and not instantaneous [87].

- Using a third party to identify a revoked user or a signature generated by a revoked user. This approach requires a signature being linked to a user [44, 79], which violates the anonymity property of ABS schemes.

Ideally any revocation approach should retain all the ABS properties. The revocation approach should meet the following requirements:

1. Dynamic: Any user can be instantaneously revoked. The revocation process does not require regeneration of the master secret or user signing keys.
2. Anonymous: A verifier, regardless of whether trusted or not, should be able to anonymously identify whether a given signature is generated by a revoked user or not. This verification process does not need to know user identities and have a trusted party as a mediator in the process.

As unlinkability is one of the core properties of an ABS scheme, it makes user revocation hard to meet the anonymous requirement. Although a number of schemes have been proposed to achieve dynamic user revocation, none of them meets the anonymous requirement.

We propose a user revocation model for ABS schemes, referred to as the *ABS-AR* model. This model makes the following contributions:

- It meets the two defined requirements: dynamic and anonymous.

- It is sufficiently flexible and requires only minor changes to an ABS scheme.

Since each ABS scheme is differently constructed, we will illustrate our AUR-ABS model on the scheme of Li et al. [83].

7.3 Anonymous User Revocation Model (AUR-ABS)

To meet the anonymous user revocation requirements, we introduce a witness attribute that will be included in every user's attribute set. This attribute is also "AND"ing with the predicate of an ABS scheme. A data owner (DO) centrally manages this attribute and issues each user a unique witness private key (referred as witness key) for this attribute.

To issue witness keys of the introduced attribute, we split the ABS master secret into two parts. One part of the secret is used by the ABS AA to issue user signing (private) keys. This key issuing process remains the same as it is in the original ABS construction. The other part of the secret is used by DO to derive witness keys. Thus, each user is given two kinds of private keys: an ABS signing key and a witness key. Both keys are used to produce signatures. Users are also identified by their unique global identifiers (*gid*s). These *gid*s are randomly selected, but not linked to the true identities of users. They are used to tie the two types of private keys together uniquely for each user to prevent collusions. A revoked user cannot get his up-to-date witness private key and is not able to generate a valid signature immediately after revocation.

For simplicity, the details of user authentication to AA and DO have been omitted in our description. The authentication and user *gid* validation are orthogonal to this AUR-ABS model.

7.3.1 Trust Model

The AUR-ABS trust model consists of four entities: AA, CSP, users, and DO, as shown in Figure 7.1. The entities and their trustworthiness are the same as they are defined in the trust model of Section 5.4.3.1.

7.3.2 Algorithm Definition and Construction

7.3.2.1 Data Structure Definition

The data structure for accumulators

The data structures used by DA algorithms are the same as they are defined in Section 5.3.2.1.

Although we will build the AUR-ABS model into the scheme of Li et al. [83] (described in Section 4.2.3), we will follow the commonly used notations in this thesis, which means that some of the denotations or notations used in [83] will be changed without impacting the original scheme.

Table 7.1 lists the commonly used notation in this chapter.

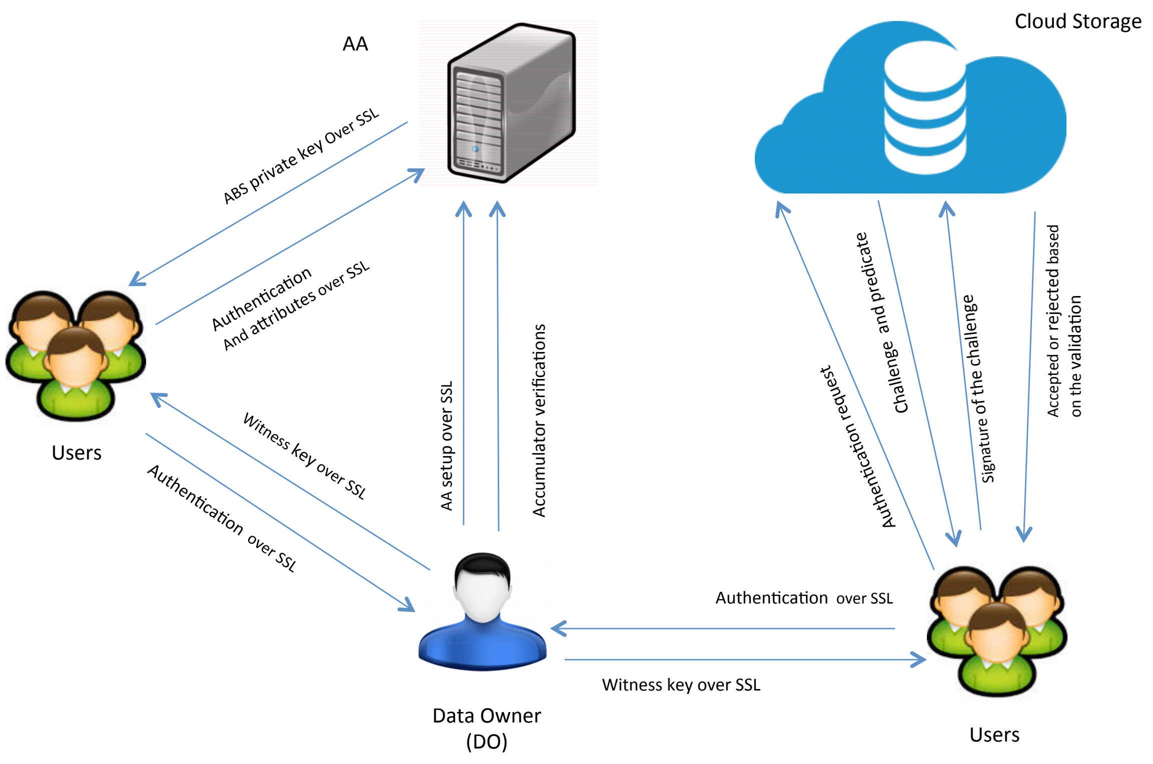


Figure 7.1: Trust Model of Anonymous User Revocation Model for ABS in Cloud Storage

Notation	Description
Att	All the attributes in the system. $\|Att\|$ donates the total number of attributes.
d	The minimum number of attributes for generating a signature
$dAtt$	A default set of d - 1 attributes that are required to generate the signature
$UAtt_{gid}$	An attribute set of user *gid*
usk_{gid}	User *gid*'s signing key issued by AA
mk_{abs}	A portion of the master secret used by AA to generate user signing keys
mk_{acc}	A portion of the master secret used by ABS-AR model to derive witness (private) keys
Δ	A data structure for storing accumulator aggregate values
Ops	The operational type - *Add* or *Delete* a *gid* to or from an accumulator
ω_{α}	The witness container for storing witnesses of users in the accumulator α
w_{gid}	The accumulator witness of user *gid*

Table 7.1: Anonymous User Revocation Model Notation

7.3.2.2 Algorithm Definition and Construction

We construct the algorithms and build the model into the scheme in [83]. The AUR-ABS model adds *IndAcc* algorithm to the scheme. Changes to the algorithms of an ABS scheme are demonstrated using the scheme in [83]. Whether those algorithms can be run privately or publicly, and which information is kept secretly or can be public will be described in the system construction section.

- *Setup*(1^n): The algorithm takes the input 1^n as security parameter and proceeds as follows:
  - Let Att contain all the attributes in the system and d be the minimum number of required attributes for generating a signature.
  - Let p be an n-bit prime.

- Use the security parameter 1^n to generate a bilinear map e: $G \times G \rightarrow G_1$.
- The following steps work in the same way as in *Setup* algorithm of [83]. The only change made is to split the master secret into two parts: mk_{abs} and mk_{acc}.
  * Map each attribute in Att to an element in Z_p uniquely: $Att = \{att_i\}_{1 \leq i \leq |Att|}$.
  * Let $dAtt = \{att_i\}_{1 \leq i \leq d-1}$ be a d - 1 default attribute set in Z_p.
  * Select a random generator: $g \in G$, and two random numbers uniformly: x_1, x_2 $\in Z_p^{\star}$.
  * Let $mk_{abs} = x_1$ and mk_{acc}= x_2.
  * Let $mk = mk_{abs} + mk_{acc}$ and set $g_1 = g^{mk}$.
  * Pick a random element $g_2 \in G$ and compute $Z = e(g_1, g_2)$.
  * Choose two hash functions H_1, H_2: $\{0, 1\}^* \rightarrow G$.
- Initialize accumulator algorithms:
  * Call *AccSetup*(G, g) (Algorithm 5.1) to set up DA. It returns the system parameters: k, β, A_k, Φ.
- Finally, output:
  * ABS verification key: $params_{abs} = (G, G_1, g, g_1, g_2, Z, d, H_1, H_2)$.
  * Master secrets: mk_{abs} and mk_{acc}.
  * DA's system parameters: $\{k, \beta, A_k, \Phi\}$.

• *KeyGen*(mk_{abs}, $UAtt_{gid}$, gid): This algorithm inputs master secret mk_{abs}, a user's attribute set $UAtt_{gid}$, and gid. It issues ABS signing key as it does in [83], except that user gid is embedded into $q(0)$ for preventing collusions:

- Choose a d - 1 degree polynomial and set $q(0) = mk_{abs}$ - gid.
- Generate a new attribute set $UAtt^{'}_{gid} = UAtt_{gid} \cup dAtt$.
- For each $i \in UAtt^{'}_{gid}$, randomly choose $r_i \in Z_p$ and compute: $d_{i0} = g_2^{q(i)}(H_1(i))^{r_i}$ and $d_{i1} = g^{r_i}$.
- Output user's signing key usk_{gid}: $\{D_i = (d_{i0},\, d_{i1})\}_{1 \leq i \leq |UAtt^{'}_{gid}|}$, where $|UAtt^{'}_{gid}|$ is the number of attributes in $UAtt^{'}_{gid}$.

• $IndAcc(mk_{acc},\ gid,\ Ops,\ \Phi,\ \beta)$: This algorithm is added to insert ABS-AR model into the scheme. It inputs master secret mk_{acc}, a user's gid, the operation type *Ops,* the accumulator container Φ, and the accumulator trapdoor β. It generates the witness private keys as follows:

- If Ops == "*Add*", call *AccAdd*(gid, Φ) (Algorithm 5.2) to add gid to an accumulator: $\{\alpha,\, \Phi\} = Add(gid,\, \Phi)$.
- If Ops == "*Delete*", call *AccDelete(gid,* Φ*)* (Algorithm 5.3) to remove gid from an accumulator: $\{\alpha,\, \Phi\} = AccDelete(gid,\, \Phi)$.
- Set $witKeys = \{\}$.
- If $\alpha^2 \neq \{\}$, do the following:
  * Select a random number: $y^{'} \leftarrow Z_p$.
  * Set $\alpha^4 = y^{'}$ and update α in Φ.
  * Compute $g^{y^{'}}$ as the new external identifier of α.
  * For each $gid_i \in \alpha^2$:

- · Randomly select $r_i \leftarrow Z_p$.
- · Compute $wsk_{w_{gid_i}} = \{wsk^1_{gid_i} = g_2^{\frac{r_i(\prod_{gid_x \in \alpha^2}(gid_x+\beta)+mk_{acc}+gid_i)}{gid_i+\beta}}, wsk^2_{gid_i} = g^{\frac{(gid_i+\beta)}{r_i}}$, $wsk^3_{gid_i} = g^{y'}\}$.
- · Add the witness private key to $witKeys$: $witKeys \leftarrow witKeys + \{gid_i$, $wsk_{gid_i}\}$

We embed gid_i in the witness private key for collusion prevention.

– Return $witKeys$.

- $Sign(m,\ usk_{gid},\ wsk_{gid},\ \Upsilon_{k,Att^\star},\ UAtt_{gid},\ d)$: This algorithm inputs message m, ABS signing key usk_{gid}, witness key wsk_{gid}, predicate $\Upsilon_{k,Att^\star}$, attributes $UAtt_{gid}$, and the number of attributes d required in the signature. It generates a signature consists of two portions: the portion generated by [83] remains exactly the same as in the original scheme, and the portion generated by AUR-ABS model is ϑ. Although both portions are generated independently, they are linked by a user gid which has been embedded to usk_{gid} and wsk_{gid}.

  – Select a subset with k number of attributes: $Att' = UAtt_{gid} \cap Att^\star$; (Note: $\Upsilon_{k,Att^\star}$ specifies the attribute set Att^* and the threshold value k).

  – Select a default attribute subset $dAtt' \subseteq dAtt$ so that $\left|dAtt'\right| = d - k$.

  – Randomly select $\gamma_i' \in Z_p$ for all $att_i \in Att' \cup dAtt'$.

  – Randomly select $s \in Z_p$.

  – Compute $\sigma_0 = \left[\prod_{i \in Att' \cup dAtt'} d_{i0}\right]\left[\prod_{i \in Att^\star \cup dAtt'} (H_1(i))^{\gamma_i'}\right] H_2(m)^s$, $\{\sigma_i = d_{i1} g^{\gamma_i'}\}_{i \in Att' \cup dAtt'}$,

$$\{\sigma_i = g^{\gamma_i^{'}}\}_{i \in Att^{\star}/Att'} \text{ and } \sigma_0^{'} = g^s.$$

- Compute $\vartheta = \{\ \vartheta_0 = (wsk_{gid}^1)^s = g_2^{\frac{sr_i(\prod_{gid_x \in \alpha^2}(gid_x+\beta)+mk_{acc}+gid)}{gid+\beta}},$

  $$\vartheta_1 = (wsk_{gid}^2)^{\frac{1}{s}} = g^{\frac{(gid+\beta)}{sr_i}})\}\ .$$

- Finally, output the signature: $\sigma = \{\sigma_0,\ \{\sigma_i\}_{i \in Att^{*} \cup dAtt'},\ \sigma_0^{'},\ \vartheta\}$.

- $Verify(m,\ \sigma,\ params_{abs},\ \Upsilon_{k,Att^{\star}},\ v)$: The algorithm takes the inputs of message m, signature σ, ABS verification key $params_{abs}$, predicate $\Upsilon_{k,Att^{\star}}$, and the accumulator aggregate value v. It validates the signature as follows:

  - Verify the input signature σ= $\{\sigma_0,\ \{\sigma_i\}_{i \in Att^{\star} \cup dAtt'},\ \sigma_0^{'},\ \vartheta\}$ on message m over the predicate $\Upsilon_{k,Att^{\star}}$, (for attribute set $Att^{\star} \cup dAtt^{'}$) by checking whether the following equation holds:

$\frac{e(g,\sigma_0)e(\vartheta_0,\vartheta_1)}{\left[\prod_{i \in Att^{\star} \cup dAtt'} e(H_1(i),\sigma_i)\right] e(H_2(m),\sigma_0^{'})e(g_2,v)} \overset{?}{=} Z$. (Note: only two components: $e(\vartheta_0,\ \vartheta_1)$ and $e(g_2,\ v)$, are added by ABS-AR to the equation.)

(7.3.1)

  - Output "*Accept*" if the above equation holds; otherwise output "*Reject*".

7.3.3 Correctness

The above equation (8.3.1) can be verified as follows:

$$\frac{e(g,\sigma_0)e(\vartheta_0,\vartheta_1)}{\prod\limits_{i \in \omega^{\star} \cup \Omega'} e(H_1(i),\sigma_i)e(H_2(m),\sigma_0^{'})e(g_2,v)}$$

$$=\frac{e(g,\left[\prod_{i\in Att'\cup dAtt'} d_{i0}\right]\left[\prod_{i\in Att^{\star}\cup dAtt'} (H_1(i))^{\gamma_i'}\right] H_2(m)^s)e((wsk_{gid}^1)^s, g^{\frac{(gid+\beta)}{sr_i}})}{\left[\prod_{i\in Att^{\star}\cup dAtt'} e(H_1(i),\sigma_i)\right] e(H_2(m),g^s)e(g_2,v)}$$

$$=\frac{e(g,\left[\prod_{i\in Att'\cup dAtt'} g_2^{q(i)}H_1(i)^{r_i}\right]\left[\prod_{i\in Att^{\star}\cup dAtt'} (H_1(i))^{\gamma_i'}\right] H_2(m)^s)e((wsk_{gid}^1)^s, g^{\frac{(gid+\beta)}{sr_i}})}{\left[\prod_{i\in Att^{\star}\cup dAtt'} e(H_1(i),\sigma_i)\right] e(H_2(m),g^s)e(g_2,v)}$$

$$=\frac{e(g,g_2^{x_1-gid}\left[\prod_{i\in Att'\cup dAtt'} (H_1(i))^{r_i+r_i'}\right]\left[\prod_{i\in Att^{\star}/dAtt'} (H_1(i))^{\gamma_i'}\right])e(g,H_2(m)^s)e((wsk_{gid}^1)^s, g^{\frac{(gid+\beta)}{sr_i}})}{\left[\prod_{i\in Att'\bigcup dAtt'} e(H_1(i),g^{r_i+r_i'})\right] e(g,\left[\prod_{i\in Att^{\star}/Att'} (H_1(i))^{\gamma_i'}\right])e(H_2(m),g^s)e(g_2,v)}$$

$$=\frac{e(g,g_2^{x_1-gid})e(g,\left[\prod_{i\in Att'\cup dAtt'} (H_1(i))^{r_i+r_i'}\right])e(g,\left[\prod_{i\in Att^{\star}/Att'} (H_1(i))^{\gamma_i'}\right])e(g,H_2(m)^s)e((wsk_{gid}^1)^s, g^{\frac{(gid+\beta)}{sr_i}})}{e(\left[\prod_{i\in Att'\bigcup dAtt'} e(H_1(i)^{r_i+r_i'},g)\right] e(g,\left[\prod_{i\in Att^{\star}/Att'} (H_1(i))^{\gamma_i'}\right])e(H_2(m),g^s)e(g_2,v)}$$

$$=\frac{e(g,g_2^{mk_{abs}-gid})e(g_2^{\frac{sr_i(\prod_{gid_x\in\alpha^2}(gid_x+\beta)+mk_{acc}+gid)}{gid+\beta}}, g^{\frac{(gid+\beta)}{sr_i}})}{e(g_2, g^{(\prod_{gid_x\in\alpha^2}(gid_x+\beta))})}$$

$$=\frac{e(g,g_2^{mk_{abs}-gid})e(g_2,g)^{\prod_{gid_x\in\alpha^2}(gid_x+\beta)+mk_{acc}+gid}}{e(g_2,g)^{(\prod_{gid_x\in\alpha^2}(gid_x+\beta))}}$$

$$=e(g,g_2)^{mk_{abs}-gid}e(g,g_2)^{mk_{acc}+gid}$$

$$=e(g,\ g_2)^{mk_{abs}-gid+mk_{acc}+gid}$$

$$=e(g,\ g_2)^{mk}$$

$$=e(g_1,\ g_2)$$

$= Z.$

7.3.4 System Description

Figure 7.2 shows the interactions between a user, DO, AA, and CSP. Some ABS schemes may have multiple AAs. It makes no difference in regard to adopting AUR-ABS model. We can generically refer to AA regardless. The system in a cloud storage environment works as follows:

1. System setup. This is a private process run by DO. It takes in a security parameter 1^n and initializes system parameters, ABS scheme, and DA. It then adds non-revoked users into accumulators, issuing witness keys to users.

   The outputs of the process are the following:

   - The versification key $params_{abs}$ and a portion of master secret mk_{abs} are sent to AA. $params_{abs}$ is public. mk_{abs} is kept secretly.
   - Witness keys are sent to users and kept secretly.
   - $\mathit{\Delta}$, which contains aggregate values tagged with their external identifiers, is sent to the CSP. It can be public.

   The detailed steps are as follows:

   - DO calls Setup(1^n) to initiate the ABS scheme and DA. The process returns the ABS verification key $params_{abs}$, master secrets mk_{abs} and mk_{acc}, and DA's system parameters $\{k, \beta, A_k, \Phi\}$.

- DO provides AA with $params_{abs}$, and mk_{abs}.
- Let U contain non-revoked *gid*s.
- For each user $gid_i \in U$, DO adds gid_i to a group: $RtWitKeys = IndAcc(mk_{acc}$, gid_i, "*Add*", Φ, $\beta)$.
- If $RtWitKeys \neq \{\}$:
  - For each gid_i in $RtWitKeys$ $(=\{gid_i,\ wsk_{gid_i}\}_{1 \leq i \leq |RtWitKeys|})$, DO sends the witness key wsk_{gid_i} to the user.
- Using Φ, DO creates $\Delta = \{v_i,\ g^{y_i}\}_{1 \leq i \leq |\Phi|}$, where v_i is the ith accumulator aggregate value α_i^1, and g^{y_i} is its external identifier ($\alpha_i \in \Phi$).
- DO sends $\mathit{\Delta}$ to CSP.

2. A new user (*gid*) requests a witness key. This process is privately run by DO. It first validates the user's eligibility. If the user is eligible, the process will add the user to an accumulator and update users of the accumulator with new witness keys. It also updates the CSP with new aggregate values.

   The outputs of the process are as follows:

   - $\perp$ if the user is not authenticated or verified.
   - Otherwise, the updated witness keys are sent to users to be kept secretly, and the updated $\mathit{\Delta}$ is sent to CSP. $\mathit{\Delta}$ can be public.

   The detailed steps are as follows:

- The user gid contacts DO for his/her witness key.
- If DO can authenticate the user and validate gid, DO proceeds as follows:
  - Add gid to an accumulator: $RtWitKeys = IndAcc(mk_{acc}, gid, \text{“}Add\text{”}, \Phi, \beta)$.
  - If $RtWitKeys \neq \{\}$, then:
    - For each gid_i in $RtWitKeys$, send the witness key wsk_{gid_i} to the user.
    - Create a new $\Delta = \{v_i, g^{y_i}\}_{1 \leq i \leq |\Phi|}$
    - Send Δ to CSP to replace the previous one.
- Otherwise DO returns $\perp$ to the user.

3. A new user (gid) requests an ABS signing key. This is the private process run by AA.

   The output of the process is one of the following

   - The user's signing key usk_{abs} is sent to the user and kept secretly if user is authenticated.
   - The output of AA is sent to the user otherwise.

   The detailed steps are as follows:

   - A user contacts AA with gid and $UAtt_{gid}$ for an ABS signing key.
   - If the user is authenticated, AA calls $KeyGen(mk_{abs}, UAtt_{gid}, gid)$ and sends usk_{gid} to the user.
   - Otherwise AA returns its output.

4. A user authenticates to a CSP. This is a process run by the CSP and a user. The CSP requires the user to make a signature on a randomly generated message. The user signs the message and sends the signature back to the CSP. The CSP can verify and detect whether the signature is valid or signed by a revoked user.

   The output of the process is one of the following:

   - "*Accept*" if the signature is valid and generated by a non-revoked user.
   - "*Reject*" otherwise.

   The detailed steps are as follows:

   - CSP randomly generates a message m and sends m and predicate $\Upsilon_{k,Att^\star}$ to the user.
   - The user calls $Sign(m, usk_{gid}, wsk_{w_{gid}}, \Upsilon_{k,Att^\star}, UAtt_{gid}, d)$ to generate the signature σ_m.
   - The user sends σ_m and g^y to CSP, where g^y is the external identifier of *gid*'s witness key.
   - CSP uses g^y to locate v in $\mathit{\Delta}$.
   - If v can be found, CSP calls $Ret = Verify(m, \sigma_m, params_{abs}, \Upsilon_{k,Att^\star}, g^y, v)$.
     - If Ret = "*Accept*", the user is authenticated.
     - Otherwise the user is rejected.
   - CSP send Ret to the user.

5. DO revokes user (*gid*). This process is privately run by DO. It removes the user from

an accumulator and updates other user new witness keys. It also updates CSPs with the new aggregate value.

The outputs of the process are as follows:

- Updated witness keys are sent to users and kept secretly.
- New $\mathit{\Delta}$ is sent to CSPs. $\mathit{\Delta}$ can be public.

The detailed steps are as follows:

- DO removes gid: $RtWitKeys = IndAcc(mk_{acc}, gid, \text{“}Delete\text{”}, \Phi, \beta)$.
- If $RtWitKeys \neq \{\}$, then:
  - For each gid_i in $RtWitKeys$, send witness private key $wsk_{w_{gid_i}}$ to the user.
  - Create a new $\mathit{\Delta} = \{v_i, g^{y_i}\}_{1 \leq i \leq |\Phi|}$
  - Send $\mathit{\Delta}$ to CSP to replace the previous one.

7.4 Security Analysis

The AUR-ABS model enables dynamic user revocation and anonymous signature verification for ABS schemes. However, this model has to be built into an ABS scheme. We assume that the ABS scheme adopting AUR-ABS model meets the security properties of ABS:

- Collusion Resistance
- Unforgeability

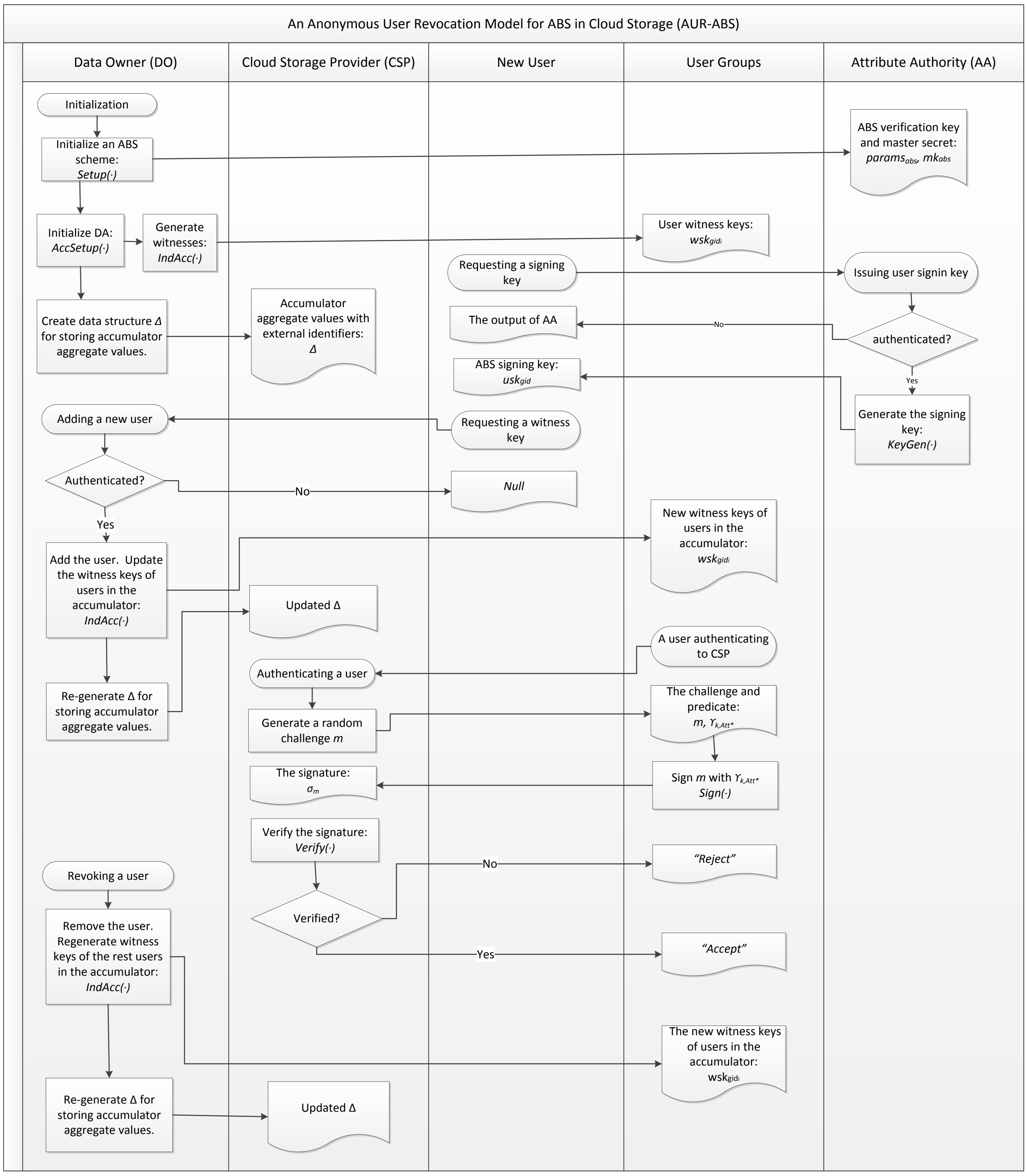

An Anonymous User Revocation Model for ABS in Cloud Storage (AUR-ABS)
Data Owner (DO)
Cloud Storage Provider (CSP)
New User
User Groups
Attribute Authority (AA)
Initialization
Initialize an ABS scheme: Setup(·)
ABS verification key and master secret: $params_{abs}$, mk_{abs}
Initialize DA: AccSetup(·)
Generate witnesses: IndAcc(·)
User witness keys: wsk_{gid_i}
Requesting a signing key
Issuing user signin key
Create data structure Δ for storing accumulator aggregate values.
Accumulator aggregate values with external identifiers: Δ
The output of AA
No
authenticated?
Yes
ABS signing key: usk_{gid}
Generate the signing key: KeyGen(·)
Adding a new user
Requesting a witness key
Authenticated?
No
Null
Yes
New witness keys of users in the accumulator: wsk_{gid_i}
Add the user. Update the witness keys of users in the accumulator: IndAcc(·)
Updated Δ
A user authenticating to CSP
Authenticating a user
Re-generate Δ for storing accumulator aggregate values.
Generate a random challenge m
The challenge and predicate: m, Υ_{k,Att^*}
The signature: σ_m
Sign m with Υ_{k,Att^*} Sign(·)
Verify the signature: Verify(·)
No
"Reject"
Revoking a user
Verified?
Yes
"Accept"
Remove the user. Regenerate witness keys of the rest users in the accumulator: IndAcc(·)
The new witness keys of users in the accumulator: wsk_{gid_i}
Re-generate Δ for storing accumulator aggregate values.
Updated Δ

Figure 7.2: Interaction Diagram of Anonymous User Revocation Model for ABS in Cloud Storage

- Anonymity (unlinkability)

With those in mind, the security goals of the AUR-ABS model equip an ABS scheme with three additional features:

- No ABS signing key needs to be re-generated because of a user revocation.
- A revoked user is not able to generate a verifiable signature with or without colluding with a non-revoked user;
- Users are anonymous to CSP in the signature verification process. No other party needs to be involved in the process.

Therefore, the security of the AUR-ABS model requires:

- Collusion Prevention: Users cannot collude with each other to produce a valid signature.
- Unforgeability: Users, including revoked users having valid ABS signing keys, should not be able to forge any valid signatures.
- Anonymity (unlinkability): The AUR-ABS model extends anonymity to user revocation for ABS schemes. Users are anonymous during the signature verification process, even for a signature generated by a revoked user. No signature can be linked to a particular user.

We assume that ABS schemes with AUR-ABS model only resist PPT adversaries.

Collusion Resistance

Collusion prevention is implemented by using user identifiers (gids) to link ABS signing keys with witness keys. We assume that AA validates users against their submitted gids prior to issuing the keys. We also assume that the underlying ABS scheme has been built with collusion prevention. That is, users are prevented from pooling their partial ABS signing keys together to generate any valid ABS signature. AUR-ABS model does not change the way how the adapted ABS scheme generates signatures. Any signature generated by an ABS scheme using an ABS signing key is intact. It is one portion of an outputted signature of $Sign$ algorithm. The other potion of the signature is generated by AUR-ABS model using a witness key. If we can stop users from combining a witness key with an ABS signing key to generate a verifiable signature, then the whole system is protected from collusion.

It is not difficult to see that the system indeed prevents users from colluding as follows:

- Suppose that user gid_1 has the ABS signing key: usk_{gid_1}: $\{D_i = (d_{i0},\, d_{i1})\}_{1\leq i\leq |UAtt'_{gid_1}|}$, and user gid_2 has the witness key: wsk_{gid_2}= $\{wsk^1_{gid_2} = g_2^{\frac{r_i(\prod_{gid_x\in\alpha}(gid_x+\beta)+mk_{acc}+gid_2)}{gid_2+\beta}}$, $w^2_{gid_2}$ $= g^{\frac{(gid_2+\beta)}{r_i}}$, $w^3_{gid_2} = g^{y'}$) .

- Suppose that user gid_1 colludes with gid_2 by using gid_1's ABS signing key and gid_2's witness key to sign a message m. Then a signature is of the form: $\sigma_m = \{\sigma_0,\, \{\sigma_i\}_{i\in\omega^*\cup\Omega'}$, $\sigma'_0,\, \vartheta\}$.

- When σ_m is verified, the signature is calculated and comes to the following result:

$e(g, g_2)^{mk_{abs} - gid_1} e(g, g_2)^{mk_{acc} + gid_2}$.

Since each *gid* is unique, $gid_1 \neq gid_2$. Therefore the two *gid*s cannot cancel each other out. At the end, the signature cannot be validated as $e(g,\ g_2)^{mk}$.

Based on the above analysis, we conclude that an ABS scheme within the AUR-ABS model prevents users from colluding.

Unforgeability

In order to simplify our analysis, we use the following game to analyze unforgeability. The game involves a challenger C and a forger F over a selected predicate $\Upsilon^*_{k,Att^\star}$. At the beginning, F submits $\Upsilon^*_{k,Att^\star}$ to be challenged.

1. *Setup:* C sets up the system by running $Setup(1^n)$ to generate the verification key $params_{abs}$, master secrets mk_{abs}, mk_{acc}, and DA's system parameters. C gives $params_{abs}$ to F, and keeps the rest information.
2. *Queries*: F is allowed to query polynomial times to the following:
   (a) Request signing keys and witness keys for a set of *gid*s with the restriction that the attributes owned by a *gid* either do not comply with $\Upsilon^*_{k,Att^\star}$, or comply with$\Upsilon^*_{k,Att^\star}$, but the *gid* is revoked.
   (b) Request a signature on a message m over predicate $\Upsilon_{k',Att'}$, or $\Upsilon^*_{k,Att^\star}$ with witness keys of revoked users. Note: $\Upsilon_{k',Att'} \neq \Upsilon^*_{k,Att^\star}$.

   (c) Make multiple calls on the choices of different m over different $\Upsilon^{'}_{k^{'},Att^{'}}$ as long as $\Upsilon^{'}_{k^{'},Att^{'}} \neq \Upsilon^{*}_{k,Att^{\star}}$; or over $\Upsilon^{*}_{k,Att^{\star}}$ as long as the witness private key is from a revoked user.

3. *Challenge*: At the end, F forges a signature $\sigma^{\star}$ on m^{*} for the predicate $\Upsilon^{*}_{k,Att^{\star}}$ with the restriction that m^{*} has never been submitted. If $\sigma^{\star}$ can be validated by $Verify$ algorithm, then F wins the game. That is F successfully breaks the *unforgeability* of an ABS scheme with AUR-ABS model in PPT.

The unforgeability of AUR-ABS relies on the possibility for an F, as described above, to be able to forge a valid signature. We use the security assumptions to analyze the possibility as follows:

Suppose F repeatedly makes requests to obtain private keys for a selected *gid*s with those restrictions. There are two possible cases for C to generate private keys as follows:

Case 1: A non-revoked user does not have the attributes satisfying the predicate: $\Upsilon_{k,Att^{\star}}$. As *gid* does not have attribute set complying with the policy, based on the security of ABS scheme, the signature portion generated by ABS scheme (σ_0, $\{\sigma_i\}_{i\in\omega^{*}\cup\Omega^{'}}$, $\sigma^{'}_0$) should not be able to verified, even though the user can generate a valid portion of signature by a valid witness key.

Case 2: A revoked user has the attributes satisfying the predicate $\Upsilon^{*}_{k,Att^{\star}}$. Since the user is removed from his/her accumulator, his/her witness private key is invalid. Based on the witness unforgeability of DA, F is not able to forge the witness of any accumulator that is not aggregated with the user's *gid*. F can try to forge a witness key as follows:

Suppose F has a revoked user's witness private key before the revocation:

$$wsk_{w_{gid_i}} = \{wsk^1_{gid} = g_2^{\frac{r_i(\prod_{gid_x \in \alpha^2}(gid_x+\beta)+mk_{acc}+gid)}{gid+\beta}}, wsk^2_{gid} = g^{\frac{(gid+\beta)}{r_i}}, wsk^3_{gid} = g^y\}$$

Under the DLP assumption, F is not able to extract $r_i(\prod_{gid_x \in \alpha^2}(gid_x + \beta) + mk_{acc} + gid)$. F may try to compute $e(wsk^1_{gid}, wsk^2_{gid}) = e(g_2, g)^{\prod_{gid_x \in \alpha^2}(gid_x+\beta)+mk_{acc}+gid}$ and still cannot extract $\prod_{gid_x \in \alpha^2}(gid_x + \beta) + mk_{acc}$ under the DLP assumption, even he can get rid of $e(g_2, g)^{gid}$.

Without being able to forge a valid witness private key, the signature portion generated by witness private key, ϑ, cannot be correctly verified.

Based on the above intuitive analysis, we conclude that an ABS scheme with AUR-ABS model meets the unforgeability requirement.

Anonymity (unlinkability):

User privacy protection of an ABS scheme makes the signature reveal nothing about the identity or the attributes of a signer beyond what is explicitly revealed by the attributes used in a predicate. The AUR-ABS model preserves this kind of user privacy and further extends it to the user revocation process.

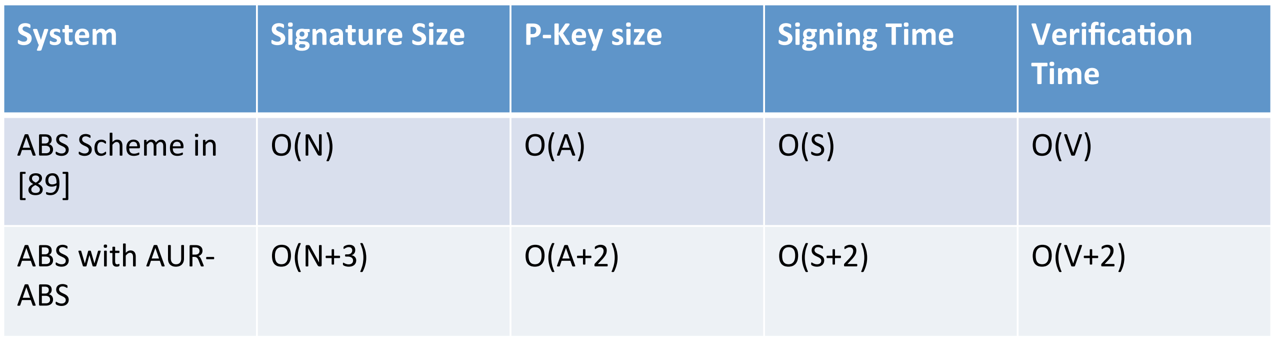
The ABS scheme in [83] is attribute-signer private. Adding AUR-ABS model does not change the way of the scheme in [83] for generating and validating signatures. It only adds an additional portion in a signature.

Let us assume the following as an example of signatures:

$\sigma = \{\sigma_0, \{\sigma_i\}_{i \in Att^{\star} \cup dAtt'}, \sigma_0', \vartheta\}$

where ϑ is the portion of the signature generated by AUR-ABS model and the rest of the signature is generated by the scheme in [83]. As described in the *Sign* algorithm, ϑ is considered to be independently generated using a witness key. User *gid*s, which do not contain any user identifies, are embedded into portions of signatures for collusion prevention. They cannot disclosed under the DLP assumption. No user identities is used or embedded in signatures and witness keys. Accumulators and witnesses based on the scheme of [15] provide anonymity protections, so does ϑ. Signature verification process and algorithms do not require any user identity information, no matter if a user is revoked or not. Further, signatures are randomized every time to make them different even though they are generated by the same user with the same witness key. In this way, signatures are unlinkable. Therefore the ABS scheme within the AUR-ABS model protects the privacy of signers and revoked signers.

Based on the above analysis, we conclude that AUR-ABS model meets the security requirements.



System	Signature Size	P-Key size	Signing Time	Verification Time
ABS Scheme in [89]	O(N)	O(A)	O(S)	O(V)
ABS with AUR-ABS	O(N+3)	O(A+2)	O(S+2)	O(V+2)

Figure 7.3: Performance Comparison of ABS and AUR-ABS

7.5 Performance Analysis

Integrating a dynamic accumulator into an ABS scheme incurs a small amount of overhead, part of which comes from the witness attribute. Figure 7.3 indicates the overheads for a typical ABS scheme adopting the revocation model. Suppose the ABS scheme in [83] has $O(N)$ size of signature, $O(A)$ size of attribute private key, $O(S)$ time of signing, and $O(V)$ time of verification. An ABS scheme with the AUR-ABS model has $O(N + 3)$ size of signature, $O(A + 2)$ size of private key (attribute private key + witness key), $O(S + 2)$ time of signing, and $O(V + 2)$ time of verification.

The other overhead comes from the accumulator management and witness key updates. To reduce the overhead of witness key updates, each user is assigned to one of n accumulators. For example, users can be grouped by regions, or their roles. As a result, the average number of updates is about n - 1 times fewer than the one for a single accumulator. Also the witness key update can be conducted out-of-band.

7.6 Conclusions

Attribute-based signature schemes seem to be a promising mechanism for fine-grained anonymous access control in untrusted cloud computing environments. We have designed and constructed a dynamic user management and anonymous revocation model (AUR-ABS) while integrating a dynamic accumulator into an ABS scheme. This AUR-ABS model is efficient and scalable. It does not use a third party in the signature verification path to identify a revoked signer. We use accumulator clusters to reduce the overheads of witness key updating and believe that AUR-ABS model is practical for cloud storage systems.

Chapter 8

Concluding remarks

My research journey started from developing a survey paper [144] of security controls and technologies in distributed storage systems. As a type of distributed storage system, cloud storage provides many potential benefits (see Section 2.5). However, cloud storage providers might not be trusted by users. This leads to concerns about data and user privacy protection, which are the focus of this thesis.

While cryptography supports data confidentiality and integrity, attribute-based cryptography provides particularly promising mechanisms for facilitating access control to protect data that has been placed in cloud storage.

In this thesis, we have considered how to make ABE and ABS more practical for providing distributed data and user privacy protection in an efficient and flexible manner in untrusted cloud storage environments.

As the security concerns about cloud storage arise from two major aspects: the trust of service providers and multi-tenant storage environments, we identified the required characteristics for using attribute-based cryptography to protect data and user privacy in untrusted cloud storage environments. We identified two major requirements:

- Flexible user and attribute management: This requires user access management to be dynamic and scalable, in particular to avoid user private key re-issuing in user and attribute revocation processes. Users are highly distributed in cloud storage environments. Re-issuing user private keys everytime when a user or an attribute is revoked has scalablility and performance issues (see Section 2.5).
- Protecting user access privacy: This requires user access privacy to be properly protected during authentication and data retrieval processes.

Although user and attribute revocations have been studied in the literature, these are generally considered with respect to particular schemes. User access privacy protection has not been previously considered. In this thesis we focused on providing general mechanisms for supporting revocation which are independent of underlying schemes, and on user access privacy protection.

The core adopted approach is to build a revocation control layer on top of an ABE or ABS scheme, so that the scheme can be treated as either a complete or partial black-box. (By the latter we mean that only minimum modifications need to be made to the underlying scheme.) A modified dynamic accumulator scheme [15] is used in our user and attribute revocation systems, which provides dynamic user and attribute management in anonymous ways.

Two dynamic user and one attribute revocation systems for ABE, and a model for anonymous user revocation for ABS have been proposed. Their security features have been analyzed against the defined security requirements.

We have shown that attribute-based cryptography can be made more practical for deployment in cloud storage environments with small additional costs. There are a number of potential areas for continuing this work. A natural next step is to implement simulations of the proposed systems and models, and assess their performance in real systems. Several other features could also be examined in order to further enhance the usability of ABE and ABS in cloud storage. Those include:

- Privacy and integrity protection of access policies in ABE schemes: In ABE, access policies are attached to ciphertexts in clear text. When data is updated, the data needs to be re-encrypted based on its access policy. This raises a concern if a user could have a malicious intent to alter the access policy and prevent legitimate users from accessing data or enable non-legitimate users to access data. When ciphertexts are stored in storage controlled by an untrusted CSP, the access policy can be changed by malicious insiders to launch a *denial-of-service* (DoS) attack. In addition, an access policy itself may be sensitive since it can leak information concerning user identities or the criticality of the data.
- Composable decryption keys: In ABE schemes, all the data is encrypted by a system public key and parameters. Decryption keys are derived from a master secret. Users who

comply with the access policy can reconstruct their decryption keys. However, it might be possible that different sets of data require different sets of encryption and decryption keys. This requires the reconstructed keys to be different. Therefore encryption and decryption keys may need to be derived or composed from multiple pairs of public and master keys. ABS schemes face the same situation.

- Non-repudiation using ABS schemes: In ABS schemes, a group of users share the same master signing key, although each user's private (signing) key is constructed uniquely from the same master key. Although this mechanism protects user privacy for generating a signature, it does not prevent users from sharing their private keys with anyone outside the group. Thus the concept of non-repudiation or accountability does not exist.

- Combining attribute-based encryption and attribute-based signature schemes together: There may be situations where a user needs to either encrypt data or make digital signatures. It is desirable that the user only needs one pair of keys, instead of two pairs of keys. However, some existing schemes, such as [51], use one pair of keys associated with an ABE scheme to encrypt data, and the other pair of keys associated with an ABS scheme to generate signatures.

It is our opinion that attribute-based cryptography is an extremely useful tool for protecting data in cloud storage environments. It is hoped that the research in this thesis has helped further this case by making its deployment more practical.

Bibliography

[1] 5 advantages and disadvantages of cloud storage. http://bigdata-madesimple.com/5-advantages-and-disadvantages-of-cloud-storage/.

[2] Amazon ebs service. http://aws.amazon.com/ebs; access date: Noverber, 2015.

[3] Amazon S3. http://aws.amazon.com/s3/; access date: April, 2015.

[4] Direct-attached storage. https://en.wikipedia.org/wiki/Direct-attached_storage; access date: April, 2015.

[5] Dropbox. https://www.dropbox.com/business; access date: April, 2015.

[6] Google Drive. http://www.google.com/drive/about.html; access date: April, 2015.

[7] A design principle for hash functions. In *Advances in Cryptology — CRYPTO' 89, Lecture Notes in Computer Science*, volume 435, pages 416 – 427. Springer, 1989.

[8] Public key infrastructures (pki) overview. https://www.sslshopper.com/public-key-infrastructure-pki-overview.html; access date: Sept, 2017, 2002.

[9] Carlisle Adams and Steve Lloyd. *Understanding PKI: concepts, standards, and deployment considerations.* Addision-Wesley Professional, 2003.

[10] Hamdan O. Alanazi, A. A. Zaidan, B. B. Zaidan, Hamid A. Jalab, and Zaidoon Kh. Al-Ani. New comparative study between DES, 3DES and AES within nine factors. *Journal of Computing*, 2:152 – 157, 2010.

[11] James P. Anderson. Computer security technology planning study. Technical report, Deputy For Command and Management Systems HQ Electronic Systems Division (AFSC), 1972.

[12] Annie Anton and Jessica D. Young. How internet users' privacy concerns have evolved since 2002. *Security & Privacy, IEEE*, 8:21 – 27, 2010.

[13] Nuttapong Attrapadung, Javier Herranz, Fabien Laguillaumie, Benoît Libert, Elie de Panafieu, and Carla Ràfols. Attribute-based encryption schemes with constant-size ciphertexts. *Theoretical Computer Science*, 422(9):15–38, March 2012.

[14] Nuttapong Attrapadung, Benoît Libert, and Elie de Panafieu. Expressive key-policy attribute-based encryption with constant-size ciphertexts. In *Public Key Cryptography, Lecture Notes in Computer Science*, volume 6571, pages 90–108. Springer, 2011.

[15] Man Ho Au, Patrick P. Tsang, Willy Susilo, and Yi Mu. Dynamic universal accumulators for DDH groups and their application to attribute-based anonymous credential systems. In *Topics in Cryptology - CT RSA, Lecture Notes in Computer Science*, volume 5473, pages 295 – 308. Springer, 2009.

[16] Amos Beimel. *Secure Schemes for Secret Sharing and Key Distribution.* PhD thesis, Israel Institute of Technology, Technion, Haifa, Israel, 1996.

[17] M. Bellare, A. Desai, E. Jokipii, and P. Rogaway. A concrete security treatment of symmetric encryption: Analysis of the des modes of operation. In *38th Annual Symposium on Foundations of Computer Science*. IEEE, 1997.

[18] Mihir Bellare and Philiph Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *First ACM Conference on Computer and Communications Security*, pages 62 – 72. ACM, 1993.

[19] Josh Benaloh and Michael de Mare. One-way accumulators: A decentralized alternative to digital signatures. In *Workshop on the Theory and Application of Cryptographic Techniques on Advances in Crypto - EUROCRYPT'93, Lecture Notes in Computer Science*, pages 274 – 285. Springer, 1993.

[20] John Bethencourt, Amit Sahai, and Brent Waters. Ciphertext-policy attribute-based encryption. In *Proceedings of IEEE Symposium on Security and Privacy - SP '07*, pages 321–334. IEEE, 2007.

[21] Ray Bird, Inder Gopal, Amir Herzberg, Phil Janson, Shay Kutten, Refik Molva, and Moti Yung. Systematic design of two-party authentication protocols. In *Advances in Cryptology - CRYPTO'91, Lecture Notes in Computer Science*, volume 576, pages 44 – 61. Springer, 1991.

[22] Dan Boneh. The decision Diffie-Hellman problem. In *the Third Algorithmic Number Theory Symposium, Lecture Notes in Computer Science*, volume 1423, pages 48 – 63. Springer, 1998.

[23] Dan Boneh, Xavier Boyen, and Hovav Shacham. Short group signatures. In *Advances In Cryptology - CRYPTO 2004, Lecture Notes in Computer Science*, volume 3152, pages 227 – 242. Springer, 2004.

[24] Dan Boneh and Matthew Franklin. Identity-based encryption from the weil pairing. In *Advances in Cryptology - CRYPTO 2001, Lecture Notes in Computer Science*, volume 2139, pages 213 – 229. Springer, 2001.

[25] Dan Boneh and Matthew Franklin. Identity-based encryption from the weil pairing. In *SIAM Journal on Computing*, volume 32, pages 586–615, 2003.

[26] Ernie Brickell, Jan Camenisch, and Liqun Chen. Direct anonymous attestation. In *the 11th ACM Conference on Computer and Communications Security*, pages 132 – 145. ACM, 2004.

[27] Jan Camenisch and Anna Lysyanskaya. Dynamic accumulators and application to efficient revocation of anonymous credentials. In *Advances in Cryptology - CRYPTO 2002, Lecture Notes in Computer Science*, volume 2442, pages 61 – 76. Springer, 2002.

[28] Ran Canetti, Oded Goldreich, and Shai Halevi. The random oracle methodology, revisited. *Journal of the ACM*, 51(4):557 – 594, 2004.

[29] Ran Canetti, Shai Halevi, and Jonathan Katz. Chosen-ciphertext security from identity-based encryption. In *Advances in Cryptology - EUROCRYPT 2004, Lecture Notes in Computer Science*, volume 3027, pages 207 – 222. Springer, 2004.

[30] Dan Cao, Xiaofeng Wang, Tianzuo Wang, and Jinshu Su. An expressive attribute-based signature scheme without random oracles. In *the 2nd International Conference on Computer Application and System Modeling*, pages 560 – 564. IEEE, 2012.

[31] Dan Cao, Baokang Zhao, Xiaofeng Wang, Jinshu Su, and Yijiao Chen. Authenticating with attributes in online social networks. In *the 14th International Conference on Network-Based Information Systems (NBiS), 2011*, pages 607 – 611. IEEE, 2011.

[32] Dan Cao, Baokang Zhao, Xiaofeng Wang, Jinshu Su, and Guofei Ji. Multi-authority attribute-based signature. In *the 3rd International Conference on Intelligent Networking and Collaborative Systems*, pages 668 – 672. IEEE, 2011.

[33] Mariana Carroll, Alta van der Merwe, and Paula Kotzé. Secure cloud computing: Benefits, risks and controls. In *Information Security South Africa (ISSA)*, pages 1 – 9. IEEE, 2011.

[34] Melissa Chase. Multi-authority attribute based encryption. In *the 4th Conference on Theory of Cryptography, Lecture Notes in Computer Science*, volume 4392, pages 515–534. Springer, 2007.

[35] Melissa Chase and Sherman S.M. Chow. Improving privacy and security in multi-authority attribute-based encryption. In *the 16th ACM Conference on Computer and Communications Security*, number 121-130. ACM, 2009.

[36] D. Chaum and E. van Heyst. Group signatures. In *Advances in Cryptology - Eurocrypt' 91, Lecture Notes in Computer Science*, volume 547, pages 257 – 265. Springer, 1991.

[37] Cheng Chen, Jie Chen, Hoon Wei Lim, Zhenfeng Zhang, Dengguo Feng, San Ling, and Huaxiong Wang. Fully secure attribute-based systems with short ciphertexts/signatures and threshold access structures. In *Topics in Cryptology – CT-RSA 2013, Lecture Notes in Computer Science*, volume 7779, pages 50 –67. Springer, 2013.

[38] Cheng Chen, Zhenfeng Zhang, and Dengguo Feng. Efficient ciphertext policy attribute-based encryption with constant-size ciphertext and constant computation-cost. In *Provable Security, Lecture Notes in Computer Science*, volume 6980, pages 84 – 101. Springer, 2011.

[39] Yanpei Chen, Vern Paxson, and Randy H. Katz. What's new about cloud computing security? Technical report, EECS Department, University of California, Berkeley, January 2010.

[40] Ling Cheung and Calvin Newport. Provably secure ciphertext policy abe. In *the 14th ACM conference on Computer and Communications security*, pages 456–465. ACM, 2007.

[41] R. Chow, P. Golle, M. Jakobsson, R. Masuoka, J. Molina, E. Shi, and J. Staddon. Controlling data in the oloud: Outsourcing computation without outsourcing control. In *ACM Cloud Computing Security Workshop (CCSW)*, pages 85 – 90. ACM, 2009.

[42] Jean-Sebastien Coron, Jacques Patarin, and Yannick Seurin. The random oracle model and the ideal cipher model are equivalent. In *Advances in Cryptology - CRYPTO 2008, Lecture Notes in Computer Science*, volume 5157, pages 1 – 20. Springer, 2008.

[43] Joan Daernen and Vincent Rijmen. *The design of Rijndael: AES the Advanced Encryption Standard.* Springer, 2002.

[44] Baokang Zhao Jinshu Su Dan Cao, Xiaofeng Wang and Qiaolin Hu. Mediated attribute based signature scheme supporting key revocation. In *the 8th International Conference on Information Science and Digital Content Technology*, volume 2, pages 277 – 282. IEEE, 2012.

[45] Alexander W. Dent. Fundamental problems in provable security and cryptography. *IACR Cryptology ePrint Archive*, 2006:278 – 292, 2006.

[46] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22:644 – 654, 1976.

[47] Keita Emura, Atsuko Miyaji, Akito Nomura, Kazumasa Omote, and Masakazu Soshi. A ciphertext-policy attribute-based encryption scheme with constant ciphertext length. In *Information Security Practice and Experience, Lecture Notes in Computer Science*, volume 5451, pages 13 – 23. Springer, 2009.

[48] Keita Emura, Atsuko Miyaji, and Kazumasa Omote. A dynamic attribute-based group signature scheme and its application in an anonymous survey for the collection of attribute statistics. In *International Conference on Availability, Reliability and Security*, pages 487 – 492. IEEE, 2009.

[49] Alex Escala, Javier Herranz, and Paz Morillo. Revocable attribute-based signatures with adaptive security in the standard model. In *Progress in Cryptology - AFRICACRYPT 2011, Lecture Notes in Computer Science*, volume 6737, pages 224 – 241. Springer, 2011.

[50] Michael Factor, Kalman Meth, Dalit Naor, Ohad Rodeh, and Julian Satran. Object storage: the future building block for storage systems. In *Local to Global Data Interoperability - Challenges and Technologies*, pages 119 – 123. IEEE, 2005.

[51] Martin Gagné, Shivaramakrishnan Narayan, and Reihaneh Safavi-Naini. Threshold attribute-based signcryption. In *Security and Cryptography for Networks, Lecture Notes in Computer Science*, volume 6280, pages 154 – 171. Springer, 2010.

[52] A. Ge, C. Ma, and Z. Zhang. Attribute-based signature scheme with constant size signature in the standard model. *IET Information Security*, 6:47 – 54, 2012.

[53] Aijun Ge, Cheng Chen, Chuangui Ma, and Zhenfeng Zhang. Short and efficient expressive attribute-based signature in the standard model. *IACR Cryptology ePrint Archive*, (125), 2012.

[54] Garth A. Gibson and Rodney Van Meter. Network attached storage architecture. *Communications of the ACM*, 43:37 – 45, 2000.

[55] Joel Gibson, Darren Eveleig, Robin Rondeau, and Qing Tan. Benefits and challenges of three cloud computing service models. In *Computational Aspects of Social Networks (CASoN), 2012 Fourth International Conference on Computational Aspects of Social Networks (CASoN), 2012 Fourth International Conference on Fourth International Conference on Computational Aspects of Social Networks (CASoN)*, pages 198 – 205. IEEE, 2012.

[56] David K. Gifford. Cryptographic sealing for information secrecy and authentication. *Communications of the ACM*, 25:274 – 286, 1982.

[57] Oded Goldreich. *Modern Cryptography, Probabilistic Proofs and Pseudorandomness*, volume 17. Springer-Verlag, 1998.

[58] Shafi Goldwasser and Silvio Micali. Probabilistic encryption & how to play mental poker keeping secret all partial information. In *the Fourteenth Annual ACM Symposium on Theory of Computing*, pages 365 – 377, 1982.

[59] Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28:270–299, 1984.

[60] Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing*, 17(2):281 – 308, 1988.

[61] Charles P. Wright Gopalan Sivathanu and Erez Zadok. Ensuring data integrity in storage: techniques and applications. In *The 1th International Workshop on Storage Security and Survivability*, pages 26 – 36. ACM, 2005.

[62] Vipul Goyal, Abhishek Jain, Omkant Pandey, and Amit Sahai. Bounded ciphertext policy attribute based encryption. In *Automata, Languages and Programming, Lecture Notes in Computer Science*, volume 5128, pages 579 – 591. Springer, 2008.

[63] Jens Groth and Amit Sahai. Efficient non-interactive proof systems for bilinear groups. In *Advances in Cryptology – EUROCRYPT 2008, Lecture Notes in Computer Science*, volume 4865, pages 415 – 432. Springer, 2008.

[64] Jinguang Han, Willy Susilo, Yi Mu, and Jun Yan. Privacy-preserving decentralized key-policy attribute-based encryption. *IEEE Transactions on Parallel and Distributed Systems*, 23:2150 – 2162, 2012.

[65] Anthony Harrington and Christian Jensen. Cryptographic access control in a distributed file system. In *the 8th ACM symposium on Access control models and technologies*, pages 158 – 165, 2003.

[66] Myagmar Hasan, Lee, and Yurcik. Toward a threat model for storage systems. In *the 2005 ACM workshop on Storage security and survivability*, pages 94 – 102. ACM, 2005.

[67] Brian Hayes. Cloud computing. *Communications ACM*, 51:9 – 11, 2008.

[68] Javier Herranz, Fabien Laguillaumie, Benoît Libert, and Carla Ràfols. Short attribute-based signatures for threshold predicates. In *Topics in Cryptology – CT-RSA 2012, Lecture Notes in Computer Science*, volume 7178, pages 51 – 67. Springer, 2012.

[69] Javier Herranz, Fabien Laguillaumie, and Carla Ràfols. Constant size ciphertexts in threshold attribute-based encryption. In *Public Key Cryptography - PKC 2010, Lecture Notes in Computer Science*, volume 6056, pages 19 – 34. Springer, 2010.

[70] Junbeom Hur and Dong Kun Noh. Attribute-based access control with efficient revocation in data outsourcing systems. *IEEE Transactions on Parallel and Distributed Systems*, 22: 1214–1221, 2011.

[71] Luan Ibraimi, Qiang Tang, Pieter Hartel, and Willem Jonker. Efficient and provable secure ciphertext-policy attribute-based encryption schemes. In *Information Security Practice and Experienc, Lecture Notes in Computer Science*, volume 5451, pages 1 – 12. Springer, 2009.

[72] Sonia Jahid, Prateek Mittal, and Nikita Borisov. Easier: Encryption-based access control in social networks with efficient revocation. In *Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security*, pages 411 – 415, 2011.

[73] Antoine Joux. A one round protocol for tripartite Diffie-Hellman. In *Algorithmic Number Theory, Lecture Notes in Computer Science*, volume 1838, pages 385 – 393. Springer, 2000.

[74] Antoine Joux and Kim Nguyen. Separating decision Diffie-Hellman from Diffie-Hellman in cryptographic groups. *Journal of Cryptology*, 16:239 – 247, 2003.

[75] Pascal Junod and Alexandre Karlov. An efficient public-key attribute-based broadcast encryption scheme allowing arbitrary access policies. In *Proceedings of the Tenth Annual ACM Workshop on Digital Rights Management*, pages 13 – 24, 2010.

[76] M. Karchmer and A. Wigderson. On span programs. In *The 8th Annual Structure in Complexity Theory Conference*, pages 102 – 111. IEEE, 1993.

[77] Jonathan Katz and Yehuda Lindell. *Introduction to Modern Cryptography, Second Edition.* Chapman and Hall/CRC, 2014.

[78] Len Seligman Ken Smith and Vipin Swarup. Everybody share: The challenge of data-sharing systems. *Computer*, pages 54 – 61, 2008.

[79] Dalia Khader. Attribute based group signature with revocation. *IACR Cryptology ePrint Archive*, (241), 2007.

[80] Allison Lewko, Amit Sahai, and Brent Waters. Revocation systems with very small private keys. In *2010 IEEE Symposium on Security and Privacy*, pages 273 – 285. IEEE, 2010.

[81] Allison Lewko and Brent Waters. Decentralizing attribute-based encryption. In *Advances in Cryptology – EUROCRYPT 2011, Lecture Notes in Computer Science*, volume 6632, pages 568–588. Springer, 2011.

[82] Allison B. Lewko, Tatsuaki Okamoto, Amit Sahai, Katsuyuki Takashima, and Brent Waters. Fully secure functional encryption: Attribute-based encryption and (hierarchical)

inner product encryption. In *Advances in Cryptology – EUROCRYPT 2010, Lecture Notes in Computer Science*, volume 6110, pages 62–91. Springer, 2010.

[83] Jin Li, Man Ho Au, Willy Susilo, Dongqing Xie, and Kui Ren. Attribute-based signature and its applications. In *the 5th ACM Symposium on Information, Computer and Communications Security*, pages 60 – 69, 2010.

[84] Jin Li, Qiong Huang, Xiaofeng Chen, Sherman S. M. Chow, Duncan S. Wong, and Dongqing Xie. Multi-authority ciphertext-policy attribute-based encryption with accountablility. In *ACM Symposium on Information, Computer and Communications Security - ASIACCS'11*, pages 386 – 390, 2011.

[85] Jin Li and Kwangjo Kim. Attribute-based ring signatures. *Cryptology ePrint Archive, Report 2008*, (394), 2008.

[86] Jingwei Li, Duncan Wong, Jin Li, Xinyi Huang, and Yang Xiang. Secure outsourced attribute-based signatures. *IEEE Transactions on Parallel and Distributed Systems*, 25: 3285 – 3294, 2014.

[87] Yanling Lian, Li Xu, and Xinyi Huang. Attribute-based signatures with efficient revocation. In *the 5th International Conference on Intelligent Networking and Collaborative Systems (INCoS)*, pages 573 – 577. IEEE, 2013.

[88] Huang Lin, Zhenfu Cao, Xiaohui Liang, and Jun Shao. Secure threshold multi authority attribute based encryption without a central authority. In *Progress in Cryptology -*

INDOCRYPT 2008, Lecture Notes in Computer Science, volume 5365, pages 426 – 436. Springer, 2008.

[89] Ximeng Liu, Jianfeng Ma, Jinbo Xiong, Tao Zhang, and Qi Li. Personal health records integrity verification using attribute based proxy signature in cloud computing. In *Internet and Distributed Computing Systems, Lecture Notes in Computer Science*, volume 8223, pages 238 – 251. Springer, 2013.

[90] Hemanta K. Maji, Manoj Prabhakaran, and Mike Rosulek. Attribute-based signatures: Achieving attribute-privacy and collusion-resistance. *IACR Cryptology ePrint Archive*, (328), 2008.

[91] Hemanta K. Maji, Manoj Prabhakaran, and Mike Rosulek. Attribute-based signatures. In *Topics in Cryptology – CT-RSA 2011, Lecture Notes in Computer Science*, volume 6558, pages 376 – 392. Springer, 2011.

[92] Erika McCallister, Erika McCallister, and Karen Scarfone. Guide to protecting the confidentiality of personally identifiable information (pii). NIST Special Publication 800-122, April 2010.

[93] Kevin McCurley. The discrete logarithm problem. In *Symposia in Applied Mathematics*, pages 49 –74. American Mathematical Society, 1990.

[94] Peter Mell and Timothy Grance. The NIST definition of cloud computing. http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf, October 2009; access date: April, 2015.

[95] Alfred Menezes, Tatsuaki Okamoto, and Scott A. Vanstone. Reducing elliptic curve logarithms to logarithms in a finite field. *IEEE Transactions on Information Theory*, 39: 1636 – 1646, 1993.

[96] Mike Mesnier, Gregory R. Ganger, and Erik Riedel. Object-based storage. *IEEE Communications Magazine*, 41:84–90, August 2003.

[97] Gerome Miklau and Dan Suciu. Controlling access to published data using cryptography. In *the 29th International Conference on Very Large Data Bases*, pages 898 – 909. ACM, 2003.

[98] H. Gilbert Miller and John Veiga. Cloud computing: Will commodity services benefit users long term? *IT Professional; IEEE Computer Society*, 11(6):57 – 59, Nov - Dec 2009.

[99] Moni Naor and Moti Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *the 22nd Annual ACM Symposium on Theory of Computing*, pages 427 – 437, 1990.

[100] Lan Nguyen. Accumulators from bilinear pairings and applications. In *Topics in Cryptology – CT-RSA 2005, Lecture Notes in Computer Science*, volume 3376, pages 275 – 292. Springer, 2005.

[101] Ventzislav Nikov and Svetla Nikova. New monotone span programs from old. *IACR Cryptology ePrint Archive*, (282), 2004.

[102] Tatsuaki Okamoto and Katsuyuki Takashima. Fully secure functional encryption with general relations from the decisional linear assumption. In *Advances in Cryptology – CRYPTO 2010, Lecture Notes in Computer Science*, volume 6223, pages 191–208. Springer, 2010.

[103] Tatsuaki Okamoto and Katsuyuki Takashima. Efficient attribute-based signatures for non-monotone predicates in the standard model. In *Public Key Cryptography – PKC 2011, Lecture Notes in Computer Science*, volume 6571, pages 35 – 52. Springer, 2011.

[104] Tatsuaki Okamoto and Katsuyuki Takashima. Decentralized attribute-based signatures. In *Public-Key Cryptography – PKC 2013, Lecture Notes in Computer Science Volume*, volume 7778, pages 125 – 142. Springer, 2013.

[105] Rafail Ostrovsky, Amit Sahai, and Brent Waters. Attribute-based encryption with non-monotonic access structures. In *the 14th ACM Conference on Computer and Communications Security*, pages 195 – 203. ACM, 2007.

[106] Bryan Parno, Mariana Raykova, and Vinod Vaikuntanathan. How to delegate and verify in public: Verifiable computation from attribute-based encryption. In *Theory of Cryptography, Lecture Notes in Computer Science*, volume 7194, pages 422 – 439. Springer, 2012.

[107] Birgit Pfitzmann and Michael Waidner. Composition and integrity preservation of secure reactive systems. In *the 7th ACM Conference Computer and Communications Security*, pages 245–254. ACM, 2000.

[108] M. Pirretti, P. Traynor, P. McDaniel, and B. Waters. Secure attribute-based systemsibute-based systems. In *the 13th ACM Conference on Computer and Communications Security*, pages 99 – 112. ACM, 2006.

[109] Charles Rackoff and Daniel R. Simon. Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. In *Advances in Cryptology — CRYPTO '91, Lecture Notes in Computer Science*, volume 576, pages 433 – 444. Springer, 1991.

[110] Peter Reiher. File profiling for insider threats. Technical report, Department of Computer Science, California University, Los Angelas, Febuary 2002.

[111] K. Ren, C. Wang, and Q. Wang. Security challenges for the public cloud. In *IEEE Internet Computing*, volume 16, pages 69 – 73. 2012.

[112] R.L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, pages 120 – 126, 1978.

[113] Ronald Rivest, Adi Shamir, and Yael Tauman. How to leak a secret. In *Advances in Cryptology — ASIACRYPT 2001, Lecture Notes in Computer Science*, volume 2248, pages 554–567. Springer, 2001.

[114] Phillip Rogaway. On the role of definitions in and beyond cryptography on the role of definitions in and beyond cryptography on the role definitions in and beyond cryptography. In *Advances in Computer Science - ASIAN 2004; Lecture Notes in Computer Science*, volume 3321, pages 13–32, 2004.

[115] Sushmita Ruj, Milos Stojmenovic, and Amiya Nayak. Privacy preserving access control with authentication for securing data in clouds. In *the 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, pages 556 – 563, 2012.

[116] A. Sahai and B.Waters. Fuzzy identity based encryption. In *Advances in Cryptology – EUROCRYPT 2005, Lecture Notes in Computer Science*, volume 3494, pages 457 – 473. Springer, 2005.

[117] Amit Sahai, Hakan Seyalioglu, and Brent Waters. Dynamic credentials and ciphertext delegation for attribute-based encryption. In *Advances in Cryptology – CRYPTO 2012, Lecture Notes in Computer Science*, volume 7414, pages 199 – 217. Springer, 2012.

[118] Pierangela Samarati and Sabrina De Capitani di Vimercati. Access control: Policies, models, and mechanisms. In *Foundations of Security Analysis and Design, Lecture Notes in Computer Science*, volume 2171, pages 137–196. Springer, 2001.

[119] R. Sandberg, D. Golgberg, S. Kleiman, D. Walsh, and B. Lyon. Design and implementation or the sun network filesystem. In *USENIX Summer Technical Conference*, number 12, 1985.

[120] Siamak F. Shahandashti and Reihaneh Safavi-Naini. Threshold attribute-based signatures and their application to anonymous credential systems. In *Progress in Cryptology – AFRICACRYPT 2009, Lecture Notes in Computer Science*, volume 5580, pages 198 – 216. Springer, 2009.

[121] Adi Shamir. How to share a secret. *Communications of the ACM*, 22:612 – 613, 1979.

[122] Claude Shannon. Communication theory of secrecy systems. *Bell Systems Technical Journal*, 28:656 – 715, 1949.

[123] Karsten Sohr, Michael Drouineaud, Gail-Joon Ahn, and Martin Gogolla. Analyzing and managing role-based access control policies. *IEEE Transactions on Knowledge and Data Engineering*, 20:924 – 936, 2008.

[124] William Stallings. *Cryptography and Network Security.* Prentice Hall, 2011.

[125] Greenan Storer and Ethan L. Miller. Longterm threats to secure archives. In *Storage Security And Survivability*, pages 9 – 16. ACM, 2006.

[126] Jinshu Su, Dan Cao, Baokang Zhao, Xiaofeng Wang, and Ilsun You. ePASS: An expressive attribute-based signature scheme with privacy and an unforgeability guarantee for the internet of things. *Future Generation Computer Systems*, 33:11 – 18, 2014.

[127] Yevgeniy Vahlis Tal Malkin, Isamu Teranishi and Moti Yung. Signatures resilient to continual leakage on memory and computation. In *Theory of Cryptography, Lecture Notes in Computer Science*, volume 6597, pages 89 – 106. Springer, 2011.

[128] John Talbot and Dominic Welsh. *Complexity and cryptography an introduction.* Cambridge, 2006.

[129] E. Verheul. Self-blindable credential certificates from the weil pairing. In *Advances in Cryptology — ASIACRYPT 2001, Lecture Notes in Computer Science*, volume 2248, pages 533 – 551. Springer, 2001.

[130] Amit Sahai Vipul Goyal, Omkant Pandey and Brent Waters. Attribute-based encryption for fine-grained access control of encrypted data. In *the 13th ACM Conference on Computer and Communications Security*, pages 89 – 98. ACM, 2006.

[131] Cong Wang, Qian Wang, Kui Ren, and Wenjing Lou. Ensuring data storage security in cloud computing. In *the 17th International Workshop on Quality of Service*, pages 1 – 9. IEEE, 2009.

[132] Guojun Wang, Qin Liu, Jie Wu, and Minyi Guo. Hierarchical attribute-based encryption and scalable user revocation for sharing data in cloud servers. *Computers & Security*, 30: 320 – 331, 2011.

[133] Pengpian Wang, Dengguo Feng, and Liwu Zhang. Towards attribute revocation in key-policy attribute based encryption. In *Cryptology and Network Security, Lecture Notes in Computer Science*, volume 7092, pages 272 – 291. Springer, 2011.

[134] Wenqiang Wang and Shaozhen Chen. Attribute-based ring signature scheme with constant-size signature. *Information Security, Institution of Engineering and Technology*, 4:104 – 110, 2010.

[135] Brent Waters. Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization. In *Public Key Cryptography - PKC 2011, Lecture Notes in Computer Science*, volume 6571, pages 53 – 70. Springer, 2011.

[136] Hoeteck Wee. Efficient chosen-ciphertext security via extractable hash proofs. In *Advances in Cryptology – CRYPTO 2010, Lecture Notes in Computer Science*, volume 6223, pages 314–332. Springer, 2010.

[137] Aaron Weiss. Computing in the clouds. *NetWorker*, 11:16 – 25, 2007.

[138] Wang Wenqiang and Chen Shaozhen. An efficient attribute-based ring signature scheme. In *2009 International Forum on Computer Science-Technology and Applications*, pages 147 – 150. IEEE, 2009.

[139] Michael E. Whitman and Herbert J. Mattord. *Principles of Information Security*. Cengage Learning, 2012.

[140] J. Li X. Xie, H. Ma and X. F. Chen. New ciphertext-policy attribute-based access control with efficient revocation. In *Information and Communication Technology; Lec- ture Notes in Computer Science*, volume 7804, pages 373 – 382. Springer, 2013.

[141] Zhiqian Xu and Keith M. Martin. Dynamic user revocation and key refreshing for attribute-based encryption in cloud storage. In *the 11th IEEE International Conference on Trust, Security and Privacy in Computing and Communications*, pages 844–849, 2012.

[142] Zhiqian Xu and Keith M. Martin. A practical deployment framework for use of attribute-based encryption in data protection. In *the 15th IEEE International Conference on High Performance Computing and Communications*, pages 1593 – 1598, 2013.

[143] Zhiqian Xu and Keith M. Martin. Anonymous user revocation for using attribute-based signature in cloud computing. In *the 6th IEEE International Conference on Cloud Computing Technology and Science*, pages 358 – 365, 2014.

[144] Zhiqian Xu, Keith M. Martin, and Clifford Kotnik. A survey of security services and techniques in distributed storage systems. In *2011 International Conference on Security and Management (SAM '11)*, pages 3 – 9. Wold Congress in Computer Science, 2011.

[145] Kan Yang, Xiaohua Jia, and Kui Ren. Attribute-based fine-grained access control with efficient revocation in cloud storage systems. In *the 8th ACM SIGSAC Symposium on Information, Computer and Communications Security*, pages 523 – 528, 2013.

[146] Shucheng Yu, Kui Ren, and Wenjing Lou. Attribute-based content distribution with hidden policy. In *the 4th IEEE Workshop on Secure Network Protocols*, pages 39 – 44, 2008.

[147] Shucheng Yu, Cong Wang, Kui Ren, and Wenjing Lou. Achieving secure, scalable, and fine-grained data access control in cloud computing. In *the 29th Conference on Computer Communications*, pages 534 – 542. IEEE, 2010.

[148] Shucheng Yu, Cong Wang, Kui Ren, and Wenjing Lou. Attribute based data sharing with attribute revocation. In *the 5th ACM Symposium on Information, Computer and Communications Security*, pages 261–270, 2010.

[149] Wenying Zeng, Yuelong Zhao, Kairi Ou, and Wei Song. Research on cloud storage architecture and key technologies. In *the 2nd International Conference on Interaction Sciences: Information Technology, Culture and Human*, pages 1044 – 1048. ACM, 2009.

[150] Fangming Zhao, Takashi Nishide, and Kouichi Sakurai. Realizing fine-grained and flexible access control to outsourced data with attribute-based cryptosystems. In *Information Security Practice and Experience, Lecture Notes in Computer Science*, volume 6672, pages 83 – 97. Springer, 2011.